Imperial College London
Department of Computing

Analysing High Frame-Rate Camera Tracking

Ankur Handa

September 2013

Supervised by Prof. Andrew Davison

Submitted in part fulfilment of the requirements for the degree of PhD in Computing and the Diploma of Imperial College London. This thesis is entirely my own work, and, except where otherwise indicated, describes my own research.

Abstract

High frame-rate offers benefits of robust and accurate camera tracking for rapid motion. However, the benefits are generally understated arguing that it is not possible to operate on high frame-rates due to stringent processing budgets and that even today 10-60Hz is treated as a standard real-time frame-rate range. How exactly does the choice of a given frame-rate varies as computational budget is changed? This thesis explores the possibilities of tracking at frame-rates higher than this range and argues that the computational cost per frame in trackers that use prediction is substantially reduced when the frame-rate is increased. Additionally, considering the physics of image formation, high frame-rate implies that the upper bound on the shutter time is reduced leading to less motion blur but more noise. On the other hand, low frame-rate often leads to motion blur but reduced noise in the images. Carefully considering the scene lighting that affects the image noise and the camera motion that affects the motion blur and putting these factors together, how are application-dependent performance requirements of accuracy, robustness and computational cost optimised as frame-rate varies? We study 3D camera tracking from a known rigid model as our test problem and analyse the fundamental image alignment approach to understand the choice of frame-rate that affects tracking. We systematically investigate this via a careful synthesis of photorealistic video using ray-tracing of detailed 3D scene, experimentally obtained photo-realistic reponse and noise models and rapid camera motions and later validate the conclusions with a well-controlled real experiment. The thesis provides quantitative conclusions about frame-rate selection, fundamental connections between frame-rate and image resolution and highlights the crucial role of full consideration of physical image formation process in pushing tracking performance.

Dedication

To my parents, sister and brother for their continual love, support and encouragement.

Acknowledgements

I am very fortunate to have Professor Andrew Davison as my supervisor. He spent a lot of his time in guiding and directing my thesis. His patience, enthusiasm, and constant encouragement have given me never ending motivation and determination to overcome all the challenges that I faced on my way. I have had some wonderful and the most enjoyable interactions with him during the course of my PhD that will be much cherished. Besides that he has been my great '*karaoke buddy*' and being avid sports fans that we both are, we traded fantastic stories and opinions. This all makes me feel the most pampered student of him.

I am also grateful for the mentoring I received from my colleagues. I would like to begin with thanking Margarita who was my first colleague I interacted with and whose work I built upon in my first year. To Adrien, for all the advice, encouragement, his classic french humour, all the help during my initial years of PhD and for being a fantastic friend. I am also thankful to his girlfriend Lidia for spoiling me with macaroons. Thanks are due to many other colleagues for the time I shared and the opportunity I had, to learn from them experience. Steve, for his excellent coding skills, Hauke for his valuable suggestions and *do all possible experiments* advice. Gerardo for being a great friend. Richard with whom I had the opportunity to learn a lot during my last two years of PhD and the ideas he lent very generously that made my PhD all the more interesting. I thank him for all the late night discussions and debates. He challenged me a lot and it was always a very wonderful and rewarding experience talking to him. Renato for lending his codes very generously and suffering my unexpected evening singing in the lab. I thank Ravi for all the useful discussions I had with him, his criticism on my work, occasional weekend food outings and for being a great friend. His invaluable comments on the work made me rethink of my approach many a times. Thank you to Renzo for lending his high-speed gyro.

I also had the opportunity to interact with the new students that joined the group. I thank Jacek, Hanme, Robert and Jan for all the fruitful discussions I had with them and that improved my grip on various topics. I also thank Ping-Lin who I had the opportunity to discuss various ideas with and learn from his coding expertise. Various other colleagues at Imperial that suffered me, I would like to take this opportunity to thank them for their company that made is possible for me to enjoy my stay. In particular, I thank Ogi for being a great friend and hosting my trip to Serbia, Robin, Claire, Emma, Hatice, Brias, Anil, Bihan, Stavros and Sanjay Bilakhia. Klaus for hosting my stay at Munich. With him and Adrien it was always fun to hang out with. Their humour was just out of the extraordinary. Pablo for all the spanish lessons and improving my knowledge about Spain.

I also thank various other members of the department who helped me in many ways. I begin with Joanne (who was involved in the interview during my hiring) who was kind

enough to prepare all sorts of letters for various embassies that made it possible for me to travel to different places beginning with Sydney back in 2009 till the end of my PhD, to present my work. Thank you for your prompt replies and quick responses. Lloyd and Duncan who manage the ICT services always went out of their way to help me set up various things, fixing the computers and being very patient with me. They deserve massive thanks. Deborah too who suffered me a lot with my time-sheets filling but was always very patient to give her advice on correcting them. Thank you. Teresa and her for managing the expenses claims. Ann for managing common room on the fourth floor and for sending my posts.

Outside lab I met some fantastic people Sanjay Bhatt and Sanjay Priyadarshi for being such wonderful friends. They always listened to me patiently but suffered my humour in return. Marta and Gosia, my flatmates who I thank them profusely for being such fantastic friends. Paul Robertson for weekend meet-ups and his advice and banter. Also, to the people who run Jakobs cafe at Gloucester Road, for all the banter and food I enjoyed at their cafe regularly. My old childhood friends Udit and Jimmy (Varun) for always there for me. The friends of my undergraduate hostel, the first floor front wing and the wing-list they have maintained that is always bubbling with laughter, banter, new ideas and all sorts of daily chit-chat. Special thanks to Supreeth Achar, Ankit Garg, Akashdeep and Anand Srinivas with whom I began my learning in robotics and that sowed the seeds of research path. Thank you to Inder Rishi for the phone calls and being always there to listen and have a chat. I thank my old supervisors Prof. Jayanthi and Prof. Madhava Krishna who invested in me and gave me a great platform to begin my research with.

My last thanks are reserved for my parents who have been with me through my thick and thin, before and after. I owe them a huge debt of gratitude for almost everything. They gave me lot more freedom than necessary but always believed in me. All the qualities I have today, I have inherited from them. I am relieved that my PhD has come to an end. My younger sister '*missy*' Kanika will stop harassing me. She never lets such an opportunity go away like that. Thanks to her and my younger brother Karan for all the love. My thanks to Paruthi Uncle for his unexpected phone calls to inquire my well being and coming to meet me leaving his important work many a times. My hope is that one day they all realise how special they are to me.

Thank you to my examiners Dr. Vincent Lepetit and Dr. Tae-Kyun Kim who examined my thesis in a very short time period and their valuable corrections. I am also very fortunate for the ERC funding I received for my PhD.

Contents

CHAPTER 1

Introduction

Contents

In many real-world tasks, it is extremely important for an intelligent system or an artificial robotic agent to respond and act quickly, sometimes even faster than an animal. This could either mean preventing an accident on road, driving an autonomous car, sifting and sorting things in an industrial application, or interacting with any computing device *e.g.* typing in a mobile phone — they all need very quick responsive system. In many other situations, it has to continually deliver the updates of its location in a real-time loop to keep up with the demands of the application. Alternatively, it is the limited processing power that the robotic system is endowed with that compels its decision making to occur within a fixed time-budget.

Importantly, a real-time system always has some parameters that can be tuned so that it can perform within the fixed time budget. However, the performance varies as the budget is made tighter — the accuracy may suffer, the robustness may degrade etc. This thesis focusses on a systematic understanding of how the performance of robotic system in the context of camera tracking varies as the computational budget is made tighter as desired by an application.

1.0.1 Why Real-Time is Important?

Essence of Real-Time

Real-time operation is enforced when a system has a practical application — a robot that takes days to pick up a cup just few centimeters next to it would clearly be impractical. In essence, building a real-time system opens new doors for practical applications that can be employed in real-world scenarios. Any such real-time system employed for a practical application must then intelligently use the rich temporal coherence of the incoming data it has access to, to be able to keep on top of the application.

Consequences of Strict Timing Demands

Strict timing demands for any real-time application also require the standards of performance to be rather stiffer than for an offline application. An application that could afford an occasional failure before becomes a virtually certain failure when put in a real-time loop. For instance, when a real-time camera tracking system gets lost, it becomes nearly useless any further unless reinstated somehow to initial conditions. Moreover, the real-world environments that a robot is operating in, are very dynamic in their nature, so robot must have a mechanism to adapt itself according to the variations of the input coming from these changing environments for instance, a camera based surveillance system operating in a train station needs to be constantly performing in very dynamic and cluttered environments. Thus, the robustness demands for such applications become very crucial.

Access to Rich Temporal Coherence

Somewhat paradoxically, as real-time constraints get tighter, robustness standards are expected to improve naturally as a result of growing coherence between two consecutive incoming data (observation) samples. In fact, it is this sort of knowledge that can provide a strong prior to any real-time system that it must use intelligently to do inference within a strict timing budget. *Imagine being told to match images of the same scene but without knowing anything about their relative displacement and imagine being told that these images are consecutive frames of a video sequence.* Such domain knowledge, even if very weak, can greatly reduce the search that is carried out to infer the different state parameters it is expected to. We look into detail the domain knowledge present in the task that can greatly reduce the computational bottleneck to allow a system to operate in real-time. In the following, we outline general principles and domain knowledge present in the task and focus more on systems that are aimed to track the movements of an autonomous system.

1.1 Representations and Domain Knowledge of the Task

How should an intelligent system find its own location in an environment? How should it relate itself to the world it is operating in? These are very elementary questions that an autonomous and intelligent system has to answer most of the time when put to use in a real world application.

Humans, as intelligent machines, are doing this continuously all the time without consciously thinking about it. We actively rely on our senses for perception — our routine day to day activities involve an integrated sensor fusion to enable us to interact with the environment we work in. This could involve picking up a cup, moving a chair or walking in the street. Perception and inference based on this perception just happen. If such a system is to mimic or act faster than an animal (as demanded by the application), it must be imparted with similar or even better attributes to operate in the real world.

Most of the time, we have *a priori* knowledge of the object to a large degree of certainty — our life long information gathered over time greatly facilitates the task of understanding and interaction with the object. Similar to that, it is imperative for any intelligent system to have the ability to sense the environment or an object it is used to interact with. Ideally, it would take a sensor measurement and be able to answer instantly *where it is relative to the environment* to be able to make further plausible decisions. An *a-priori* knowledge has to be injected into the system's machinery for it to construct any inference on the location based on measurements.

In small scale and well structured environments, it is possible to provide near-ideal conditions for such a system to operate in. This could be how the environment (or a part of it) appearance beforehand together with knowledge of its own location or other state parameters that might be required depending on the application, with respect to that environment. In situations where the lack of the structure of the environment shrouds this, a continuous feed of sensory measurements have to be collated to bootstrap the system. This knowledge of environment and other parameters collectively is termed as *reference* that is crucial for any future decision the system will make. Therefore, as it operates, it can keep track of its relative movements by always comparing against this reference. Now what sort of reference this is is entirely dependent on the application. This reference could be a 3D model, it could be a reference shape or it could be a reference colour histogram or an entire image. What becomes more important is how it associates the information coming for a new movement to the available reference to keep track of its future movements.

The task of relating the measurements with the reference is formally termed as **Data Association** — a crucial step in keeping track of the movements or state parameters of the system. The domain knowledge present in the application allows us to break it down into different levels of abstraction, depending upon what kind of reference is available, for instance:

1. ***Do we already have a model of the entity we wish to track?*** This model could either be a geometric model, shape model or colour model. Further this model can either be parametrised or a non-parametric point cloud or histogram. This model could be obtained beforehand *e.g.* using a CAD model, and it remains unchanged throughout the tracking over different frames of the sequence. This has remained a subject of interest under the name *model-based tracking* or *tracking via prediction*. For instance, a useful piece of information in the form of an *a priori* knowledge of environment can greatly enhance the ability of robot to interact - the task becomes then only controlling its movements within the environment using the prior information.

2. ***Can we track without having to build the model online?*** This could either mean frame-to-frame tracking. All the necessary information comes from the previous frame to guide the tracking in the subsequent frame. This strongly hinges on the assumption that there is large coherence and continuity between the two frames and that the world or the object in the world has not changed much in the time interval between the frames. This is quite often termed as *visual odometry* or *tracking via detection* in computer vision community.

3. ***If no, do we have a mechanism to build the model online in the loop and track using this model in the new incoming frames?*** In many cases, objects and things that we wish

to track cannot be easily parametrised or defined by a collection of some templates. Although it is possible to define to a some extent objects in a structured environment but this is largely not the case and that many real-world objects are defined rather by their geometric point-cloud model. Moreover, the object or world models are not available beforehand. This necessitates to build the model on-the-fly and track using this model. It is important to remember that the model is not perfect and that it is updated in the loop online to encompass the changes and various deformations it is undergoing. This is a very popular and well studied problem called *Simultaneous Localisation and Mapping* (SLAM) or real-time *Structure from Motion* (SfM) in computer vision.

All the new information that arrives in these three cases involves associating it with the reference data. However, the real-world data is always plagued with ambiguities and that a democratic mechanism is required for a unique association of data with the reference. This all depends on how the association is formulated.

1.2 The Challenge of Robust Data Association

Before we take a more "telescopic" view of these three different paradigms of data association, it is worth taking a whirlwind tour of how the criteria of deciding the best data association given the available data is established. Any technique that we would use, we would need a criteria to evaluate the quality of data association. A search scheme can be used to then trace the transformation that yields the best quality fit.

Bayesian reductionists have greatly revelled in formalising the data association as the problem of locating the optimal point in the distribution — where optimality is informally defined to be the equilibrium point that represents the best possible result achieved given the state models of likelihood and prior. It is therefore not surprising when looked in light of the early onset of data association problem in the context of Bayesian tracking that probabilistic interpretations gained more popularity. In fact, more from the perspective of Kalman filtering that has been a standard framework to solve online in-the-loop tracking. The posterior probability or likelihood used to assess the quality of the data association are obtained via the standard Bayes Rule.

$$p(\mathcal{M}|\mathcal{O}) = \frac{p(\mathcal{O}|\mathcal{M})p(\mathcal{M})}{p(\mathcal{O})}$$

Symbols $\mathcal{O}$ and $\mathcal{M}$ denote the observations and model that is to be estimated respectively. The normalisation constant $p(\mathcal{O})$ can be ignored when we are interested in only inferring model parameters. This reduces the problem to:

$$p(\mathcal{M}|\mathcal{O}) \quad \propto \quad \overbrace{p(\mathcal{O}|\mathcal{M})}^{likelihood}\underbrace{p(\mathcal{M})}_{prior}$$

Further if it is assumed that all the observations are conditionally independent of each other (though this may not be true always), the likelihood can be factorised as

$$p(\mathcal{M}|\mathcal{O}) \quad \propto \quad \left(\prod_i p(\mathcal{O}_i|\mathcal{M})\right)p(\mathcal{M})$$

Depending on the estimator used *maximum aposteriori* or *maximum likelihood*, the best model can be obtained simply as

$$\begin{aligned} \widehat{\mathcal{M}}_{MAP} &= \arg\max_{\mathcal{M}} p(\mathcal{M}|\mathcal{O}) \\ \widehat{\mathcal{M}}_{ML} &= \arg\max_{\mathcal{M}} p(\mathcal{O}|\mathcal{M}) \qquad (1.1) \end{aligned}$$

On the other hand, there are energy based formulations that view the association problem from the perspective of physics and the mechanics involved. The core idea is again to locate the transformation parameters that yields a local minima or a point of equilibrium.

$$\widehat{\mathcal{T}} = \arg\min_{\mathcal{T}} \sum_{i=1}^{N} (R_i - C_i(\mathcal{T}))^2$$

However, they can be easily connected back the to pure Bayesian perspective because in the end they are both trying to obtain the transformation that best fits data observation C_i against the reference R_i and hence a least squares framework has direct connotation with the independent Gaussian distribution measurement model.

$$\begin{aligned} \widehat{\mathcal{T}}_{ML} &= \arg\max_{\mathcal{T}} \prod_i \mathcal{N}(R_i\,;\,C_i(\mathcal{T}),\,\sigma^2) \\ \widehat{\mathcal{T}}_{ML} &= \arg\min_{\mathcal{T}} \sum_{i=1}^{N} -\log(\mathcal{N}(R_i\,;\,C_i(\mathcal{T}),\,\sigma^2) \qquad (1.2) \\ \widehat{\mathcal{T}}_{ML} &= \arg\min_{\mathcal{T}} \sum_{i=1}^{N} (R_i - C_i(\mathcal{T}))^2 \qquad (1.3) \end{aligned}$$

We now expand on the three different aspects below.

1.3 Pure Model-Based Tracking

A collective understanding of different attributes — physical, textural or a combination of both — aid in defining the object. The physical attributes include for instance geometric shape or point cloud and textured attributes for instance, a probabilistic model of the how the colour is distributed within a given spatial window. More specific attributes of any general object are summarised in [Alexe et al., 2010]. Tracking then means finding its relative location in the continuous stream of incoming data observations provided by a sensor, using this information.

Model-based tracking begins with the assumption that it already has a knowledge of how an objects looks like and a projection of its 3D-model can be obtained as a prediction of where it is in the image for instance. We want to state upfront that in **Pure Model-Based Tracking**, the model of the object remains unchanged throughout the tracking. The prediction then guides the tracker to search for the location of the object that is present in the image. Another inherent assumption behind this is that the prediction is easily available and that it is close enough to the real location. This prediction comes continually in the loop from a Kalman Filter style estimator running in the pipeline. We describe the classical and popular methods to data association that use a known 3D model. A more up-to-date summary of pure model based tracking methods is provided by [Lepetit and Fua, 2005].

1.3.1 Textureless 3D-Model

RAPiD: A Least-Squares Approach

RAPiD [Harris and Stennet, 1990], [Harris, 1992] (Real-Time Attitude And Position Determination) was one of the first of its kind to use a model to register against an incoming image of the object in *real-time*. The CAD wire-frame model of an object to be tracked is projected in the image using previously obtained pose together with motion model as the prediction. Control points are sampled on the projected model in the image and suitable correspondences for the control points are sought. The correspondence search that looks for high intensity gradients in the image (they are most likely to represent the boundaries of the object being tracked) is performed around the predicted model location along the normals to the control points. These correspondences are then used to minimise a least square error function to obtain the transformation that brings the 3D model projection tightly onto the object. This procedure is repeatedly carried out for a number of iterations until a given convergence criteria is satisfied.

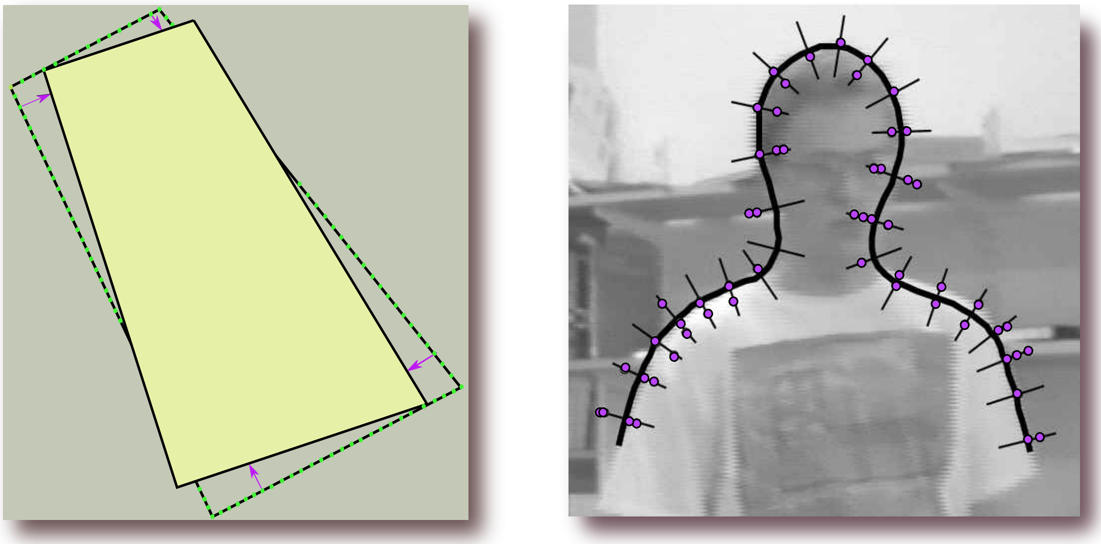
The prediction required in real-time tracking comes from the Kalman Filtering framework running in loop that fuses continuous incoming pose estimates given by the tracker. The reduced search space to look for an object offered by prediction together with clever way to search for correspondences very suitably allowed RAPiD to run at 50Hz even on moderate computing resources available that time. This has inspired a great deal of work on real-time model based tracking that operates by looking for an instance of an object in an image by searching about the predicted position coming from a state estimator than having to search the entire image by running some blanket image processing technique like edge detection as done in [Lowe, 1992] and [Gennery, 1992].

However, a major drawback of RAPiD was its fragility against background clutter since it did not have any mechanism to preclude a wrong association from contributing towards pose recovery. [Armstrong and Zisserman, 1995] detail different ambient conditions that degrade its performance and propose improvements to increase its robustness. An important of all is preventing the tracker from associating wrong pixel locations in the image with the control points on the projected model. This arises quite often when occlusions, shadows and lighting changes are present in the image. RANSAC based outlier culling together with a weighted least squares is used to obtain the final pose.

[Drummond and Cipolla, 1999a] and [Drummond et al., 2002] use an M-estimator to remove outliers in the framework to improve robustness. Their system also allows to track more complex structures than RAPiD by making the visibility of the prediction in the image with an accelerated BSP-tree [1] representation.

Notable extensions have also been proposed by [Shahrokni et al., 2004] that use the texture based edge detection replacing the 1D intensity gradient search originally employed by RAPiD and multi-modal extension of [Kemp and Drummond, 2004] and [Kemp and Drummond, 2005] that prevent the tracker from getting deceived by a local minima as is the case most often with uni-modal estimation. It is therefore worth noting that the general theory and intuitive appeal that RAPiD offers has seduced many researchers to approach the problem of textureless real-time 3D-to-2D model-based tracking in the same principled manner as RAPiD does.

[1] A *binary space partitioning* is a technique used in the computer graphics to sort the elements of an object based on their depth given a current view.



(a) RAPiD (b) CONDENSATION

Figure 1.1: **Left:** RAPiD begins the association task by projecting the available 3D model on to the image using the predicted camera pose coming from Kalman Filtering framework. The predicted image of the object is sampled at the control points and search for correspondence is performed by looking for strong edge-like gradients along the normals. These correspondences are then jointly solved for association transformation that snaps the prediction on the the real object location. All this is carried out via standard iterative least squares. **Right:** CONDENSATION follows similar principles with prediction followed by search for correspondences but is able to perform in the presence of background clutter by maintaining multi-hypothesis framework. However, unlike RAPiD this does not operate at video-rate of 30–50Hz.

CONDENSATION: A probabilistic multi-hypothesis framework

Uni-modal distributions that are used quite often to model the measurement likelihoods lack the ability to deal with false positives (a wrong data association) present in the data. As a result, a system operating in the real-world that is formalised within a purely uni-modal distribution framework fails to cope with the ambiguity present in the real data. *How should then a system deal with false positives without knowing beforehand which one of them is actually false?* A multi-hypothesis framework offers a means to cope with this ambiguity allowing all the associations to be treated in a unified framework under the assumption that only one of the associations is correct. As new observations arrive, evidences of the hypotheses changes appropriately, leading to an eventual dominant hypothesis that is most representative of the real data.

The CONDENSATION algorithm proposed in [Isard and Blake, 1996] is a popular multi-hypothesis data association framework. It is an Importance Sampling approach *à la* Particle Filtering that rigourously treats multiple data associations in a very principled stochastic manner. Each hypothesis is represented with a weight that reflects how confident is the algorithm about that hypothesis being true. These weights are propagated over time and change according to the lack or presence of confirmatory evidence in the data.

Particles are sampled from the prior distribution, that is readily available due to Bayesian filtering machinery, according to the weights and together with the measurement likelihood process that is modelled with a mixture of Gaussians, a posterior estimate is sought that is most likely to have generated the observation. Impressive results were shown tracking contours of hands, leaves and person dancing in the presence of lot of clutter. As in RAPiD, the search for correspondences is performed along the normals of the projected contour to find high intensity pixel gradient locations. In a cluttered background, many putative pixel locations that arise are wrongly associated with the contour. However, hypothesis originating from background clutter decay as object moves since they are a function of background and hypothesis coming from the object being tracked remain dominant.

Although this was demonstrated mostly in the context of tracking 2D contours in the image, such rigourous treatment of false positives in the data has paved way for many tasks, specially camera tracking, in computer vision that require a robust mechanism to deal with false copies of structures/objects that are being looked for in the image.

PWP3D: A level set approach

A recent advance in the textureless 3D model based tracking is due to [Bibby and Reid, 2008] and [Prisacariu and Reid, 2012] who formulate the problem of registration in a rather fuzzy way unlike RAPiD and CONDENSATION where data association decisions are discrete. A probabilistic colour model is used as an additional semantic cue on top of the shape prior available in the form of contour. This shape prior is embedded in signed distance transform (the sign normally identifies the inside or outside of the shape, here is used to represent the foreground or background category) that encapsulates the minimum distance of any point in the image from the contour along the normal direction. The level set of this transform gives a slice that represents all the points that are at a given distance from the contour. An exceptionally large basin of convergence offered by distance transforms allows the model to lock onto the image of object even from very poor predictions.

The distance transforms precompute the distance of any point in the domain from the contour allowing the algorithm to avoid doing an explicit line search along normals as done in RAPiD and CONDENSATION to find correspondences. However, the dependence of tracker on colour information means that the performance will degrade when foreground and background are very similar in appearance. They show the performance of tracker only when the foreground and background colours are distinct.

1.3.2 Textured 3D model

Textureless 3D model-based tracking only relies on the contour and the associated matched pixel locations to align the model to its image observation. On the other hand, tracking using a textured 3D model uses the information of the gradients of the texture in the image. It is very similar to tracking that uses the contour information as done in textureless 3D tracking the difference being the change in intensity of the pixel location in the image against the model intensity is used as an error measure instead of pure pixel location difference as used in textureless tracking. However, the principles are largely the same — that the information about alignment is largely encapsulated in the movement of edges present in the image. [Baker et al., 2004b] describe a classic extension of tracking using a textured 3D model so called Lucas-Kanade $2\frac{1}{2}$D tracking. Since this type of tracking uses more data, it is worth mentioning that it does not necessarily demand any explicit multiple hypothesis framework because of the data redundancy present in fully dense model alignment. Moreover, the tracking can run on whole image instead of looking for small object in the image as done in RAPiD, CONDENSATION and PWP3D.

A textured object in the 3D world would require a way to map the real-world texture values that are maintained in the SI units of the world into camera bitmap scale to obtain a prediction of texture in an image. However, unlike the 3D positions that can be simply projected into image using a perspective transformation, texture values have to be learnt from camera images because the absolute radiance values required to map to a pixel value for prediction are difficult to obtain as that would require to have a knowledge of full environment map of the scene as well as the view point. This brings us to the problem of recovering texture online and using that texture to align the model with incoming image observations. We postpone this to Section 1.5 where we detail this in the context of joint model building and tracking.

1.3.3 Summary

A *model-based tracking* paradigm needs prediction to constrain the search for the instance of an object in an image to allow real-time operation. This is made easier by running a state estimator in the loop that continuously provide estimates of the prediction. Therefore, real-time operation forces the system to employ prediction guided search in order to be able to stay on top of the application. However, the ability to search for the object then greatly depends on how good the prediction is. In cases of poor predictions or fast image motion, this makes the tracking quite precarious due to distractions that quite often camouflage their appearance of the true object, leading the tracker to snap onto a false copy of the object. This wrong data association can then severely affect the overall system and more importantly a wrong association when fused into the system cannot be revisited. As a result even a single failure in the tracking can bring the system to a halt as the prediction for the next frame may well completely be wrong. Multi-hypothesis framework provide the tracker the ability to deal with ambiguity that arises quite often in real world but the computational complexity involved in maintaining the hypotheses prevents the tracker from operating in real-time. On the other hand, 3D model tracking using textures requires the texture be estimated online leading to joint texture mapping and tracking. It is applicable for more advanced full image alignment as well as aligning only small 3D structures as in textureless trackers.

An essential message that model-based tracking delivers is the requirement of an *a priori* knowledge of model that is crucial for long-term drift-less tracking. In fact, this is also the major drawback because a-priori model is not always available when the object shape is not very well defined and when tracking in large scale urban environments. In the next section, we discuss another form of tracking that does not necessarily rely on the available model and instead uses the information contained in the consecutive images to track camera pose.

1.4 Tracking by Frame-to-Frame Alignment

Recursive *model-based tracking* is underpinned by assumption that tracker successfully locked onto the object in the previous frame. However, due to fast camera motion, occlusions and changes in ambient conditions model based tracking becomes prone to failures and as a result the system requires a mechanism to recover or reinstate the tracker. An alternate approach works by detecting salient features (termed *tracking-by-detection*) or uses all pixels in the two consecutive images and matches them across to obtain the relative transformation.

Such tracking method solves for the transformation from scratch every frame and as a result not as fragile as recursive model-based tracking. In vision based robotics it is also termed as pure *visual odometry*, a term that was first used in [Nistér et al., 2004] [2]. Navigation on surfaces where wheel odometry is not very reliable, for instance, a vehicle moving on uneven surfaces, visual odometry is an appealing alternative. It has also been used successfully in Mars rovers exploration project of NASA.

Since camera pose is obtained by detecting features in the previous frame and tracking them only till the next, the system is prone to drift in long term. In real-time large scale explorations maintaining a huge 3D model of a consistent map is computationally quite expensive therefore, tracking is performed in the style of pure visual odometry to obtain the camera poses continuously while mapping can run independently using an offline batch optimisation in the background. Local optimisations *àla sliding window bundle adjustment* or *global bundle adjustment* are needed to prevent the estimated trajectory obtained from pure *visual odometry* from drifting. It is important to remember that the purpose of a tracker is to deliver estimates of camera pose. Whether it uses 3D model available beforehand or being built online or pure frame-to-frame information, it is a choice purely driven by application. Following we categorise different sorts of features matching paradigms available to obtain camera pose in the context of *visual odometry*, inspired from [Scaramuzza and Fraundorfer, 2011].

1.4.1 2D-to-2D

In this approach only pixel locations and/or their intensities are matched across two or more frames to obtain the relative transformation. No explicit 3D structure is involved and the camera pose parameters are directly obtained using only image data. An outlier rejection method is employed on top in the form of RANSAC or M-Estimator (we detail more about RANSAC in Section 1.5), if required, to provide robustness to the pose estimation process.

Feature based

In a standard feature based method, salient pixel locations are detected by running a blanket detection algorithm that performs intensity tests around a local window of a pixel location

[2] The first real-time large scale single-camera pose estimation method that demonstrated the use of pure visual odometry for estimating 6-DoF camera pose unlike the filtering approaches used in small workspaces that use history of the camera trajectory

to check for the presence of corner or blob like structures. To aid matching across different frames, these pixels locations are described with a descriptor that encodes the local structure around it. Given two images detection and augmentation with descriptors is applied and putative correspondences are obtained that are later solved for relative camera pose estimate.

[Kruppa, 1913] are the first to provide solution to the relative camera pose using minimal five 2D image correspondences. However, it is [Nistér, 2004] who later gave an efficient analogue to this for a calibrated camera setting and using the minimal 5-point correspondence solution within RANSAC framework to provide robustness to the estimation from outliers. The famous direct linear transformation (DLT) based 8-point algorithm proposed by [Longuet-Higgins, 1981] has primarily been the standard method when using an uncalibrated camera. The main idea in both cases is that correspondences can be related by Essential matrix (in the calibrated setting) or Fundamental matrix (general uncalibrated setting) and that the rotation and translation of the camera can be obtained simply by SVD factorisation of the corresponding matrices.

Dense whole image based

Dense methods use all the pixels in the image contrary to the feature based method that use only limited pixel locations and as a result dense methods are able to benefit from the highly redundant set of dense measurements given provided by all pixels. Moreover, the two step procedure of feature extraction and matching is put together in one loop of global transformation recovery in an optimisation framework. An explicit outlier rejection step is handled by using a robust estimator. We mention some of the recent real-time systems that employ this method.

[Comport et al., 2007] describe a method to obtain 3D visual odometry using stereo camera. Quadrifocal constraints are used to relate the motion of pixels from reference stereo pair to the current without requiring the need to have a 3D model *i.e.* the quadri-focal tensor allows computing the transformation directly from 2D-2D image correspondences from stereo pairs. The iterative minimisation is carried out using an Efficient Second Order Minimisation (ESM) [Malis, 2004].

[Lovegrove et al., 2011] showed a system for monocular visual odometry using dense whole image alignment. This image alignment procedure is performed on the images coming from a downward looking camera mounted at the back of the car. The motion between two consecutive frames can be appropriately described by a planar homography for the road and as a result they are able to use ESM style minimisation scheme to efficiently recover camera pose.

It is worth mentioning that the trajectory estimates obtained from dense visual odometry are very competitive with the ground truth provided by GPS. These methods are also able to perform in many degrading environmental conditions where pure feature based methods would fail or yield only modicum of success. In fact, the promising effects of dense approaches have led to methods of recovering Fundamental matrix (as done in feature based methods) within the dense framework [Valgaerts et al., 2012]. Also, the availability of less computational power before was a primary reason to adopt feature based methods. However, with recent computationally powerful resources available today in GPUs, dense methods are an appealing choice.

1.4.2 3D-to-3D

In the previous section we briefly described methods that perform camera pose optimisation using 2D-to-2D correspondences. The correspondence search is made within the optimisation and largely benefits from the 2D grid structure naturally provided by the image. However, when matching 3D points there is as such no prior grid-like structure given and therefore either an $O(n^2)$ exhaustive search or search using an accelerated scheme (*e.g. kd*-trees) needs to be performed first to be able to run any optimisation tool that gives pose estimates. In which case the problem becomes more of iteratively searching for correspondence and optimising using these correspondences. We describe the camera pose estimation in a case of known correspondences (assumed to be highly reliable to a large extent) and then detail the most popular scenario when the correspondences are unknown and are searched heuristically first before running least-squares optimisation to obtain the transformation.

Known Correspondences

When 3D correspondences are known already and are highly reliable, a non-iterative algebraic solution proposed by [Arun et al., 1987] can be used to obtain the relative transformation. The objective function that measures the quality of the transformation is the standard squared-$\mathcal{L}_2$ distance between the two 3D point sets.

$$\widehat{\mathcal{T}} \quad = \quad \arg\min_{\mathcal{T}_k} \sum_i ||p_k^i - \mathcal{T}_k p_{k-1}^i||^2 \qquad (1.4)$$

They decouple the translation and rotation where the translation can be directly obtained from the difference of the centroids of the 3D points. The translation is

$$t \quad = \quad \bar{p}_k - R_k \bar{p}_{k-1} \qquad (1.5)$$

$$R \quad = \quad VU^T \qquad (1.6)$$

The rotation is obtained by algebraic manipulation of the 3D points and is returned by the SVD where $USV^T = \text{svd}((p_k - \bar{p}_{k-1})(p_k - \bar{p}_{k-1})^T)$. This is a one shot solution to obtaining the transformation than having to iteratively run least-squares optimisation. A similar method has also been used by [Maimone et al., 2007] for Visual Odomtery on mars exploration rovers with the correspondences weighted by the uncertainties.

Unknown Correspondences

In many alignment tasks correspondences are not available and have to be searched for to run a least-squares style of optimisation scheme to obtain the transformation. Therefore, the task of alignment also involves a subtask of finding appropriate correspondences first to begin with. The transformation obtained from the correspondences is greatly dependent on how good the correspondences are and as a result is directly affected by it. Therefore, an iterative alternation between correspondence search and least-squares transformation fit is performed to find the best alignment between the point sets.

The well known [Besl and McKay, 1992] proposed ICP (*iterative closest point*) has been used successfully in aligning two 3D point clouds and range images. ICP begins with putative correspondences obtained by simply looking for the closest point in the other set based on some distance metric (straightforward euclidean distance yields what is known as *point-to-point* metric). The alignment transformation is the obtained by minimising the least square cost function that relates the correspondences via the transformation that is sought. Since a closed form solution cannot be obtained for a general transformation, this correspondence search and minimise procedure is iterated until a convergence criteria is satisfied.

It is the simplicity that has made ICP quite a popular algorithm to register point clouds. [Tykkala et al., 2011] use the ICP for pure visual odometry using point-to-point metric while [Newcombe et al., 2011a] use point-to-plane metric inspired from

[Rusinkiewicz and Levoy, 2001] to do the tracking to align two different point clouds obtained from range images.

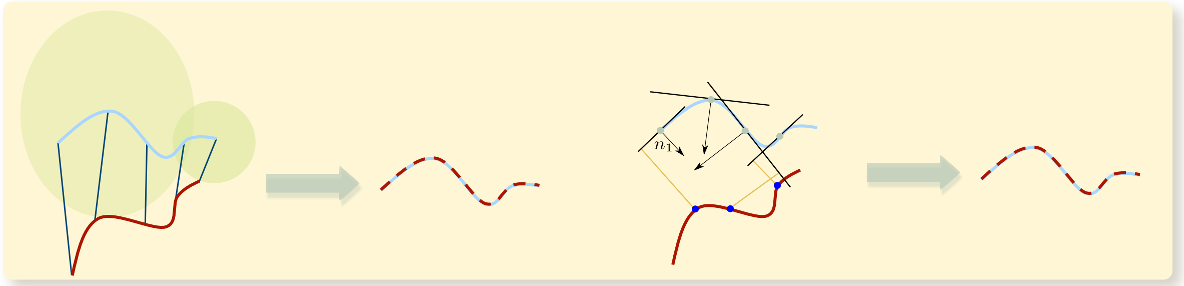


Figure 1.2: ICP begins with finding valid correspondences based on some appropriate metric. The correspondences are then related via the transformation that is being sought and in turn embedded in a conventional least squares optimisation that iteratively brings the two point clouds into alignment. The image shows two distance metrics namely the point-to-point metric and point-to-plane metric for establishing correspondences.

1.4.3 3D-to-2D

Tracking from a known 3D model can perform without drift over long term as discussed in pure 3D model tracking (cf. Section 1.3). However, working in large scale urban environments in *real-time* poses a big challenge in maintaining huge maps. As a result, only environment information within a temporal window is maintained — a subtle difference between this paradigm with model-based tracking is that the model only reflects the local structure within a small temporal window. All the past information about the structure is thrown away unlike standard Bayesian filtering approaches. A local 3D model built from past few frames is used to align with an incoming image to obtain the transformation. We again categorise the tracking into feature based and dense methods.

Feature Based

[Nistér et al., 2004] proposed one of the first real-time large scale visual odometry system that uses 3D-to-2D tracking in the loop. The front end of the system is feature detection and matching that allows to compute the relative transformation between two camera positions from where the images were taken. Features tracked over a certain number of frames provide tracks that are used to obtain the 3D positions via triangulation to get a local 3D map. These 3D positions are then projected in the new incoming images and detected features are matched against these projections. The 3D-2D associations are reminiscient of standard bundle adjustment [Triggs et al., 1999] technique used to minimise reprojection errors.

The manner in which the relative transformation is obtained is a straightforward application of Perspective *n-point* problem (P*n*P) and RANSAC based outlier culling to obtain camera transformation. A minimum of three correspondences are needed to lock down the camera transformation parameters.

Dense methods

Dense methods have recently gained more popularity due to the availability of range sensors like Microsoft Kinect and Asus Xtion Pro. An integration of range and associated texture image obtained from the RGB-D camera is jointly solved for the transformation between two consecutive frames. [Tykkala et al., 2011] show a visual odometry system that uses a bi-objective function that aims to minimise a weighted sum of the photo-consistency error of images and depth in 2D image grid. The weight is obtained on the fly from the ratio of median deviations in both error terms. They show that depth alignment when fused with photo-consistency term produces less drift compared to photo-consistency alone. In the same conference, [Steinbrucker et al., 2011] also show visual odometry using image data where photo-consistency error is related via the depth. Recently [Whelan et al., 2013] also use an integrated ICP and RGBD photo-consistency term to provide robustness to the process. They show how ICP fails when the observations are all obtained from a planar surface (*e.g.* floor) and a combined image texture and ICP are able to properly lock down the degrees of freedom of the transformation being sought.

1.4.4 Summary

The recursive nature and prediction required to perform model-based tracking has a natural advantage of making tracking easier by reducing the search space of where the observation can lie. However, it is the recursive nature that greatly makes the system susceptible to breaking when data association goes wrong due to occlusions or other scene vagaries. Visual Odometry or general frame-to-frame tracking provides an alternate route to tracking by detecting salient points of interest in the image using a standard feature detection or using all pixels and searching for correspondences in the other image. Although the search for correspondence is facilitated by an *a priori* knowledge of the temporal coherence between frames but this is still very heuristic (as done in [Nistér et al., 2004] where a heuristically chosen spatial window of 11×11 is used to search for the feature in the next frame) when compared to the active search space [Davison, 2003] naturally provided by the model-based prediction. Also, such type of tracking is prone to drift in long term since it performs matching from scratch at every frame and joint optimisation in the form of *sliding window bundle adjustment* or *bundle adjustment* is required to keep the drift in check. Next, we take a look at the tracking paradigm that uses a model that is being built online in the loop by fusing incoming observations. The difference is that this model is not very large scale as in urban environments while not too small at the time like in pure-model based tracking.

1.5 Joint Tracking and Model Building *à la* SLAM

In most indoor scenes and small scale unstructured desktop environments, 3D-model is not available beforehand and needs to be jointly built with camera poses estimation given by tracking purely from images coming from camera. Data association in this case works very similar as in model-based tracking however an important difference is that the model is not as accurate (atleast in the early stages of tracking) as CAD models and needs multiple image measurements to be integrated to obtain a sensible looking model capable of allowing pure model-based tracking in future. Therefore, data association becomes even more crucial in the pipeline. A wrong data association can then severely affect the overall system. Not only does it lead to wrong camera transformation but also corrupts the whole 3D-model because of the circular dependency of tracking and mapping on each other. As such, a wrong data association when fused into the system, cannot be revisited which can make the system very fragile.

A rich body of research has produced robust and effective methods for data association to mitigate the effects of false positives that arise in real-world images. These methods have been successfully used in model consisting of modest number of sparse features or fully dense 3D-models. In systems where the uncertainty of 3D-model can also be obtained, it can be used to assess the quality of data association as done in EKF-style systems that maintain a sparse set of features (or *landmarks*) to represent the 3D-model. On the other hand, methods that maintain fully dense models are able to obtain a prediction of full image and align it against an incoming image to obtain the camera transformation. In the following, we briefly outline the popular and fashionable data association techniques and discuss limitations and benefits of each in the context of joint tracking and mapping. Similar to previous tracking methods, we categorise them into feature based and dense whole image based.

1.5.1 Feature Based

Maximum Likelihood Data Association

The earliest and the most popular approach to data association has been the simple nearest neighbourhood test for observation and landmark that is sometimes also referred to as *Individual Compatability* (IC) or *nearest neighbour* (NN) test. In many sparse feature (or *landmarks*) based approaches where EKF machinery has greatly facilitated the joint model building and tracking, the uncertainties present in the feature positions can be used to assess the quality as well as the proximity of the feature observation pair – given the uncertainties it can be easily checked whether the observation lies inside the covariance region of the feature. In the earlier systems that used the LASER based range sensing the process was carried out directly in 3D while in vision based set-up where the 3D model is projected onto the image, the feature-observation pair consistency is checked on the 2D grid. The closest feature-observation pair based on this criteria is deemed as the match. However, this method is clearly prone to erroneous associations if there is ambiguity in the robot position or features that arises when the 3D-model of the scene gets bigger. As a result, a wrong data association greatly hampers the conventional EKF based joint model building and tracking, leading to diverging estimates of the robot position with time.

Joint Compatibility Branch and Bound

Nearest neighbour data association fails catastrophically when a wrong data association is assigned – each feature is matched independently without any joint model or batch consistency to enable mutual agreement between different associations. More specifically, NN is rendered ineffective when the distance between features is smaller than their uncertainties. In fact, mapping of some of the observations to the same landmark in the scene cannot be ruled out if each feature is matched independently as done in NN.

[Neira and Tardós, 2001] developed an approach called 'Joint Compatibility Branch and Bound' inspired by the Interpretation Tree [Grimson and Lozano-Perez, 1987] that takes care of mutual compatibility of observations. The core idea is that matches have to be jointly compatible to ensure that they maintain the probabilistic consistency of the distribution. Traversing along the tree in the depth-first ordering, JCBB, proceeds rejecting further branching of the tree if the data association hypothesis is unlikely to be correct. There is an obvious worst case scenario of testing exponential number of hypothesis but they show that many branches of the tree can easily be rejected for traversing.

Again, depending upon the sensor and the measurement model, the associations happen either in 3D if range sensor is used or 2D image grid if camera is used.

Combined Constraint Data Association

[Bailey, 2002] present a similar approach to JCBB to obtain consistent joint associations between features and their observations. All possible associations are represented as the nodes with edges between two different nodes representing the compatibility of the associations in a "Correspondence Graph". The associations are determined by nearest neighbourhood test while the compatibility is determined by the standard NIS (*normalised inverse square*) threshold test often termed as *Mahalanobis distance*. Finally, the maximum clique in the graph represents the pairings that are compatible with each other. The results of JCBB and CCDA are very similar but CCDA is also able to operate in the case when the pose of the robot is completely unknown in the map.

Scan Matching

Scan matching, an ICP-style [Besl and McKay, 1992] approach, was proposed in [Lu and Milios, 1997a, Lu and Milios, 1997b] and later [Nieto et al., 2006, Nieto et al., 2007] in the context of EKF-SLAM called Scan-SLAM. Scan-SLAM maintains the pipeline of conventional EKF-SLAM the only difference being the representation of the features. EKF-SLAM has by and large defined features as geometric identities with a well defined shapes while Scan-SLAM represents features as raw sensor scan provided by laser/sonar. The centroid of the raw sensor scan is taken to be the estimated position of the landmark. When a new measurement arrives, scans are aligned *à la* ICP to recover the association and in turn the transformation to obtain the robot position estimate.

It is worth stressing that the alignment is very different from a filter operation. The prediction that is required for alignment is merely a seed for the process and necessary to prevent the alignment from getting stuck in a local minima. Consistent pose estimation (CPE) [Gutmann and Konolige, 1999, Konolige, 2004] build their data-association using scan

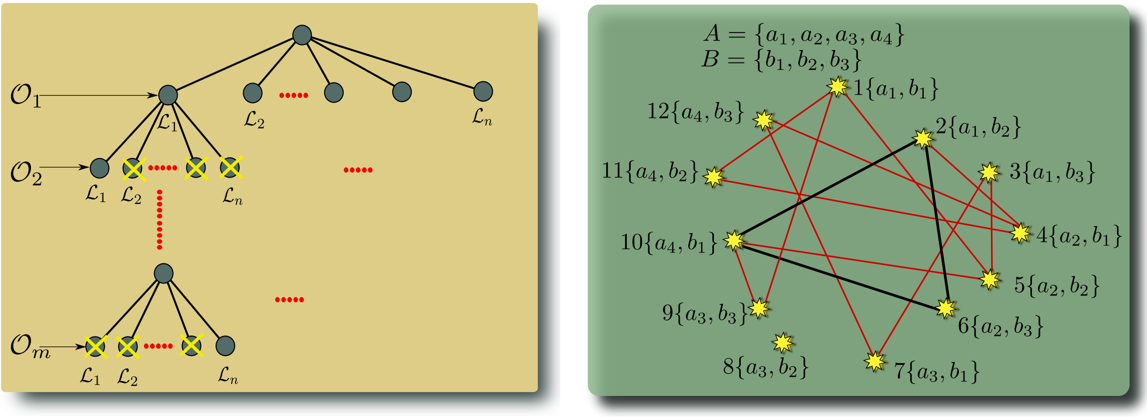


Figure 1.3: **Left**: JCBB works by assigning putative associations of landmarks ($\mathcal{L}_i$) to observations ($\mathcal{O}_i$) and collects evidence of a given association by branching further down in the interpretation tree. To avoid traversing branches that fail to yield confirmatory evidence, heuristics are employed to prevent the explosion of associations. **Right**: Combined constrained data association proposed by Tim Bailey. The nodes denote the associations while the edges denote the compatibility. Nodes that do not share an edge do not satisfy the compatibility constraint. The maximal clique in the graph represents the associations that are consistent among each other.

matching. Particle filtering approaches [Hahnel et al., 2003] and [Eliazar and Parr, 2004] also show demonstrations of scan matching in large scale map reconstructions.

Multiple Hypothesis Tracking

Multiple hypothesis tracking is a robust alternative to data association that maintains a uni-modal hypothesis for data association. The earliest MHT tracking algorithm was proposed by [Reid, 1979] who maintain multiple hypothesis coming out of different valid data associations related to particular landmark/observation. Since then they have been successively used in many other tracking algorithms too where clutter tends to distract the actual data associations. To prevent the number of hypothesis from bombarding, suitable heuristics are employed to prune weaker ones to enable efficient data association. Among others, [Nebot et al., 2003] successfully used multi-hypothesis tracking to disambiguate data association.

RANSAC and variants

RANSAC originally proposed in [Fischler and Bolles, 1981] has emerged as one of the very popular methods to model fitting on data corrupted by noise. Assuming that there is weak or little prior information about the distribution of inliers and/or noise, RANSAC runs many trials until the probability of obtaining the correct model hypothesis reaches a given pre-defined percentage threshold. It is generally assumed that correspondences have already been established although they may all not be correct. Therefore, RANSAC selects randomly the minimum number of data points required to hypothesise a model and remaining data points vote for the hypothesis being correct. The hypothesis that wins the most number of democratic votes is finally decided to be the best hypothesis returned by the algorithm. Many variants have later popped out that aim to add or use more sophisticated *a priori* information of the data into the framework to quickly obtain the correct hypothesis. In particular, [Torr and Zisserman, 2000] and [Tordoff and Murray, 2005] use prior information to guide RANSAC and propose to use likelihood probability over simple counting when collecting the votes to assess the quality of hypothesis. Also, [Chum and Matas, 2002], [Chum and Matas, 2005] and [Chum and Matas, 2008] propose a series of papers highlighting the improvements they make on top of the standard RANSAC method to efficiently arrive at a correct hypothesis. Notable also is the work of [Nistér, 2003] who proposed a different way of finding the best hypothesis using a preemptive scheme that weeds out the hypothesis contaminated by an outlier. A pre-generated set of hypothesis are progressively tested against observations in a breadth-first manner and a pruning criteria is decided that selects best hypotheses at any time t and as more observations are tested weak hypotheses are pruned resulting in the selection of best hypothesis under a given computational budget. [Botterill et al., 2009] maintain the history of failed hypothesis to assign a probability to data point reflecting how likely is that being an inlier. These weights are updated as new generated hypothesis that fail to gather enough votes. More new hypothesis are generated then by sampling from the data points that are more likely to be inliers. [Scaramuzza et al., 2009] and later [Civera et al., 2009] show 1-Point RANSAC in the context of monocular visual odometry.

Active Matching

[Chli and Davison, 2008]'s algorithm Active Matching also takes a democratic approach to hypothesis selection but the search for correspondences is made with in the loop of matching. Active matching emerges from its precursor Active Search [Davison, 2005] that performs sequential feature matching guided by Information Theory. The major difference between Active Matching and RANSAC/JCBB type approaches is that (a) *it does not match features independently and then resolve the consensus* (b) *once a feature is measured to obtain correspondence that information is used in reducing the search space for matching other features (unlike in JCBB) and as a result feature correspondences and matching are combined in one step*. In SLAM type systems, where feature correlations are also maintained, a joint probability distribution of the features locations in the image is already available. This means that measuring one feature, one should be able to reduce the uncertainty (covariance) in the location of rest of the features. This is what [Davison, 2005] showed in their paper and later [Chli and Davison, 2008] showed the utility of this approach doing real experiments with MonoSLAM [Davison, 2003]. To provide an extra layer of robustness against false positives the standard multiple hypothesis framework is used as a ready-made engine. However, the complexity and decision making in choosing the best feature to measure prohibits its usage in matching at each step large number of features. Although, the original Active Matching showed matching only a dozen of features, further improvements were made in [Handa et al., 2010] to enable matching 400 features in nearly 170*ms*. Such decide-and-measure approach though more probabilistically consistent lags behind RANSAC type methods that gain grounds on speed by making random decisions and are able to match an order of magnitude more number of features with their two stage procedure.

$$\boxed{P(x_1,x_2,x_3,.....,x_N)} = \boxed{\boxed{\boxed{\boxed{P(x_1)}P(x_2|x_1)}P(x_3|x_1x_2)}...P(x_N|x_1x_2,...,x_{N-1})}$$

Figure 1.4: The joint probability distribution can be factorised according to the Bayes rule. Active matching begins measuring x_1 and later uses that information to reduce the uncertainty of x_2. Further, as more features are measured, the uncertainties of remaining features shrink leading to more certain feature matching. The uncertainty reduction is via the update rules as mentioned in [Eustice et al., 2005].

1.5.2 Lucas-Kanade

Parametric image alignment using least squares has its roots dating nearly three decades ago in seminal paper by [Lucas and Kanade, 1981]. Since then the least square minimisations have become a common place in nearly all parametric image alignment methods. [Baker et al., 2003b], [Baker et al., 2003a], [Baker et al., 2004a] and [Baker et al., 2004b] describe all properties related to optimising the least square cost function and the class of transformations that can be recovered easily with it. In particular, [Baker et al., 2004b] describe Lucas-Kanade $2\frac{1}{2}$D approach to recover the 6-DoF camera parameters when aligning an image anchored to a 3D model against a new observation image coming from camera. This is excellently demonstrated by the recent real-time dense tracking and mapping system called DTAM [Newcombe et al., 2011b], [Meilland et al., 2011] and [Comport et al., 2011].

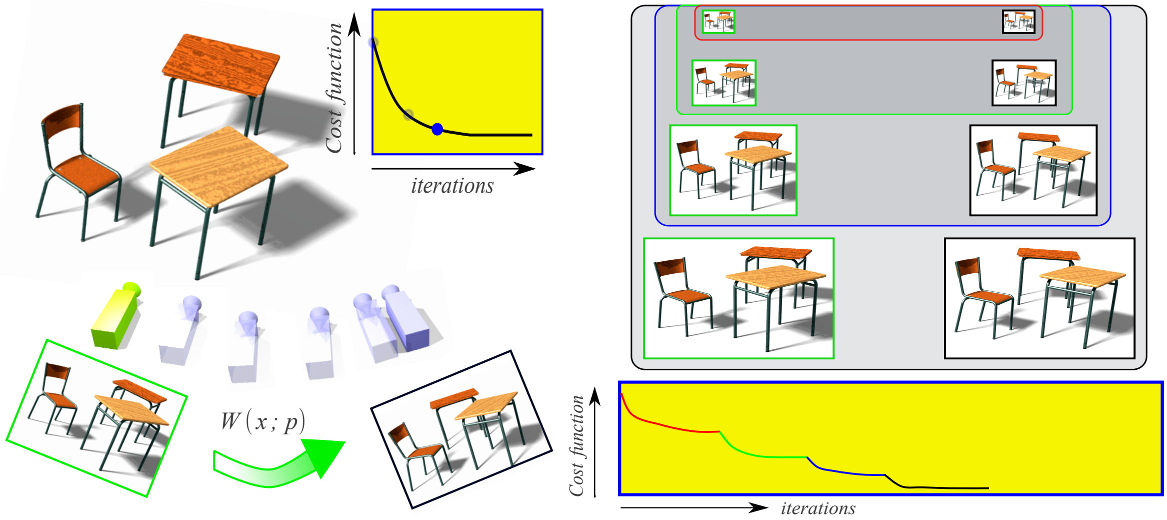


Figure 1.5: A toy example of registration of 3D model to a 2D image. Image at the previous pose (in green) is anchored to the 3D model and the depth-map corresponding to that pose is used to align a new incoming image to obtain the new pose. The parameters required to warp the images are expressed by $W(x; p)$. As iterations progress, the value of the cost function decreases leading to convergence towards a locally optimal estimate. The iterative process, in fact, is carried out in a standard coarse-to-fine pipeline to provide wider basin of convergence.

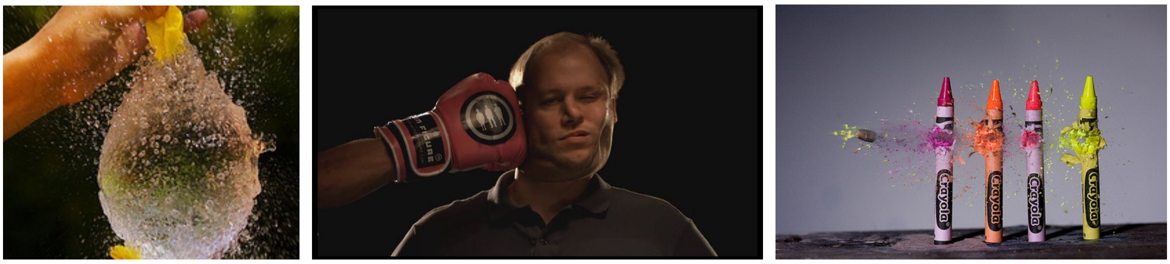
1.5.3 Signed Distance Based Tracking

Signed distance is a boundary-aware representation of curve/surface. The curve/surface serves as reference against which any other point in the domain is related by its minimum distance from the curve. The sign of the distance encodes whether the point lies inside or outside the boundary of the surface. [Danielsson, 1980] introduced distance mapping and briefly discussed about signed distance. Signed distance have emerged a preeminent framework especially in computer graphics, to represent any curve or 3D surface that cannot be defined analytically. Among others [Curless, 1997] and [Curless and Levoy, 1996] showed the remarkable properties of a signed distance transform to reconstruct 3D surfaces from range images and lately [Newcombe et al., 2011a] demonstrated their impressive real-time 3D reconstruction sytem, KinectFusion, using the Microsoft Kinect depth sensor. However, they only show the utility of signed distance for merging 3D models. Tracking based on sign distance transform has been very less visited in comparison to 3D reconstruction. But, it is not new and has also been used before by [Fitzgibbon, 2001] for aligning two 3D models and they show how wider the basin of convergence is for such tracking. Recently and in particular, [Canelhas, 2012] and [Ren and Reid, 2012] also used the signed distance transform to guide the 6-DoF pose estimation by aligning the incoming depth map against the 3D model of the surface viewed from the pose obtained from the previous step.

1.6 Motivation For High Speed Tracking: The Role of High Frame-Rate

Almost all data-associations techniques that establish correspondences within the loop of search for transformation, at some stage, employ linearisation to break away from the non-linearities that prevent a tractable and efficient search for transformation. *How good this linearised approximation is a matter of how coherent and similar the consecutive observations are or how good the prediction of the model is.* A prediction is obtained by rolling forward the time interval between the previous pose estimate and the current incoming data observation. That a prediction would naturally improve tracking is at the heart of *model-based tracking* paradigm. It is therefore expected that as the time interval between the prediction and current data observation decreases, the prediction increasingly matches with observation, leading to better, accurate and robust results. Figure 1.6 shows images taken at high frame-rates and it is hard to not to impressed by the resolution of motion they can capture that would aid any prediction based tracking system.

In real-time tracking frame-rate sets an upper bound on the maximum time budget required to finish *per-frame* tracking and as a result has a direct effect on tracking performance. Imagine observing the motion of a ball thrown at 150 Km/h with a 30Hz camera. The ball moves about 1.4m when sampled at that frame-rate, which is clearly not trackable as a result of fast motion and blur. Higher sampling rate (generally 1ms is treated as standard time interval when talking of high frame-rates) can definitely ameliorate this problem, for instance observing the same motion at 1000Hz (1ms) can reduce the motion down to 4.1cm *per-frame*. The earliest works by the famous photograher Eardward Muybridge (see Figure 1.7) that



(a) Balloon bursting (b) Boxing punch (c) Bullet through crayons

Figure 1.6: Examples from the internet of slow motion image capture that happens at high frame-rate.

captured images of a moving horse also demonstrated the temporal coherence associated with the images. Thus, the high degree of coherence and ease associated with tracking makes higher frame-rate a very appealing choice. Industrial systems like Hawkeye [3] have also been employed with a large degree of success in sports like cricket and tennis where prediction of the trajectory of the ball is made using high speed cameras.

Among others, [Kagami, 2010] pointed out that high frame-rate can make image processing simpler. In some cases, the benefits of high frame-rate can substantially outweigh the reduction in computational budget. An example they give is that of a 2D tracking of an object in the image, where the 2D translation parameters are searched for exhaustively in a given window[4]. If the frame-rate is increased by a factor of k, the allotted computational time budget get reduced by this factor. The region in which the search is carried out also decreases by a factor[5] of k. However, extrapolating this to searching in more than two dimensions means that benefits of high frame-rate would clearly be more[6]. Also, in EKF-style trackers as the search regions shrink with the increase in the frame-rate, the probability of false-positives that can occur in the search region reduces too, leading the distribution of observation to be uni-modal and Gaussian. Trackers that work by minimising a least-squares cost function need a good initialisation to converge to a local optimum. This initialisation becomes increasingly better if the frame-rate dial of the algorithm is pushed up. This is substantially beneficial again as it prevents the optimisation from getting stuck into a local minima. In fact, this is a direct consequence of the distribution turning Gaussian in Bayesian

[3] http://en.wikipedia.org/wiki/Hawk-Eye. Hawkeye runs at 106Hz as reported in http://www.espncricinfo.com/ci/content/story/530564.html.

[4] A notable difference we have is that search area is a linear function of time-interval between the frames unlike theirs which is quadratic.

[5] This is assuming that the motion model is random walk, *i.e.* the uncertainty increases as a function of Δt, the time interval between frames.

[6] For any n-dimensional sphere the volume is a function of $(\Delta t)^{\frac{n}{2}}$ going by the random walk model.

language.

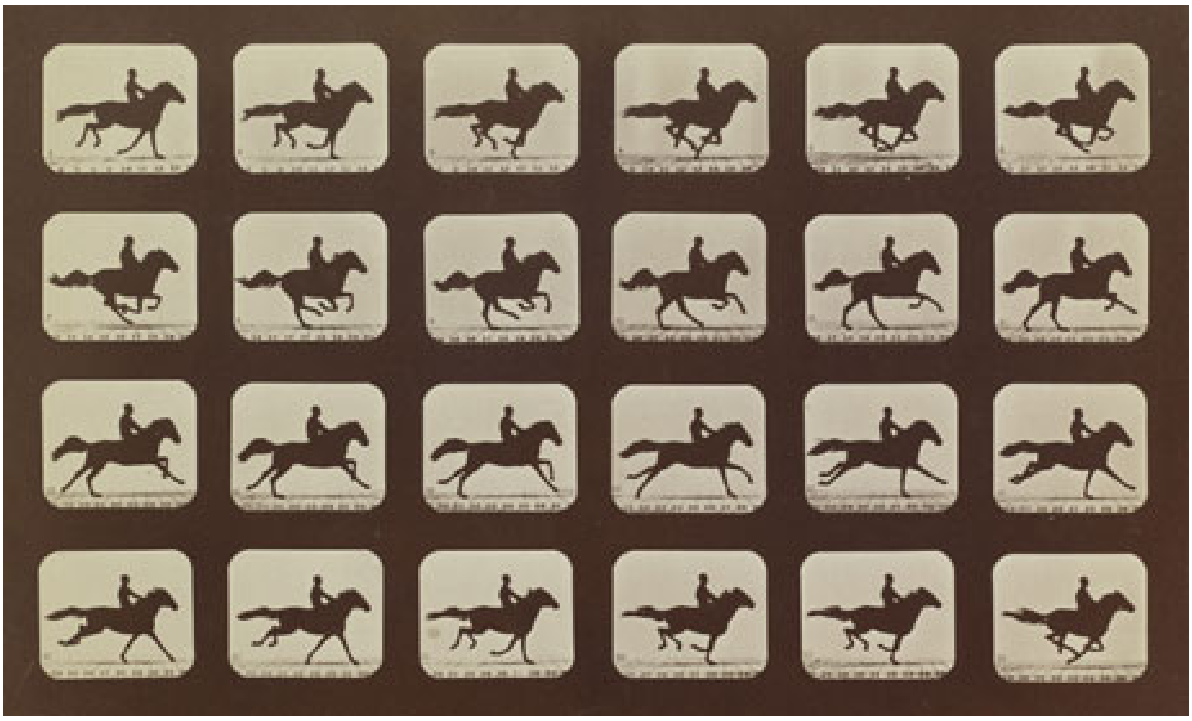
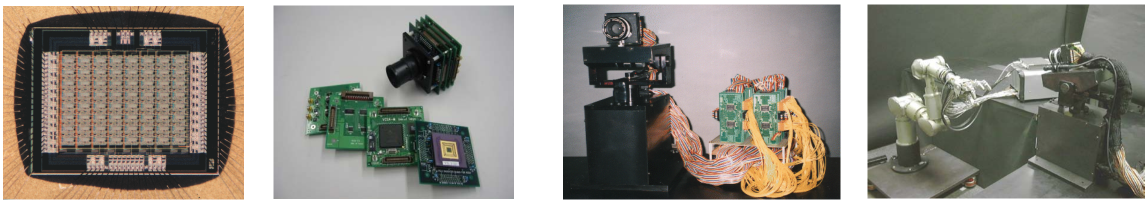


Figure 1.7: Horses running, Phryne L. Plate 40, 1879, from The Attitudes of Animals in Motion, 1881, by Eadweard Muybridge. Image courtesy of the National Gallery of Art, Washington. The image has been obtained from http://www.guardian.co.uk/artanddesign/2010/aug/29/eadweard-muybridge-tate-review

Moreover, the ability to track rapid shaky motion that high frame-rate provides has attracted many hardware enthusiasts. The bounded time interval between two consecutive frames puts a severe limit on the data transfer that can occur between the camera and the host machine doing the image processing. The first high-speed vision system introduced by [Ishikawa et al., 1992] was designed keeping in mind maximum time delay that can occur between the host and the camera must be less than a millisecond. As a result the data transfer between the camera and the host machine was replaced altogether by having a dedicated *vision chip* integrated on the camera device for high speed image processing. Such ideas have brought about a change in the way high-speed vision system building is viewed — the bottleneck of the data transfer must be minimised first to reduce or diminish the communication cost between the two devices.

Custom chips have been built that aim to perform the standard image processing tricks needed for tracking directly in the hardware to reduce the computational bottleneck. Fig-



(a) Different architectures of vision chips.

(b) Active vision systems.

Figure 1.8: Different massively parallel SIMD architectures of the vision chips that have been used to speed up the simple image processing operations to track fast moving objects. Active vision systems that are designed using these vision chips can track the motion of a bouncing ball or catch a ball thrown towards a multi-fingered hand.

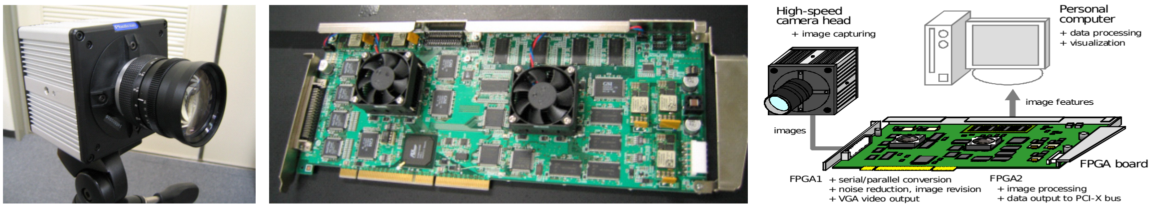


Figure 1.9: New generation H^3 vision platform that does majority of image processing on the FPGA directly linked to the camera.

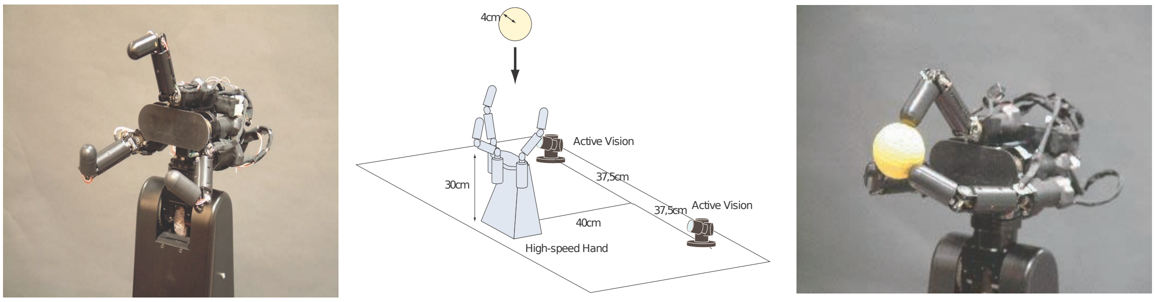


Figure 1.10: Multi-fingered Hand system.

ure 1.8 and 1.9 show images of different architectures of chips and systems (also Figure 1.10) that have used these custom made chips. The architecture of these chips is often a massively parallel SIMD processing array with a 2D grid like structure. The self-window method [Ishii et al., 1996] is able to perform object tracking in the 1ms tight budget by using very simple image processing operations. The previously obtained location of the target

Frame-rate	**Applications**
1 frame/hour	Extreme time-lapse photography.
1 frame/minute	Time-lapse photography and stop-motion animation.
18 frames/second	Early motion picture films.
24 frames/second	Worldwide standard for movie theater film projectors.
48 frames/second	Slow-motion photography.
300+ frames/second	High-speed cameras for very slow-motion photography.
2500+ frames/second	High-speed cameras for pyrotechnic photography.

Table 1.1: Range of applications for different frame-rates

object is dialated and AND'ed with the new sensory data to obtain the location of the object in new frame. [Nakabo et al., 1996] show a 1ms target tracking system that is able to track a bouncing ball without any internal model or prediction. This has also been used in visual impedance control system in [Nakabo and Ishikawa, 1998]. A high frame-rate tracking workshop was also organised at ICCV 1999 [7] where researchers have attempted to draw the attention of the robotics and vision community to high frame-rate camera trackers. [Ishii et al., 2009] introduced a high speed vision platform, H^3 that implements various simple image processing algorithms. Also, [Ishii et al., 2010] showed an optical flow estimation system that runs at 1000Hz on a customised FPGA. All the necessary image processing *e.g.* computing gradients, and adding them, were directly performed on the hardware.

The Region of Interest (ROI) feature [Monacos et al., 2001] that the new cameras provide can be used for a very high bandwidth data transfer to the host over a digital data bus like IEEE 1394, PCI Express, Gigabit Ethernet. However, there is trade off between the data transfer and the image resolution that can be used [Kagami, 2010].

1.6.1 Applications

A series of high speed vision systems have come out of Masatoshi Ishikawa's lab. They range from tracking a bouncing ball [Ishikawa et al., 1992],[Nakabo et al., 1996], high speed grasping of an object [Namiki et al., 1999], batting [Senoo et al., 2006], dribbling [Shiokata et al., 2005], throwing [Senoo et al., 2008] and folding a cloth [Yamakawa et al., 2011]. Table 1.1 also shows different applications a given frame-rate sequence can be useful in.

[7] http://vast.uccs.edu/~tboult/frame/

1.6.2 Focus of this Thesis

Keeping in view of the benefits of high frame-rate images for camera tracking, this thesis focusses on to systematically understand under what conditions and kind of motions would it make sense to use high frame-rate tracking that uses prediction coming from a model.

1.7 Contributions

This thesis has led to the following publications:

- Ankur Handa, Margarita Chli, Hauke Strasdat, Andrew J. Davison (2010). Scalable Active Matching. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

- Ankur Handa, Richard A. Newcombe, Adrien Angeli, Andrew J. Davison(2011). Applications of Lengendre-Fenchel Transformation to Computer Vision Problems. *DTR-2011, Deparmental Technical Report, Imperial College London*.

- Ankur Handa, Richard A. Newcombe, Adrien Angeli, Andrew J. Davison (2012). Real-Time Camera Tracking: When Is High Frame-Rate Best? *In Proceedings of the IEEE European Conference on Computer Vision (ECCV).*

1.8 Thesis Structure

The thesis is laid out with the following structure:

Chapter 2 gives a short tour of mathematical theory and provides some derivations and code snippets that we use in the subsequent parts of the thesis.

Chapter 3 describes the prior driven feature matching approach to image matching and novel contributions we make to provide scalability to the matching algorithm with an aim towards doing dense matching.

Chapter 4 debates various contrasting differences between the feature based and dense direct parametric tracking. It argues that for full understanding of frame-rate dense methods are more suitable and in the subsequent chapters that forms our tracking method we analyse.

Chapter 5 provides a full description of synthetic framework that renders photo-realistic images required to systematically alter various different parameters *e.g.* frame-rate and image resolution for the analysis.

Chapter 6 throws insights into the conclusions we obtain from synthetic images.

Chapter 7 validates the conclusions with real data coming from a pure 1D rotating servo with ground truth obtained from an accurate high frequency and high bandwidth gyro.

Chapter 8 concludes the thesis.

CHAPTER 2

Mathematical Preliminaries

Contents

2.1 Rigid Transformations

Transformations allow us to define the location and movements of objects, points or reference frames in a geometric world. Any representation of such transformations then enables us to write them down in a form that can be easily manipulated mathematically. It is also of interest to understand the space a transformation belongs to. A natural space for simple translations that an object undergoes in three dimensional scenes is *Euclidean space* however, rotations or a combination of rotations and translations belong to a rather different manifold which is not *Euclidean*. Therefore, the general operators that compose vectors for subtraction and addition in *Euclidean space* do not hold true in these manifolds as such. It becomes essential to understand what operators can be used to compose the transformations. We discuss first the matrix representations for the transformations we have used in the thesis and then detail the minimal parametrisation and also throw light on how to compose the transformations later.

Rigid body transformations in 3D are represented by 3×3 matrices for rotations, R and 4×4 for joint rotations and translations, (R,t).

$$\mathsf{T} = \left(\begin{array}{c|c} \mathsf{R} & \mathsf{t} \\ \hline 0 & 1 \end{array}\right), \quad \mathsf{T}_{ab} = \left(\begin{array}{c|c} \mathsf{R}_{ab} & \mathsf{t}_{ab} \\ \hline 0 & 1 \end{array}\right) . \tag{2.1}$$

The transformations are often written with subscripts, for instance T_{ab}. It is read as $\mathsf{T}_{a \curvearrowleft b}$, the transformation that transfers points represented in coordinate frame of b to coordinate frame of a. It is important to remember that whenever the transformations involve both rotation and translation, the points are expressed in homogeneous coordinates — an extra 1 is appended to the point and therefore, the points are transferred as follows:

$$\begin{pmatrix} \mathsf{p}_a \\ 1 \end{pmatrix} = \left(\begin{array}{c|c} \mathsf{R}_{ab} & \mathsf{t}_{ab} \\ \hline 0 & 1 \end{array}\right) \begin{pmatrix} \mathsf{p}_b \\ 1 \end{pmatrix} . \tag{2.2}$$

While transferring points, it is important to ensure that the subscripts match *i.e.* T_{ab} must only go with p_b[1].

A minimal vector parametrisation is often sought to express the rotations and translations instead of using full matrices with 9 and 12 variables respectively. This is essentially useful when performing optimisations on the transformations where if full matrices are used extra care has to be taken that the properties and structures of matrices remain preserved during the optimisation. For instance, if the optimisation is carried out on full rotation matrices, one must ensure that the updates must yield a rotation matrix with orthonormality constraint, when consumed. On the other hand the minimal parametrisations allows to smoothly move on the manifold of transformations without requiring any extra care on preserving the properties of matrices — the mapping of this parametrisation to the transformation ensures that automatically.

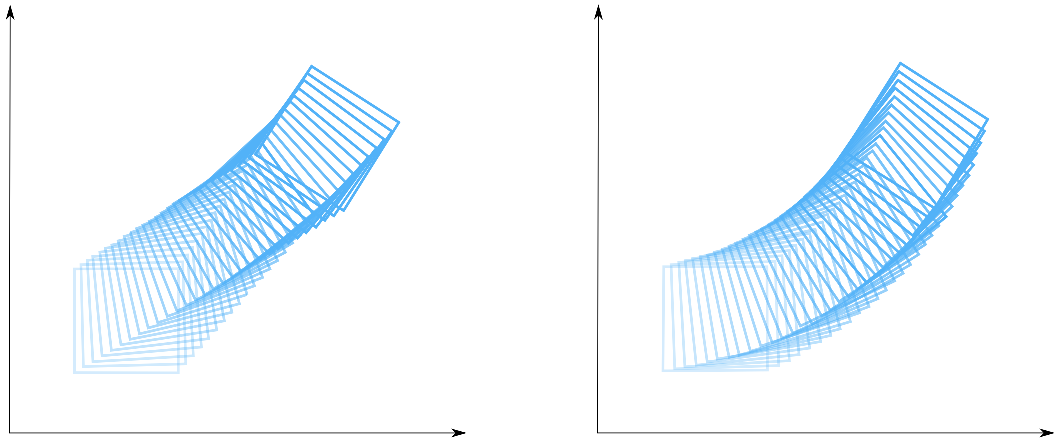
The parametrisation then provides a principled way to represent a transformation such that if a particular transformation is inverted, composed, interpolated or differentiated, it varies smoothly and still remains within that same space of transformations. For instance, Rotation composed with another Rotation is always Rotation. Such operations form an integral part of that space and the operations and the operators used in the space collectively define a Group which preserves the general properties of the transformations e.g. a transformation composed with an operator to another transformation from the same space must yield a new transformation that is a part of the space. In the following we explain the most popular parametrisation in Lie Groups to express the transformations.

[1] The red colour is only to highlight the subscripts that match.

2.2 Lie Group Framework

Formally, G is called Lie Group if $G \times G \rightarrow G$ and the inverse $G \rightarrow G$ are smooth manifolds. It obviously satisfies the general properties of group but at the same time the group operations are differentiable. The smooth manifold and Group properties are shown in Figure 2.1 where a comparison of linear interpolation on a transformation is made with the interpolation using Lie Group [2].

One of the key ideas of Lie Group is to express a global object of the group with a local version which is "infinitesimal group" and is popularly known as Lie Algebra. This is something we quite often employ when linearising transformations around a given point to obtain a first order Taylor series expansion used in non-linear least squares optimisation. Below we turn our attention to Lie Algebra.



(a) Linear Interpolation: $T(t) = I + t(\hat{T} - I)$

(b) Lie Group Interpolation: $T(t) = \exp(t\hat{T})$

Figure 2.1: Comparison between the linear interpolation and interpolation using Lie Group. The transformations interpolated using the Lie Group properties vary smoothly and the new transformation obtained is rigid and lies within the Lie Group. On the other hand, linear interpolations can provide transformations that are not rigid.

2.2.1 Lie Algebra

In simple terms, a Lie Algebra provides a minimal parametrisation and that this minimal parametrisation allows one to move on the smooth manifold of the transformation that

[2] http://www.robots.ox.ac.uk/~cmei/talks/reading_group_lie.pdf also show how to interpolate transformations in Lie Group.

belongs to the Lie Group.

A Lie Group G with a element in $\mathbb{R}^{n\times n}$ comes associated with a Lie Algebra $\mathfrak{g}$ of k degrees of freedom or parameters. The Lie Algebra is associated with the tangent space of the Lie group around its Identity and as a result helps represent infinitesimal changes around a given element of the group. The mapping from Lie Algebra to Lie Group is via the exponential-map, *i.e.*

$$\exp : \mathfrak{g} \longrightarrow G \tag{2.3}$$

The exponential-map has some interesting properties that come to use when dealing with the transformations. They are listed below $\forall\ \hat{\mathrm{x}} \in \mathfrak{g}$

- Inversion *i.e.* $(\exp(\hat{\mathrm{x}}))^{-1} = \exp(-\hat{\mathrm{x}})$
- Exponentiation property *i.e.* $\exp(s\hat{\mathrm{x}})\exp(t\hat{\mathrm{x}}) = \exp((s+t)\hat{\mathrm{x}}), \forall s,t \in R$
- Derivative property *i.e.* $\frac{\partial \exp(t\hat{\mathrm{x}})}{\partial t} = \hat{\mathrm{x}}\exp(t\hat{\mathrm{x}}) = \exp(t\hat{\mathrm{x}})\hat{\mathrm{x}}$
- Chaining in Abelian Groups *i.e.* if $\hat{\mathrm{x}}\hat{\mathrm{y}} = \hat{\mathrm{y}}\hat{\mathrm{x}} \Rightarrow \exp(\hat{\mathrm{x}})\exp(\hat{\mathrm{y}}) = \exp(\hat{\mathrm{x}}+\hat{\mathrm{y}}), \forall \hat{\mathrm{y}} \in \mathfrak{g}$
- Adjoint property in non-Abelian Groups *i.e.* $\exp(\mathrm{A}\hat{\mathrm{x}}\mathrm{A}^{-1}) = \mathrm{A}\exp(\hat{\mathrm{x}})\mathrm{A}^{-1}, \forall \mathrm{A} \in \mathbb{R}^{n\times n}$

Two particular groups that are of special interest to the robotics and camera tracking community are the group of rotations, $\mathbf{SO(3)}$ and the group of rotations and translations, $\mathbf{SE(3)}$. In the following we briefly introduce these groups and also show how the exponential figures in this mapping.

2.2.2 Group of Rotations: SO(3)

Let us consider a point q on a rigid body undergoing pure rotation. The axis of rotation is defined by $\omega = (\omega_x, \omega_y, \omega_z)$ and we also assume that it is moving with a unit angular velocity [3]. Then the instantaneous velocity at any time t is:

$$\dot{\mathrm{q}}(t) \;=\; 1\omega \times \mathrm{q}(t) \quad (\because v = \omega \times r \quad \text{for pure rotational motion})\ . \tag{2.4}$$

The cross product in the expression can be replaced by an equivalent skew-symmetric matrix multiplication operation, *i.e.*

$$\dot{\mathrm{q}}(t) \;=\; \hat{\omega}\mathrm{q}(t)\ , \tag{2.5}$$

[3] The direction of angular velocity is given by the axis of rotation.

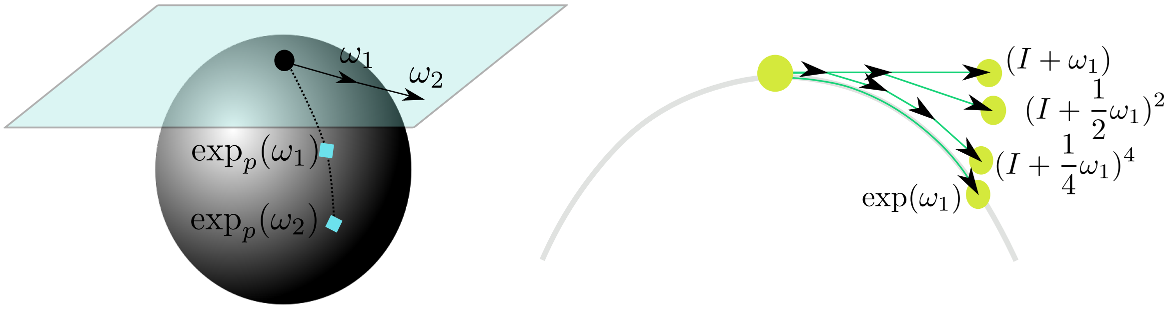


Figure 2.2: **SO**(3) and the associated exponential map. The first order approximation is a tangent plane on the surface of the sphere at a given point. Moving from one point to the other on the sphere is via the exponential map. This is a higher dimensional equivalent of moving on a circle which is via exponential-map due to polar-coordinate representation and De Moivre's theorem. The image on the right is reproduced from Tom Drummond's notes on Lie Algebra: https://dl.dropboxusercontent.com/u/23948930/Papers/3DGeometry.pdf. The approximations of exponential-map are given by $\exp(x) = \lim_{n\to\infty}(1 + \frac{1}{n}x)^n$

where the equivalent skew-symmetric matrix is

$$\hat{\omega} = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix} . \tag{2.6}$$

This leads to a time varying differential equation [Murray et al., 1994, p. 37–62] [4] which can be integrated to give

$$\dot{\mathsf{q}}(t) = \frac{\partial}{\partial t}\mathsf{q}(t) = \hat{\omega}\mathsf{q}(t) \tag{2.7}$$

$$\implies \mathsf{q}(t) = e^{\hat{\omega}t}\mathsf{q}(0) , \tag{2.8}$$

where $e^{\hat{\omega}t}$ is the matrix exponential. It means that if we rotate an object about an axis ω at unit velocity for θ units of time, then the net rotation is given by

$$R(\hat{\omega},\theta) = e^{\hat{\omega}\theta} = I + \hat{\omega}\theta + \frac{1}{2!}(\hat{\omega}\theta)^2 + ...\infty = I + \hat{\omega}\sin\theta + \hat{\omega}^2(1-\cos\theta) . \tag{2.9}$$

The closed form expression allows us to compute the exponential on computer [5]. The **SO**(3) Lie Group has an associated Lie Algebra $\mathfrak{so}(3)$ as shown in Figure 2.2 that defines an infinitesimal rotation around a given point and allows to move smoothly from one rotation to the other on a sphere via the exponential-map.

[4] The notations are largely adapted from this book.

[5] For a more general case when $||\omega|| \neq 1$, it is:

$$R(\hat{\omega},\theta) = I + \frac{\hat{\omega}}{||\omega||}\sin(||\omega||\theta) + \frac{\hat{\omega}^2}{||\omega||^2}(1-\cos(||\omega||\theta)) .$$

Each element in this group can be parametrised by three degrees of freedom corresponding to the axis about which the object is undergoing the rotation.

$$\exp : \mathfrak{so}(3) \longrightarrow \mathbf{SO}(3) \tag{2.10}$$

$$\mathsf{R} \in \mathbf{SO}(3), \ \ \mathsf{R}^T\mathsf{R} = \mathbf{I} \tag{2.11}$$

The $\mathfrak{so}(3)$ space parametrisation $\omega \in \mathbb{R}^3$ defines a corresponding rotation matrix in $\mathbf{SO}(3)$ $\in \mathbb{R}^{3\times3}$. A first order approximation of a rotation matrix can be derived from the first order approximation of the exponential-map as shown in Figure 2.2. The $\hat{\omega}$ can be described by the linear combination of three different matrices that define the generators of the rotation matrix.

$$\hat{\omega} = \omega_x \mathrm{G}_1 + \omega_y \mathrm{G}_2 + \omega_z \mathrm{G}_3 \in \mathfrak{so}(3) \tag{2.12}$$

These Generators are the basis rotation matrices that can represent a small rotation about each of the three axes and are:

$$\mathsf{G}_1 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}, \ \mathsf{G}_2 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix}, \ \mathsf{G}_3 = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} . \tag{2.13}$$

It is worthwhile to connect this to the Euler-angle representation matrix $\mathrm{R}_{\mathrm{euler}}$ that is described by yaw-pitch-roll angles as

$$\mathrm{R}_{\mathrm{euler}} = \begin{pmatrix} \cos\theta\cos\psi & -\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{pmatrix} ,$$

With a small motion assumption around each of the axes, this can be rewritten as

$$\mathrm{R}_{\mathrm{euler}} = \begin{pmatrix} 1 & -\psi + \phi\theta & \phi\psi + \theta \\ \psi & 1 + \phi\theta\psi & -\phi + \theta\psi \\ -\theta & \phi & 1 \end{pmatrix} , \tag{2.14}$$

Therefore, the first order approximation yields

$$\mathrm{R}_{\mathrm{euler}} = \begin{pmatrix} 1 & -\psi & \theta \\ \psi & 1 & -\phi \\ -\theta & \phi & 1 \end{pmatrix} = \mathrm{I} + \phi\mathsf{G}_1 + \theta\mathsf{G}_2 + \psi\mathsf{G}_3 . \tag{2.15}$$

We see a straightforward connection between different representations when a small angle assumption is made.

2.2.3 Group of Rotations and translations: SE(3)

Let us now consider a point p undergoing a general rigid body motion in three-dimensional space. The velocity can be then described by the following:

$$\dot{\mathsf{p}}(t) \quad = \quad \overbrace{\omega \times \mathsf{p}(t)}^{\text{rot. vel}} + \underbrace{\mathsf{v}(t)}_{\text{trans. vel}} \tag{2.16}$$

Here $p(t)$ is a vector from the origin of the body frame to point p and v is the translational velocity. This is standard vector addition of rotational velocity component and translational velocity component. Replacing the cross product with skew-symmetric matrix multiplication, the expression becomes

$$\dot{\mathsf{p}}(t) \quad = \quad \hat{\omega}\mathsf{p}(t) + \mathsf{v}(t) \tag{2.17}$$

Let us also define

$$\hat{\xi} \triangleq \begin{pmatrix} \hat{\omega} & \mathsf{v} \\ 0 & 0 \end{pmatrix}, \tag{2.18}$$

Equation 2.17 can be further reduced to a first order differential equation of the form

$$\begin{pmatrix} \dot{\mathsf{p}} \\ 0 \end{pmatrix} = \begin{pmatrix} \hat{\omega} & \mathsf{v} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \mathsf{p} \\ 1 \end{pmatrix}, \tag{2.19}$$

The solution of this ordinary differential equation is of the form

$$\begin{pmatrix} \dot{\mathsf{p}} \\ 0 \end{pmatrix} = \frac{d}{dt}\begin{pmatrix} \mathsf{p} \\ 1 \end{pmatrix} \quad = \quad \begin{pmatrix} \hat{\omega} & \mathsf{v} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \mathsf{p} \\ 1 \end{pmatrix}; \tag{2.20}$$

$$\implies \mathsf{p}(t) \quad = \quad e^{\hat{\xi}t}\mathsf{p}(0) \tag{2.21}$$

The corresponding exponential-map is defined from $\exp : \mathfrak{se}(3) \longrightarrow \mathbf{SE}(3)$, a mapping from $\mathbb{R}^6$ to an element in $\mathbb{R}^{4\times4}$. The closed-form expression for the exponential is

$$e^{\hat{\xi}\theta} = \begin{bmatrix} e^{\hat{\omega}\theta} & (I - e^{\hat{\omega}\theta})(\omega \times v) + \omega\omega^T v\theta \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{4\times4} \tag{2.22}$$

The corresponding set of generators is:

$$\mathsf{G}_1 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \ \mathsf{G}_2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \ \mathsf{G}_3 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.23}$$

$$\mathsf{G}_4 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \ \mathsf{G}_5 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \ \mathsf{G}_6 = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.24}$$

2.2.4 Adjoint Representation

It is sometimes required to transform the coordinate frames of a Lie Algebra. The Adjoint representation allows us to transform the associated Lie Algebra elements from one coordinate frame to the other. In the following we derive a relationship between the Lie Algebras and show where this could be useful.

Let us assume that a body in an object frame of reference or *body frame of reference*, *b*, is rotating about a *spatial frame of reference*, *a*. Therefore, any point represented in the body reference frame can be mapped to the spatial reference frame via the rigid body transformation involving rotation, *i.e.*:

$$\mathsf{q}_a(t) \quad = \quad \mathsf{R}_{ab}(t)\mathsf{q}_b\ . \tag{2.25}$$

Differentiating with respect to t to get the velocity gives us:

$$\mathsf{v}_a(t) \quad = \quad \dot{\mathsf{R}}_{ab}(t)\mathsf{q}_b\ . \tag{2.26}$$

However, this representation is cumbersome as it requires the velocity to be represented with the nine variables of $\dot{\mathsf{R}}_{ba}$. Therefore, a compact representation is sought by rewriting the velocity as

$$\mathsf{v}_a(t) \quad = \quad \dot{\mathsf{R}}_{ab}(t)\mathsf{R}_{ab}(t)^{-1}\mathsf{R}_{ab}(t)\mathsf{q}_b\ . \tag{2.27}$$

Let us call the instantaneous *spatial angular velocity* $\hat{\omega}^s_{ab} = \dot{\mathsf{R}}_{ab}(t)\mathsf{R}_{ab}(t)^{-1}$. This is the angular velocity of the object as seen from the spatial frame of reference, *a*. Similarly the *body angular velocity* is defined as $\hat{\omega}^b_{ab} = \mathsf{R}_{ab}(t)^{-1}\dot{\mathsf{R}}_{ab}(t)$. A simple mathematical manipulation yields the following two relations between the velocities

$$\hat{\omega}^b_{ab} = R^{-1}_{ab}\hat{\omega}^s_{ab}R_{ab} \quad \text{or} \quad \omega^b_{ab} = R^{-1}_{ab}\omega^s_{ab} \ . \tag{2.28}$$

Let us now consider a full rigid body motion in **SE**(3) where the transformation is defined by g_{ab} as

$$\begin{bmatrix} R_{ab} & t_{ab} \\ 0 & 1 \end{bmatrix} . \tag{2.29}$$

Let us also define the instantaneous spatial velocity $\hat{V}_{ab} \in \mathbf{SE}(3)$ similar to spatial angular velocity as

$$\hat{V}^s_{ab} = \dot{g}_{ab}g^{-1}_{ab} = \begin{bmatrix} \dot{R}_{ab} & \dot{t}_{ab} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} R^T_{ab} & -R^T_{ab}t_{ab} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \dot{R}_{ab}R^T_{ab} & -\dot{R}_{ab}R^T_{ab}t_{ab} + \dot{t}_{ab} \\ 0 & 0 \end{bmatrix} . \tag{2.30}$$

Comparing that to the standard form of $\hat{\xi}$ we get

$$v^s_{ab} = -\dot{R}_{ab}R^T_{ab}t_{ab} + \dot{t}_{ab} \ . \tag{2.31}$$

If the instantaneous velocity is represented in the body frame of reference it is

$$\hat{V}^b_{ab} = g^{-1}_{ab}\dot{g}_{ab} = \begin{bmatrix} R^T_{ab}\dot{R}_{ab} & R^T_{ab}\dot{t}_{ab} \\ 0 & 0 \end{bmatrix} . \tag{2.32}$$

This yields the translation velocity as

$$v^b_{ab} = R^T_{ab}\dot{t}_{ab} \Longrightarrow R_{ab}v^b_{ab} = \dot{t}_{ab} \ . \tag{2.33}$$

The velocity v^s_{ab} can be further rewritten as

$$\begin{aligned} v^s_{ab} &= -\omega^s_{ab} \times t_{ab} + \dot{t}_{ab} && (2.34) \\ v^s_{ab} &= t_{ab} \times (R_{ab}\omega^b_{ab}) + \dot{t}_{ab} && (2.35) \\ v^s_{ab} &= \hat{t}_{ab}R_{ab}\omega^b_{ab} + R_{ab}v^b_{ab} \ . && (2.36) \end{aligned}$$

Therefore we can summarise this by

$$\begin{aligned} \hat{V}^s_{ab} &= \dot{g}_{ab}\hat{V}^b_{ab}\dot{g}^{-1}_{ab} && (2.37) \\ \begin{pmatrix} v^s_{ab} \\ \omega^s_{ab} \end{pmatrix} &= \begin{bmatrix} R_{ab} & \hat{t}_{ab}R_{ab} \\ 0 & R_{ab} \end{bmatrix} \begin{pmatrix} v^b_{ab} \\ \omega^b_{ab} \end{pmatrix} . && (2.38) \end{aligned}$$

This is transforming the $\mathfrak{se}(3)$ representation in the body reference frame into the spatial reference frame. This transformation matrix is called Adjoint:

$$\begin{aligned} \mathrm{Ad}_a &= \begin{bmatrix} R & \hat{t}R \\ 0 & R \end{bmatrix} . && (2.39) \\ \textit{i.e.}\ \mathrm{Ad}_a &: \mathfrak{g} \longrightarrow \mathfrak{g} && (2.40) \end{aligned}$$

Where do we need it?

As shown before, transforming coordinate frames in Lie Algebra is via the Adjoint representation. If this transformation is used within least squares minimisation, it may involve taking the derivative of the parameters in spatial frame with respect to the body frame while applying chain rule as done in [Comport et al., 2007], during hand-eye calibration process[6] (*e.g.* gyro camera calibration) and it is

$$\frac{\partial \begin{pmatrix} \mathsf{v}^s_{ab} \\ \omega^s_{ab} \end{pmatrix}}{\partial \begin{pmatrix} \mathsf{v}^b_{ab} \\ \omega^b_{ab} \end{pmatrix}} = \begin{bmatrix} \mathsf{R} & \hat{\mathsf{t}}\mathsf{R} \\ 0 & \mathsf{R} \end{bmatrix} . \tag{2.41}$$

2.3 Camera Calibration

Camera calibration is an essential step in (a) understanding the imaging process and (b) if we wish to use a camera asa a measuring device. This includes mapping of the geometric structure to obtain the pixel location and scene radiance to the pixel value observed. To associate the location of a pixel to the 3D point in the scene or the ray, a geometric calibration procedure is required. On the other hand, the colour value of the pixel is obtained from the scene radiance and surface properties of the object. Therefore, a similar calibration procedure, radiometric calibration, is required that allows us to map the scene radiance to the pixel colour value. We show in a block diagram in Figure 2.3 how a ray from the scene gets mapped to a pixel location and colour value. We detail both calibrations in the following, beginning with geometric calibration.

2.3.1 Geometric Calibration

How does a camera project a 3D point to the image plane? Camera geometric calibration aims to find the parameters that relate the 3D point (x, y, z) to its projection on the image plane. The camera projection is modelled by the well known pin-hole model that involves the projection via a linear operator followed by normalisation by the z-coordinate to obtain the 2D image

[6] http://campar.in.tum.de/Chair/HandEyeCalibration

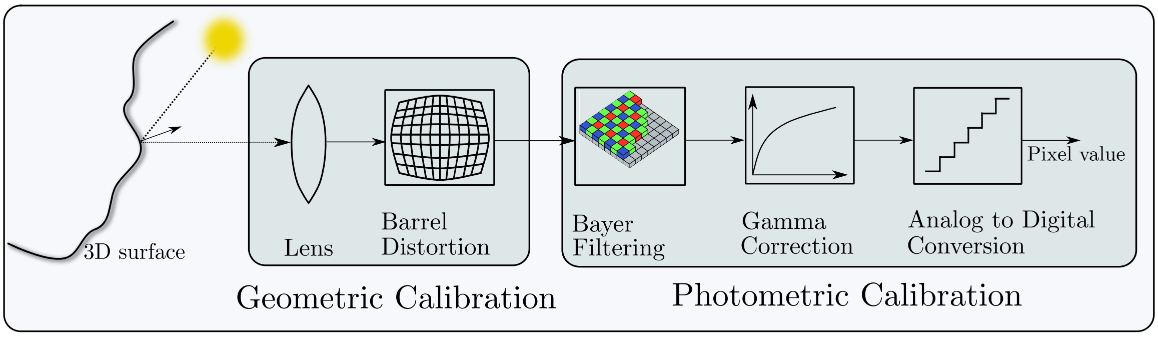


Figure 2.3: The camera image acquisition process explained. The incident ray when hits the surface gets reflected according to the surface normal. It then undergoes lens distortion before hitting the sensor plane (it is termed as *Irradiance* when it hits the sensor plane) where it is filtered according to the colour and generates a corresponding voltage value. The voltage then is digitised to a pixel value. This voltage also undergoes different distortions before it is digitised. The distortions are often removed by appropriate calibrations. The distortions produced by a lens require geometric calibration where the focal length, camera center and the distortion parameters are recovered while the distortions added to the voltage value require a photometric calibration where an appropriate Camera Reponse Function is obtained that gives the mapping of irradiance to pixel value.

projection.

$$\pi \mathsf{K}\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \pi\left(\begin{pmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \\ z \end{pmatrix}\right). \tag{2.42}$$

Matrix K is standard camera calibration matrix with parameters f_u and f_v denoting the focal lengths (in pixels) of the camera along the x and y axis while u_0 and v_0 denote the corresponding locations of the camera's center. π is the standard projection operator. Moreover, the camera lens introduces various distortions to the projection. Therefore, the projection is not exactly at the expected location; instead it is at a distorted location. We model our camera distortion by radial barrel distortion [Devernay and Faugeras, 2001, Klein and Murray, 2007] and therefore the camera projection step can be rewritten as follows:

$$\pi \mathsf{K}\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \frac{\mathrm{r}'}{\mathrm{r}}\pi\left(\begin{pmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \\ z \end{pmatrix}\right), \tag{2.43}$$

$$\mathrm{r} = \sqrt{\frac{x^2+y^2}{z^2}}, \tag{2.44}$$

$$\mathrm{r}' = \frac{1}{\omega}\arctan\left(2\mathrm{r}\tan\frac{\omega}{2}\right). \tag{2.45}$$

Camera geometric calibration then involves estimating these five parameters $(u_0, v_0, f_u, f_v, \omega)$ from a set of images. We follow the procedure as shown in [Klein and Murray, 2007] for the calibration.

2.3.2 Radiometric Calibration

How does the camera obtain the brightness value from the incident scene irradiance? Radiometric calibration aims to answer exactly this question. The importance of radiometric calibration has been largely stressed when creating High Dynamic Range (HDR) images from a set of images taken with different exposure times. However, in camera tracking algorithms, most often it is ignored and it is implicitly assumed that the camera irradiance and brightness are linearly related. This is not always true and that a calibration is needed to work in the space of irradiance where (a) imaging artefact have been removed (b) the dependence of intensity on exposure time has been eliminated and also when modelling the camera noise that may affect the tracker when high frame-rate is used.

Camera radiometric calibration involves estimating the response function, f_{CRF}, that maps the irradiance value, E, incident on the camera's sensor cavity over a given time period Δt, to brightness value, B, observed in the image.

$$B = f_{CRF}(E\Delta t) \ . \tag{2.46}$$

The standard procedure as detailed in [Debevec and Malik, 1997] is to obtain images of a static scene under different exposures and fit a second order function to obtain the response function. Since f is monotonic and invertible we can map from a given brightness value and shutter time via the inverse CRF f^{-1} back into irradiance. For each pixel i at shutter time Δt_j, we take the logarithm:

$$f_{CRF}^{-1}(B_{ij}) = E_i \Delta t_j \ ; \tag{2.47}$$

$$\ln(f_{CRF}^{-1}(B_{ij})) = \ln(E_i) + \ln(\Delta t_j) \ . \tag{2.48}$$

Denoting a notational short-hand $g = \ln(f_{CRF}^{-1})$ and together with the second order smoothness that ensures that the response function is smooth, a cost function can be defined to assess the quality of fit

$$\psi = \overbrace{\sum (g(B_{ij}) - \ln(E_i) - \ln(\Delta t_j))^2}^{\textit{data term}} + \lambda \underbrace{\sum \left(\frac{\partial^2}{\partial B^2} g(B) \right)^2}_{\textit{smoothness term}} \ . \tag{2.49}$$

Since the pixel values observed are discrete, the second order derivative used in the cost function can be obtained via a standard 3-point stencil [7] as:

$$\lambda \sum (\mathrm{g}(B-1) - 2\mathrm{g}(B) + \mathrm{g}(B+1))^2 \ . \tag{2.50}$$

The constant λ is used as a weighing factor to balance the *smoothness term* against the cost associated with fitting the response function model on the data, commonly referred to as *data term* — a large value of λ would lead to over-smoothed curve while a small value would result in a curve that is very noisy, so a balance is sought usually by experimental trails. Unknowns that we wish to find out can be stacked and compactly represented in a vector **x** as:

$$\mathbf{x} \ = \ \begin{pmatrix} \mathrm{g} \\ \mathrm{E} \end{pmatrix} . \tag{2.51}$$

where g and E are both column vectors

$$\mathrm{g} = \begin{pmatrix} \mathrm{g}(1) \\ \mathrm{g}(2) \\ \vdots \\ \mathrm{g}(2^{\mathrm{b}}) \end{pmatrix} , \ \mathrm{E} = \begin{pmatrix} \ln(E_1) \\ \ln(E_2) \\ \vdots \\ \ln(E_i) \end{pmatrix} , \ \mathrm{t} = - \begin{pmatrix} \ln(\Delta t_1) \\ \ln(\Delta t_2) \\ \vdots \\ \ln(\Delta t_j) \end{pmatrix} . \tag{2.52}$$

yielding the corresponding matrices A described below as

$$\mathrm{A} \ = \ \begin{cases} 1 & \text{if } i \leq 2^b \text{ and } \mathrm{g}(B_{ij}) \text{ s.t. } j \in e \text{ and } i \in p \\ -1 & \text{if } i > 2^b \text{ and } i \in p \\ 0 & \text{otherwise} \end{cases} \tag{2.53}$$

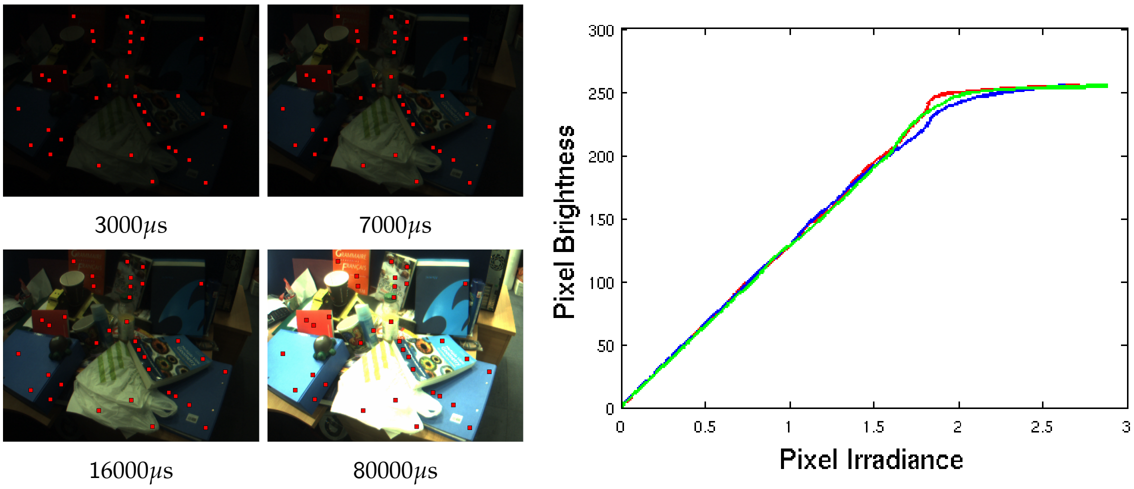
and the second derivative in the smoothness term can be described by the linear operator ∇^2 matrix

$$\nabla^2 \ = \ \begin{pmatrix} 1 & -1 & . & . & . & . & 0 \\ 1 & -2 & 1 & . & . & . & 0 \\ 0 & 1 & -2 & 1 & . & . & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & . & -1 & 1 \end{pmatrix} . \tag{2.54}$$

Therefore, the cost function ψ can then be rewritten as follows

$$\psi \ = \ (\mathrm{Ax} + \mathrm{t})^2 + \lambda(\nabla^2 \mathrm{x})^2 \ . \tag{2.55}$$

[7] A detailed explanation can be found here: http://en.wikipedia.org/wiki/Finite_difference_coefficients


3000μs

7000μs

16000μs

80000μs

Figure 2.4: Left: A selection of images obtained from a real camera with varying shutter time in microseconds. Red rectangles mark pixels manually selected to evenly sample scene irradiance whose brightness values are captured and used as input to CRF estimation. All images taken with zero gain and gamma off. Right: experimentally determined Camera Reponse Function (CRF) of our Basler piA640-210gc camera for each of the R, G and B colour channels with zero gain and gamma switched off using the method of [Debevec and Malik, 1997]. This camera has a remarkably linear CRF up until saturation; over most of the range image brightness can be taken as proportional to irradiance. Note that the irradiance values determined by this method are up to scale and not absolute.

Using a notational shorthand $C = \nabla^2$ and minimising with respect to x leads to the following system of linear equations

$$\mathrm{A}^T(\mathrm{A}\mathrm{x}+\mathrm{t}) + \lambda C^T(\mathrm{C}\mathrm{x}) = 0 \qquad \Longrightarrow \; \mathrm{x} = -(\mathrm{A}^T\mathrm{A} + \lambda \mathrm{C}^T\mathrm{C})^{-1}\mathrm{A}^T\mathrm{t}\,. \tag{2.56}$$

To calibrate our camera, we captured multiple images of a static scene with a range of known shutter times (see the left side of Figure 2.4), recording the changing brightness values at a number of pixel locations chosen to span the irradiance variation in the scene. Using measurements of 35 image pixels at 15 different shutter times, we solve for a parameterised form of f^{-1} under the $\mathcal{L}_2$ error norm with a second order smoothness prior. Figure 2.4 (right) shows the remarkably linear resulting CRF (camera gamma setting disabled) for the Basler piA640-210gc camera tested.

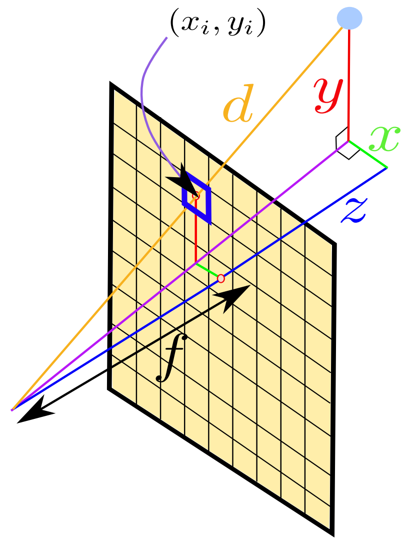
It is worth stressing that we only obtain f^{-1} up to scale and not in absolute irradiance units. However, as [Debevec and Malik, 1997] mention, it is possible to obtain irradiance in absolute units if we know more physical parameters of the camera. In our experiments, we normalise the brightness and as a result obtain the normalised irradiance.

2.4 POVRay Mathematics

We have used synthetic ray-tracing software, POVRay, to obtain ground-truth depth-map as well as the camera poses, in this thesis. The ray-tracer provides only the euclidean distance of a point in 3D to the camera as the ground-truth depth information. In our experiments, we need actual 3D coordinates from this depth information to be able to map them from one camera reference frame to the other. The next subsection describes the technical details associated with the conversion of this depth information to the z coordinate required. The subsequent subsections provide the code snippets for conversion of various meta-data provided by POVRay to the matrices needed often in computer vision.

2.4.1 Planar Depth from Euclidean Distance

Figure 2.5 illustrates the conversion of euclidean distance to depth (z-coordinate). Using identities from similar triangles one can relate the euclidean distance of the 3D point to the euclidean distance from the camera to its corresponding pixel location (x_i, y_i) [8] on the image plane, d_i.



$$\frac{d_i}{d} = \frac{f}{z} \implies z = f\,\frac{d}{d_i}$$

$$d_i = \sqrt{d_f^2 + y_i^2}$$

$$d_f = \sqrt{x_i^2 + f^2} \implies d_i = \sqrt{x_i^2 + f^2 + y_i^2}$$

$$\implies z = f\sqrt{\frac{d^2}{x_i^2 + f^2 + y_i^2}}$$

Figure 2.5: The euclidean depth provided by POVRay is converted to planar depth for our experiments. All rays converge to the camera. d_i denotes the euclidean distance from the camera to the pixel.

[8] The subscript i stands for association of a variable with image.

2.4.2 Getting Camera Parameters

```
function K = getcamK(cam_file)

f = fopen(cam_file, 'r') ;
if f ~= -1
  script=char(fread(f, inf, 'uchar')) ;
  eval(script) ;
  fclose(f) ;
end

focal   =  norm(cam_dir) ;
aspect =  norm(cam_right)   / norm(cam_up) ;
angle   =  2*atan( norm(cam_right)/2 / norm(cam_dir)  ) ;
height = 480; %cam_height
width   = 640; %cam_width

% pixel size
psx     = 2*focal*tan(0.5*angle)/width ;
psy     = 2*focal*tan(0.5*angle)/aspect/height ;
psx     = psx / focal;
psy     = psy / focal ;

Sx = psx;
Sy = psy;
Ox = (width+1)*0.5;
Oy = (height+1)*0.5;
f = focal;

K =       [1/psx     0     Ox;
             0    1/psy    Oy;
             0      0       1];
K(2,2) = -K(2,2);

end
```

2.4.3 Getting Camera Poses

```
function [R T] = computeRT(cam_file)

f = fopen(cam_file, 'r') ;
if f ~= -1
  script=char(fread(f, inf, 'uchar')) ;
  eval(script) ;
  fclose(f);
end

z = cam_dir / norm(cam_dir);
x = cross(cam_up,z);
x = x / norm(x);
y = cross(z,x);

R = [x y z];
T = cam_pos;

end
```

2.4.4 Sample Camera File

```
cam_pos       = [149.376, 451.41, -285.9]';
cam_dir       = [0.421738, -0.409407, 0.809026]';
cam_up        = [-0.0482194, 0.880868, 0.470899]';
cam_lookat    = [0, 0, 1]';
cam_sky       = [0, 1, 0]';
cam_right     = [1.20423, 0.316017, -0.467833]';
cam_fpoint    = [0, 0, 10]';
cam_angle     = 90;
```

2.4.5 Compute 3D positions

```
function [x,y,z] = compute3Dpositions(txt_file,depth_file)

    K = getcamK(txt_file);

    fx = K(1,1);
    fy = K(2,2);
    u0 = K(1,3);
    v0 = K(2,3);

    u = repmat([1:640],480,1);
    v = repmat([1:480]',1,640);

    u_u0_by_fx = (u - u0)/fx;
    v_v0_by_fy = (v - v0)/fy;

    z = load(depth_file);
    z = reshape(z,640,480)' ;  %z is radial
    z = z ./ sqrt(u_u0_by_fx.^2 + v_v0_by_fy.^2 + 1);

    x = ((u-u0)/fx).*z;
    y = ((v-v0)/fy).*z;

end
```

Chapter 3

Tracking Sparse Features

Contents

This chapter elaborates the joint work with our colleague and mentor Dr. Margarita Chli which led to a publication in Computer Vision and Pattern Recognition Conference in 2010. The idea of sparsifying graph into Chow-Liu tree (named CLAM) was developed by them.

Ankur Handa, Margarita Chli, Hauke Strasdat, Andrew J. Davison (2010). Scalable Active Matching. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

In this chapter, we focus on matching sparse point features aided by the correlations present among them that may come from the 3D structure and the camera motion. The underlying method that is developed very much stresses on the usage of the prior information that when injected reduces the space of possible feature matches (and potential false positives) and thereby leads to efficient matching.

3.1 Introduction

The task of image to image correspondences or model to image correspondences lies greatly at the heart of many problems that deal with estimating the transformation between images or pose of the moving entity. Particularly, in our sparse-SLAM style framework, camera tracking, which is performed by establishing correspondences between the sparse 3D model and their corresponding locations in the image forms the front-end of the system. In many such tasks and particularly in real-time tracking from video which we wish to focus on, there are generally strong priors available on absolute and relative correspondence locations largely due to camera motion models and 3D scene models.

3.1.1 Matching Using Priors

In most RANSAC style approaches, the first stage that establishes correspondences treats each candidate matching location independently and as a result the search of correspondence of these features is also carried out independently. Once putative correspondences are found for these features, a model of camera motion (or in general parameter model) is fit in the next stage to sift out the correspondences that agree with the model from this set — generally termed as *inliers*. Hence, the priors which we wish to use have been given very little or no attention in the past in approaches that establish correspondences.

Attempts have also been made to retrofit RANSAC algorithms with probabilistic tests and updates in the loop, leading to semi-probabilistic variants, for instance, KALMANSAC [Vedaldi et al., 2005], Guided-MLESAC [Tordoff and Murray, 2005] and 1-point RANSAC method [Civera et al., 2009]. [Raguram et al., 2009] also recently showed that modelling the uncertainty in the processes involved can greatly improve the quality of the RANSAC outcome. However, parts of these algorithms remain ad-hoc and unsatisfactory as they do not exploit the correlation of feature locations that can greatly aid the matching task. What we wish to focus on is a prior driven approach that side-steps the two stage procedure of RANSAC and does a global/joint estimation of correspondences and model parameters in the same loop. If the data was perfect and that there existed one and only one match per feature, it can be easily shown that using an Active Search approach [Davison, 2005], with decisions guided by Information theory, one can efficiently lock down the feature matches searching parsimoniously than independently searching for each feature in a defined search region. More sophisticated and improved decisions can be made about matching if we know the joint probability distribution of the features. The fact that in real time matching such priors can be trivially obtained from motion-models or 3D scenes makes this approach quite an appealing one. However, the Active Search approach was only a synthetic experiment to demonstrate the strength of the approach that does not treat each feature independently.

Active Matching [Chli and Davison, 2008] that emerged as the successor of Active Search dealt with its inability to handle more than one match for a feature by maintaining hypothesis for each feature match found. It is again an Information Theory driven approach that borrows all the related formal theory from Active Search and decides the best feature and the best hypothesis to measure based on the expected Information gain in the location of the remaining features. This is in contrast to the decisions made in the first stage of RANSAC style methodologies where a fixed threshold is used to find out only the best quality match. The hypotheses are maintained in the form of Gaussian distributions represented by the mean of the feature locations and the covariances denoting the search region around which we expect the feature to lie in. The cross-covariance elements denote the degree of dependencies of features among each other. Therefore, each feature measurement means updating these covariance matrices via standard bayesian update rule to absorb the information gained. Successive measurements provide evidence to either refute or boost a given hypothesis leading to finding the best hypothesis within the loop of search for feature matches instead of a two-step approach taken in RANSAC style methods.

In limit, ultimate performance in matching is represented by the determination of a fully dense correspondence field between images, or at least between parts of them which observe common regions of the scene. It is feasible to aim to obtain such dense correspondence information in cases where it can be assumed that the changes between two images to be matched are relatively small. This is the subject of the well known field of optical flow estimation, and there have recently been significant advances in this area. For instance, highly impressive results have been achieved in fully dense optical flow using variational optimisation, achieving real-time operation on the latest GPU processors [Zach et al., 2007]. In such methods, the assumption of small motion permits highly effective regularisation terms to 'fill in' the correspondence field, even in areas of low texture.

The regularisation term in optical flow algorithms is one example of a prior used in matching, encoding usually the assumption that the inter-frame displacement of nearby pixels will tend to vary gradually in regions of gently varying image intensity, since most scenes consist of real continuous objects relative to the size of which any motion (of scene or camera) is fairly small. Matching over wider baselines, or without such lavish processing resources, cannot usually aim to be completely dense. Instead, correspondence is generally sought only between salient 'features', parts of the image which can be characterised by descriptors with some degree of invariance to transformations. In sequential camera tracking, although frame-to-frame camera motion may be small, it is desirable to track features through as many frames as possible to best constrain camera motion estimates.

Once the aim of fully dense correspondence is reduced to that of matching a set of distinct features spread across the image, of course priors are still available on the image locations of these features. The level of prior information which will be available depends strongly on the domain knowledge present in the problem. Suppose that it is desired to match features between two images and all that is known is they are consecutive video frames taken by a moving camera — then the priors we can assume will be a distributed version of those used in optical flow estimation. On the other hand, when matching as part of sequential camera tracking system with rolling 3D camera and position estimates, strong correlated predictions of the image positions will be available.

This was precisely the situation where Active Matching (AM) algorithm [Chli and Davison, 2008, Chli and Davison, 2009a] was demonstrated, as the matcher in a filtering-based monocular Simultaneous Localisation and Mapping (SLAM) system [Davison et al., 2007]. Matching priors are built into the heart of this algorithm. The joint distribution on feature locations they predict is explicitly projected into the image, used to make decisions guided by information theory about which features to measure when, and incrementally refined towards a matching posterior as measurement results come in. Using a mixture of Gaussians representation to represent and refine multiple hypotheses, and taking into account of per-feature appearance statistics if required, AM demonstrated similar accuracy but much improved computational performance compared to the older probabilistic technique for data association Joint Compatibility Branch and Bound (JCBB) [Neira and Tardós, 2001].

While AM is therefore technically appealing, detailed performance analysis presented in [Chli and Davison, 2009a] has revealed its poor scalability of the with the number of features to be matched per frame. AM or other fully probabilistic matching algorithms have previously not proven their ability to handle hundreds of matches in real-time due to the costly overhead of intermediate Bayesian calculations. RANSAC and variants gain ground in making random decisions simply because there is as such no need to maintain any hypothesis information or big matrices to encode correlations. Therefore, AM's weakness was that the overhead induced by intermediate Bayesian updates required meant poor scaling to cases where many correspondences were sought. We later show that relaxation of the rigid probabilistic model of AM, the fully dense canonical graph of correlations, where every feature measurement directly affects the prediction of every other, permits dramatically more scalable operation without affecting accuracy. We take a general graph-theoretic view of the structure of prior information in matching to sparsify and approximate the interconnections. We demonstrate the performance of a new variation, termed as SubAM (*subset* Active Matching), in the context of sequential camera tracking. This algorithm is highly competitive with other techniques at matching hundreds of features per frame while retaining great intuitive appeal and the full probabilistic capability to digest prior information.

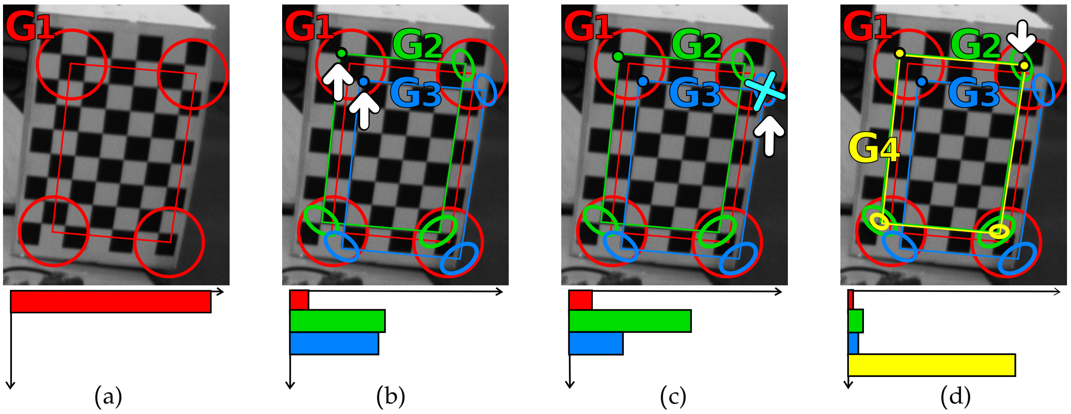
In the following we briefly review the mathematical minutiae related to Active Matching, the computation of Information a feature can provide, and the predict, update rules for the necessary information gained on measuring a feature.

3.2 The Active Matching Paradigm

A naïve approach to matching features that is most commonly used in wide baseline matching, begins with the assumption that no prior information is available about the relations among features, treats each feature independently before relegating the burden of sifting wrong matches out from inliers to a next stage that fits either the parameter model to recover the transformation between two images or 6DoF camera pose motion model.

Not only does this kind of matching rely on very minimal or no prior information, but also loses the ability to handle mismatches in the first stage. In most cases, particularly when dealing with image stream coming from a moving camera, it would be incredibly naïve not to exploit the prior information. Such prior usually is available in the form of joint probability distribution of the location of 3D features in the image. Active Matching targets exactly this sort of matching where prior information is readily available and can be used to constrain the matching task. Particularly, when a feature is matched, the joint probability distribution allows to constrain the uncertainty of other features that are yet to be measured. Repeated trails of such matching procedure progressively decrease the uncertainty of the location of the unmeasured features and very quickly one arrives to a point where one becomes nearly very certain of the location of the rest of relatively many unmeasured features.

Given a new image, AM [Chli and Davison, 2008] sets its initial matching search-state to



(a) (b) (c) (d)

Figure 3.1: A mini example of AM [Chli and Davison, 2008]. The initial search-state $\mathbf{G}_1$ in (a) describes the expected configuration of matches. In (b) a search is made for the top-left of the four predicted features, and the two candidate matches found cause the spawning of new Gaussian hypotheses $\mathbf{G}_2$ and $\mathbf{G}_3$, pushing the weight λ_1 of $\mathbf{G}_1$ down (illustrated in the histogram). A failed search in $\mathbf{G}_3$ in (c) reduces λ_3, while the match in (d) spawns $\mathbf{G}_4$ which becomes the dominant hypothesis.

the input probabilistic prior $p(\mathbf{z})$ over the image locations $\mathbf{z} = (\mathbf{z}_a, \mathbf{z}_b, \ldots)^\top$ of the measurable features. The evidence gathered by measuring features one-by-one causes progressive updates in the search-state. A mixture of Gaussians is employed to handle the multiple matching-hypotheses arising. Each Gaussian $\mathbf{G}_k$ has an associated probability λ_k of representing the true scenario:

$$p(\mathbf{z}) = \sum_{k=1}^{K} p(\mathbf{z}^k) = \sum_{k=1}^{K} \lambda_k \mathbf{G}_k, \text{ where } \sum_{k=1}^{K} \lambda_k = 1 \tag{3.1}$$

where we have now used the further notational shorthand $\mathbf{G}_i = \mathbf{G}(\mathbf{x}_m^i, \mathsf{P}_{\mathbf{x}_m^i})$. The algorithm follows a predict-measure-update loop which terminates when all features have been searched for and the mixture converges to a probabilistically dominant Gaussian. The priors that arise from representing the joint probabilistic distribution modelled by Gaussians over the positions of the features and camera pose represented in a state vector shown below:

$$\hat{\mathbf{x}}_m = \begin{pmatrix} \hat{\mathbf{x}} \\ \mathbf{z}_1 \\ \mathbf{z}_2 \\ \vdots \end{pmatrix} = \begin{pmatrix} \hat{\mathbf{x}} \\ \mathbf{h}_1(\hat{\mathbf{x}}) \\ \mathbf{h}_2(\hat{\mathbf{x}}) \\ \vdots \end{pmatrix}, \mathsf{P}_{\mathbf{x}_m} = \begin{bmatrix} \mathsf{P}_x & \mathsf{P}_x \frac{\partial \mathbf{h}_1}{\partial \mathbf{x}}^T & \mathsf{P}_x \frac{\partial \mathbf{h}_2}{\partial \mathbf{x}}^T & \ldots. \\ \frac{\partial \mathbf{h}_1}{\partial \mathbf{x}} \mathsf{P}_x & \frac{\partial \mathbf{h}_1}{\partial \mathbf{x}} \mathsf{P}_x \frac{\partial \mathbf{h}_1}{\partial \mathbf{x}}^T + \mathsf{R}_1 & \frac{\partial \mathbf{h}_1}{\partial \mathbf{x}} \mathsf{P}_x \frac{\partial \mathbf{h}_2}{\partial \mathbf{x}}^T & \ldots. \\ \frac{\partial \mathbf{h}_2}{\partial \mathbf{x}} \mathsf{P}_x & \frac{\partial \mathbf{h}_2}{\partial \mathbf{x}} \mathsf{P}_x \frac{\partial \mathbf{h}_1}{\partial \mathbf{x}}^T & \frac{\partial \mathbf{h}_2}{\partial \mathbf{x}} \mathsf{P}_x \frac{\partial \mathbf{h}_2}{\partial \mathbf{x}}^T + \mathsf{R}_2 & \ldots. \\ \vdots & \vdots & \vdots & \end{bmatrix} \tag{3.2}$$

The lower-right portion of $\mathsf{P}_{\mathbf{x}_m}$ representing the covariance of is known as the innovation covariance matrix S in Kalman filter tracking. The correlations between different candidate measurements mean that generally S will not be block-diagonal and contains off-diagonal correlations between the predicted measurements of different features. With this single Gaussian formulation, the mutual information in bits between any two partitions α and β of $\mathbf{x}_m$ can be calculated according to this formula:

$$I(\alpha;\beta) = \frac{1}{2}\log_2 \frac{|\mathsf{P}_{\alpha\alpha}|}{|\mathsf{P}_{\alpha\alpha} - \mathsf{P}_{\alpha\beta}\mathsf{P}_{\beta\beta}^{-1}\mathsf{P}_{\beta\alpha}|} \tag{3.3}$$

where $\mathsf{P}_{\alpha\alpha}$, $\mathsf{P}_{\alpha\beta}$, $\mathsf{P}_{\beta\alpha}$ and $\mathsf{P}_{\beta\beta}$ are sub-blocks of $\mathsf{P}_{\mathbf{x}_m}$. This representation however can be computationally expensive as it involves matrix inversion and multiplication so exploiting the properties of mutual information we can reformulate into:

$$\begin{aligned} I(\alpha;\beta) &= H(\alpha) - H(\alpha|\beta) = H(\alpha) + H(\beta) - H(\alpha,\beta) && (3.4)\\ &= \frac{1}{2}\log_2 \frac{|\mathsf{P}_{\alpha\alpha}||\mathsf{P}_{\beta\beta}|}{|\mathsf{P}_{\mathbf{x}_m}|} && (3.5) \end{aligned}$$

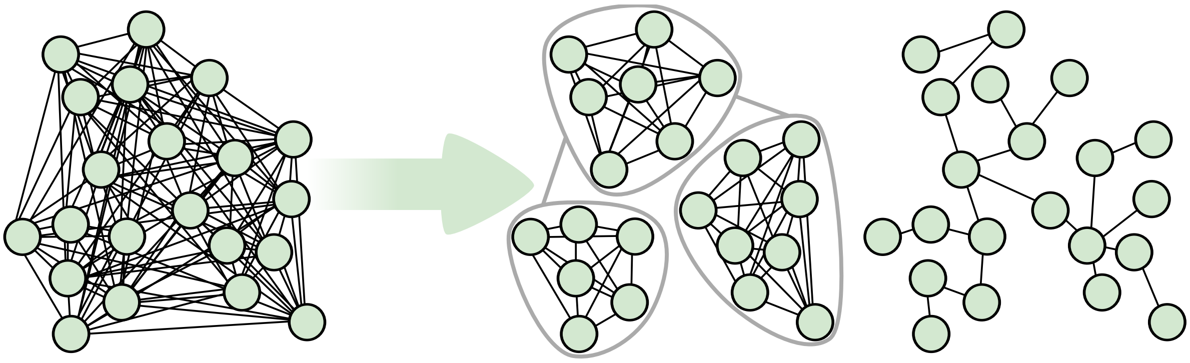
Below we describe each of these stages briefly which can be visualised in the example of Figure 3.1:

- **Predict:** Evaluate the expected utility (in terms of mutual information) of all measurement candidates in terms of how much they should help to resolve the ambiguity and decrease variance in the mixture.

- **Measure:** Search for template matches corresponding to the candidate predicted to provide the most mutual information per pixel needed to search (*i.e.* its 3σ gated ellipse).

- **Update:** Redistribute the weights according to the new evidence obtained (*e.g.* if no match was found, then λ_i of the measured $\mathbf{G}_i$ should diminish). If the measurement stage yields M matches, M new Gaussians get spawned each to represent that one of these matches corresponds to the true feature, while $\mathbf{G}_i$ is updated to represent that all M matches are false-positives. Finally, any Gaussians with very weak weights get pruned off the mixture.

These three steps are performed repeatedly until all the features are measured in the most dominant Gaussian. While AM exhibits great robustness to mismatches following the probabilistic maintenance of hypotheses, it spends precious processing time into 'thinking' of where to look for matches next. Following this realisation, here study different sparsifications of the joint input prior to provide more scalability to the matching as illustrated in Figure 3.2 and 3.3.



(a) Complete Graph (b) Tree of clusters (c) Tree of nodes

Figure 3.2: Representing matching priors, the predicted joint distribution over image feature locations as a graph, and sparsifying it for efficiency. Considering this distribution as a graph of measurement predictions and correlation potentials, we aim to sparsify the complete graph as considered in AM with a tree of clusters in SubAM and a tree of nodes in CLAM.

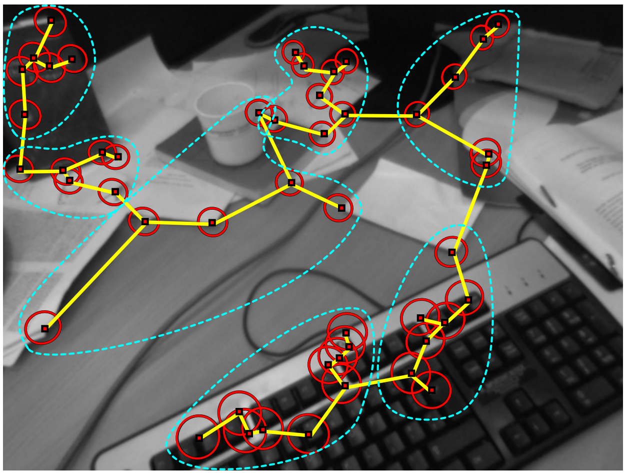


Figure 3.3: Approximating the joint prior distribution over feature predictions in matching using graphs. In our algorithms we simplify a fully connected graph to a unit-width tree (CLAM), or a tree of subsets (SubAM) to achieve real-time matching of many features.

3.3 Feature Matching Priors

Matching priors are expressed as a joint distribution on the predicted positions of features in an image before any image processing is done. Generally, these priors encode strong correlation information between the predictions which is the key to robust consensus matching. We describe in the following general connectivity rules and updates in the graph.

3.3.1 Probabilistic Predictions in a Graph

The effect of correlations can be visualised as a network of springs connecting feature predictions. Pinning down the exact location of one feature in the image $\mathbf{z}_a$ will result in an associated shift in the rest of the predictions. Formalising this analogy, we consider the joint prior $p(\mathbf{z})$ as a general graph structure where nodes correspond to individual feature predictions $\mathbf{z}_a$ and edges represent the correlation potentials between these nodes.

In order to model $p(\mathbf{z})$ with a Gaussian $\mathbf{G} = \{\hat{\mathbf{z}}, \mathtt{S}\}$ we construct the mean $\hat{\mathbf{z}}$ and covariance matrix $\mathtt{S}$ consulting the input graph structure: each partition $\hat{\mathbf{z}}_a$ holds the predicted image location of node $\mathbf{z}_a$, while block $\mathtt{S}_{aa}$ [1] describes the uncertainty in $\hat{\mathbf{z}}_a$. The potential of the link shared between $\mathbf{z}_a$ and $\mathbf{z}_b$ is stored in $\mathtt{S}_{ab}$.

While in principle $\mathtt{S}$ is a dense matrix we need not estimate the covariance blocks of any nodes not sharing direct links as these links will never be used to propagate information (as shown in Figure 3.2). Note that in the language of Kalman filter tracking, $\mathtt{S}$ is the 'innovation covariance' and is explicitly available at every frame.

3.3.2 Mutual Information Between two Candidates Feature Locations

AM looks for matches *on demand* while searching for consensus in a process driven by Mutual Information (MI): at every step it chooses to measure the candidate predicted to provide the highest MI to the current matching state, divided by the number of pixels needed to search for image processing. As defined in Shannon Information Theory, MI provides a measure of the expected reduction in uncertainty in the current state upon part observation of this state.

How much information is a potential measurement for $\mathbf{z}_b$ predicted to provide to prediction $\mathbf{z}_a$? The **P**airwise MI score quantifies this information as:

$$I(\mathbf{z}_b; \mathbf{z}_a) = E\left[\log_2 \frac{p(\mathbf{z}_a|\mathbf{z}_b)}{p(\mathbf{z}_a)}\right] = \frac{1}{2}\log_2 \frac{|\mathtt{S}_{aa}||\mathtt{S}_{bb}|}{|\mathtt{S}_{a,b}|}\ , \tag{3.6}$$

where $\mathtt{S}_{a,b}$ is the joint covariance of both $\mathbf{z}_a$, $\mathbf{z}_b$. As explained in [Chli and Davison, 2009b] this score provides an absolute, normalised measure of the correlation between any two measurement candidates. Transforming all the correlation potentials into Pairwise MI links, we can form a 'MI graph'. It is important to note here that the MI score used in AM

[1] $\mathtt{S}_{aa}$ describes an ellipse in image space, often referred to as the 'active search' region for feature $\mathbf{z}_a$

is different as there we consider the information shared between $\mathbf{z}_b$ and the rest of the candidates in $\mathbf{z}$ (*i.e.* $I(\mathbf{z}_b; \mathbf{z}_a,\ \mathbf{z}_c, \dots)$).

3.4 CLAM: Chow Liu Active Matching

As a first attempt to tackle the costly manipulation of large, dense matrices in AM we considered thinning the complete graph of the joint prediction $p(\mathbf{z})$ into a singly-connected tree as in Figure 3.2(c). While this can indeed be a big approximation, careful selection of the edges preserved can be very beneficial to the closeness of approximation.

3.4.1 The Chow-Liu Tree

A joint density over $\mathbf{z} = (\mathbf{z}_1,\ \mathbf{z}_2, \dots, \mathbf{z}_n)^\top$ can be approximated with to a tree-shaped model via factorisation:

$$p(\mathbf{z}) = p(\mathbf{z}_n) \prod_{i=1}^{n-1} p(\mathbf{z}_i | \mathbf{z}_{i+1} \dots \mathbf{z}_n) \approx p(\mathbf{z}_n) \prod_{i=1}^{n-1} p(\mathbf{z}_i | \mathbf{z}_{i+1}). \tag{3.7}$$

Out of all such tree factorisations, Chow and Liu [Chow and Liu, 1968] showed that the optimal approximation can be formed by retaining the links corresponding to the maximum spanning tree[2] of the complete MI graph (as defined in Section 3.3.2).

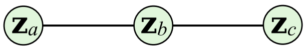
Inspired by the power of the Chow-Liu (CL) tree to capture the most representative correlation structure in the scene in [Chli and Davison, 2009b], here we propose using it to represent the distribution of expected feature locations input to AM in our new Chow Liu Active Matching (CLAM) algorithm. While in AM the update stage involves costly EKF-updates, the simple tree structure in CLAM allows Belief Propagation (BP) updates of $O(n)$ in the worst case.

3.4.2 Belief Propagation for CLAM

Given observations for some tree nodes, BP provides exact inference computing marginals for all other nodes by recursively propagating messages along the tree. Bishop [Bishop, 2006] discussed how a full update requires two passes of the tree (from the leaves

[2] The acyclic path connecting all nodes in a weighted graph which yields the maximal sum of weights.

to the root and back) so that every node receives updates from all its neighbours. However, here we only observe one node at a time which permits updates in a single pass by designating the measured node as the root and propagating messages all the way to the ends of the tree.



The key idea behind the BP methodology is the exploitation of the properties of d-separation: there is only one path between any two nodes in the tree, hence an update-message originating from observed node $\mathbf{z}_a$ is bound to update the probability distributions of any intermediate nodes in the way until it reaches its final destination, node $\mathbf{z}_c$.

Propagating Updates

Let us consider the case that the joint distribution $p(\mathbf{z})$ of this tree is described by a Gaussian $\mathbf{G} = \{\hat{\mathbf{z}}, \mathtt{S}\}$ partitioned as follows:

$$\hat{\mathbf{z}} = \begin{pmatrix} \hat{\mathbf{z}}_a \\ \hat{\mathbf{z}}_b \\ \hat{\mathbf{z}}_c \end{pmatrix}, \; \mathtt{S} = \begin{bmatrix} \mathtt{S}_{aa} & \mathtt{S}_{ab} & \mathtt{S}_{ac} \\ \mathtt{S}_{ba} & \mathtt{S}_{bb} & \mathtt{S}_{bc} \\ \mathtt{S}_{ca} & \mathtt{S}_{cb} & \mathtt{S}_{cc} \end{bmatrix} . \tag{3.8}$$

Given the observation $\mathbf{z}_a = \mathbf{a}$, applying Schur's complement on the $\mathtt{S}$ we can obtain the conditioned covariance:

$$\begin{bmatrix} \mathtt{S}_{bb|a} & \mathtt{S}_{bc|a} \\ \mathtt{S}_{cb|a} & \mathtt{S}_{cc|a} \end{bmatrix} = \begin{bmatrix} \mathtt{S}_{bb} & \mathtt{S}_{bc} \\ \mathtt{S}_{cb} & \mathtt{S}_{cc} \end{bmatrix} - \begin{bmatrix} \mathtt{S}_{ba} \\ \mathtt{S}_{ca} \end{bmatrix} \mathtt{S}_{aa}^{-1} \begin{bmatrix} \mathtt{S}_{ab} & \mathtt{S}_{ac} \end{bmatrix} , \tag{3.9}$$

while similar update-rules apply for the means vector:

$$\begin{pmatrix} \hat{\mathbf{z}}_{b|a} \\ \hat{\mathbf{z}}_{c|a} \end{pmatrix} = \begin{pmatrix} \hat{\mathbf{z}}_b \\ \hat{\mathbf{z}}_c \end{pmatrix} - \begin{bmatrix} \mathtt{S}_{ba} \\ \mathtt{S}_{ca} \end{bmatrix} \mathtt{S}_{aa}^{-1} \left(\hat{\mathbf{z}}_a - \mathbf{a} \right) . \tag{3.10}$$

However, the block $\mathtt{S}_{ca}$[3] is not explicitly known since $\mathbf{z}_a$ does not share a direct link with $\mathbf{z}_c$. Considering the effect of propagating a measurement for $\mathbf{z}_b$ instead and enforcing $\mathtt{S}_{ca|b} = \mathbf{0}$ (since $\mathbf{z}_{a|b}$ and $\mathbf{z}_{c|b}$ become independent), one can arrive to the expression $\mathtt{S}_{ca} = \mathtt{S}_{cb}\mathtt{S}_{bb}^{-1}\mathtt{S}_{ba}$. Substituting for $\mathtt{S}_{ca}$ back to (3.9) and (3.10) it becomes evident that:

[3] Note that $\mathtt{S}_{ca} = \mathtt{S}_{ac}^{\top}$ since $\mathtt{S}$ is symmetric

$$
\begin{aligned}
\mathtt{S}_{cc|a} &= \mathtt{S}_{cc} - \mathtt{S}_{cb}\mathtt{S}_{bb}^{-1}\left(\mathtt{S}_{bb} - \mathtt{S}_{bb|a}\right)\mathtt{S}_{bb}^{-1}\mathtt{S}_{bc} && (3.11)\\
\mathtt{S}_{bc|a} &= \mathtt{S}_{cc} - \mathtt{S}_{cb}\mathtt{S}_{bb}^{-1}\left(\mathtt{S}_{bb} - \mathtt{S}_{bb|a}\right) && (3.12)\\
\hat{\mathbf{z}}_{c|a} &= \hat{\mathbf{z}}_{c} - \mathtt{S}_{cb}\mathtt{S}_{bb}^{-1}\left(\hat{\mathbf{z}}_{b} - \hat{\mathbf{z}}_{b|a}\right) \; . && (3.13)
\end{aligned}
$$

The above expressions demonstrate the recursive nature that the updates can have, since when evaluating $p(\mathbf{z}_c|\mathbf{z}_a)$ one can use the moments $\{\hat{\mathbf{z}}_{b|a}, \mathtt{S}_{bb|a}\}$ of $p(\mathbf{z}_b|\mathbf{z}_a)$. Hence, $\mathtt{S}$ needs to contain explicit entries only for nodes sharing a direct link in the CL-tree. Interestingly, upon successful measurement of a feature $\mathbf{z}_b$, then the Gaussian spawned to represent the hypothesis that the match obtained is a true-positive will have zero $\mathtt{S}$-blocks for the links of $\mathbf{z}_b$ to zero, essentially isolating $\mathbf{z}_b$ from the rest of the tree. As a result, the problem of matching is progressively broken down in smaller sub-trees reducing the computation time greatly.

Evaluating MIs

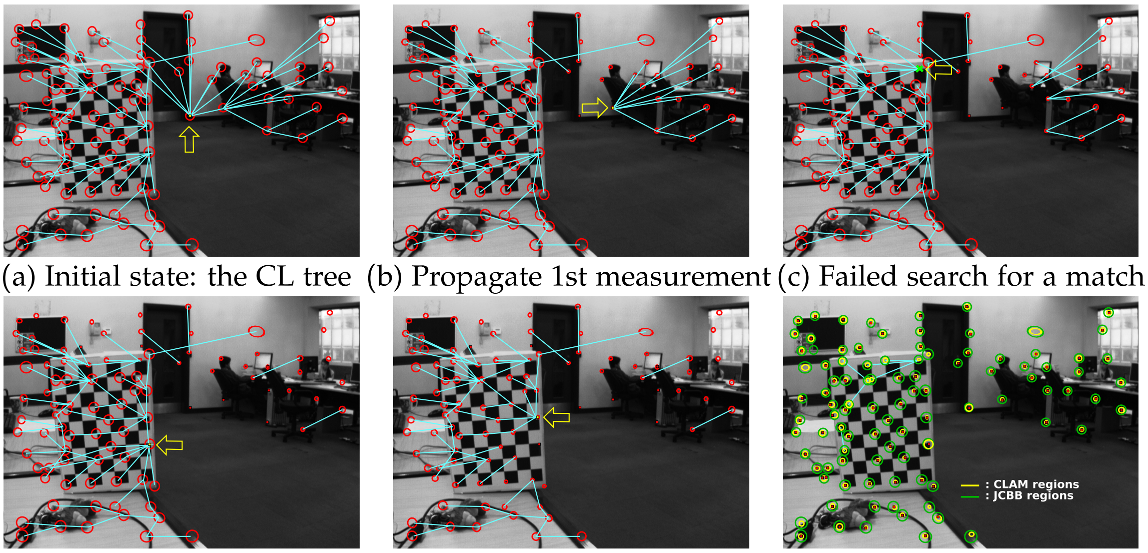
The flow of information along the branches of tree using BP has even more attractive properties when evaluating MIs of candidates. Let us consider the MI that $\mathbf{z}_a$ can give to the rest of the tree nodes in our tree example above:

$$
I(\mathbf{z}_b; \mathbf{z}_a, \mathbf{z}_c) = \int_{\mathbf{Z}} p(\mathbf{z}_a, \mathbf{z}_b, \mathbf{z}_c) \log_2 \frac{p(\mathbf{z}_b, \mathbf{z}_c|\mathbf{z}_a)}{p(\mathbf{z}_b, \mathbf{z}_c)} d\mathbf{z} \; . \tag{3.14}
$$

Applying Bayes' rule on the ratio inside the logarithm:

$$
\frac{p(\mathbf{z}_a, \mathbf{z}_b, \mathbf{z}_c)}{p(\mathbf{z}_a)p(\mathbf{z}_b, \mathbf{z}_c)} = \frac{p(\mathbf{z}_a)p(\mathbf{z}_b|\mathbf{z}_a)p(\mathbf{z}_c|\mathbf{z}_b)}{p(\mathbf{z}_a)p(\mathbf{z}_b)p(\mathbf{z}_c|\mathbf{z}_b)} = \frac{p(\mathbf{z}_b|\mathbf{z}_a)}{p(\mathbf{z}_b)} . \tag{3.15}
$$

The general rule that arises from further investigation into more complex tree structures is that the MI of a given node with the rest of the variables in a tree is equal to the MI it shares with its immediate neighbours alone. As a result, the evaluation of MIs in CLAM becomes trivial: the costly manipulation of the full covariance matrix necessary in AM gets replaced by a few fast message-passing operations within the sub-tree spanning the candidate node and its immediate neighbours only. Moreover, due to the partitioning of the tree into smaller sub-trees while matching, the MI scores of any sub-trees not updated within a particular matching-iteration can be carried forward to the next step.



(a) Initial state: the CL tree (b) Propagate 1st measurement (c) Failed search for a match

(d) Measure a hub-like feature (e) Updated state (f) Searched area

Figure 3.4: Matching using CLAM. The prior distribution and the computed CL tree are illustrated in (a). The arrow points at the feature selected for measurement by MI. Propagating the match found in (a) results in cuts of links in (b) and reduction of variance for the rest of the features. The match found in (b) yields updates in (c) for that subtree only. The failed search for matches in (c) preserves the same tree structure in (d). Finally (f) demonstrates the reduced regions searched in CLAM *w.r.t.* conventional methods like RANSAC or JCBB. Note that the CL tree links projected in every image correspond to the most probable Gaussian for the sake of clarity, while more Gaussians emerged in the mixture during matching.

3.4.3 CLAM: A Step-By-Step Example

Figure 3.4 illustrates a step-by-step example of the way CLAM works within a given frame. Given the joint prior $p(\mathbf{z})$, we compute all the pairwise MI links and then sparsify the MI graph to form the CL tree as shown in Figure 3.4(a). Any features tracked consistently and moving coherently throughout the sequence share strong correlations hence they lie close to each other in the tree space (*e.g.* the features on the checker-board). Following the measurement of the hub-like feature selected by MI in 3.4(b), we propagate update-messages causing reductions in uncertainty of different magnitude to all other nodes depending on their closeness in the tree structure. Since no more information can now be passed though the matched node, any links connecting it to the rest of the tree can be cut. As a result, matching is essentially partitioned into subtrees corresponding to different parts of the image, which are highly intercorrelated. Subsequent measurements and updates of the distribution result in successful matching for this frame as shown in (f) where the searched regions of CLAM and traditional methods like JCBB are superimposed.

3.5 SubAM: Subset Active Matching

The tree approximation of the joint prior in CLAM achieves a dramatic reduction in timings with respect to AM as demonstrated in Section 3.7 allowing real-time matching for hundreds of features. However, due to the fact that MI guides the division of the matching problem into smaller subtrees no particular care is taken to balance the size of the partitions. As a result, CLAM becomes unsuitable for super-dense online matching. Following this realisation, we developed Subset Active Matching (SubAM) which explicitly aims at balanced partitions into subsets connected in a tree (*e.g.* Figure 3.2(b)). All correlation links between features of the same subset are preserved as well as any links shared by features belonging to subsets in the order they get measured.

3.6 Assumptions

We have used the following assumption while carrying out our experiments.

1. The correlations can come from a prior information coming from a either a SLAM system or an algorithm that maintains the history of the pixels or features seen so far.

2. These correlations are generally stronger compared to the standard regularisation priors generally used in optical flow and stereo matching. Most importantly, they also capture the long range dependencies among pixels locations, which is generally captured by series of conditionally dependent short range dependencies in standard optical flow methods.

3.6.1 The SubAM Algorithm

Our new SubAM algorithm aimed at matching relatively larger set of features works by matching in clusters of features progressively. To obtain clusters (we used clusters and subsets interchangeably here), we construct Chow-Liu tree from the input prior $\mathbf{G}_1$ and then form groups of features ('subsets') by considering their proximity in the tree: given a target group-cardinality c (represents a trade off between timings and goodness of the approximation), we place partitions at nodes where the number of their descendants not already grouped is $\geq c - 1$. Note that this strategy can lead to subsets smaller than c, however we maintain that the minimum size is strictly c_{min}. This is exposited in Figure 3.5. Although we believe a more sophisticated algorithm could be designed to obtain clusters, we found that our heuristic method quite satisfactory in our experiments purely for the sake of efficiency. Similar ideas to that in SubAM have been used in other recent matching algorithms which

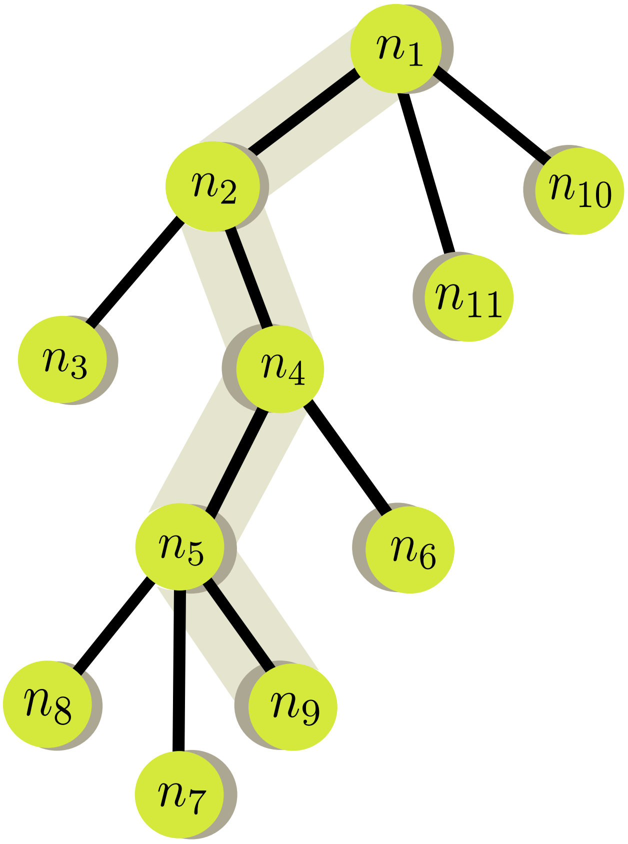
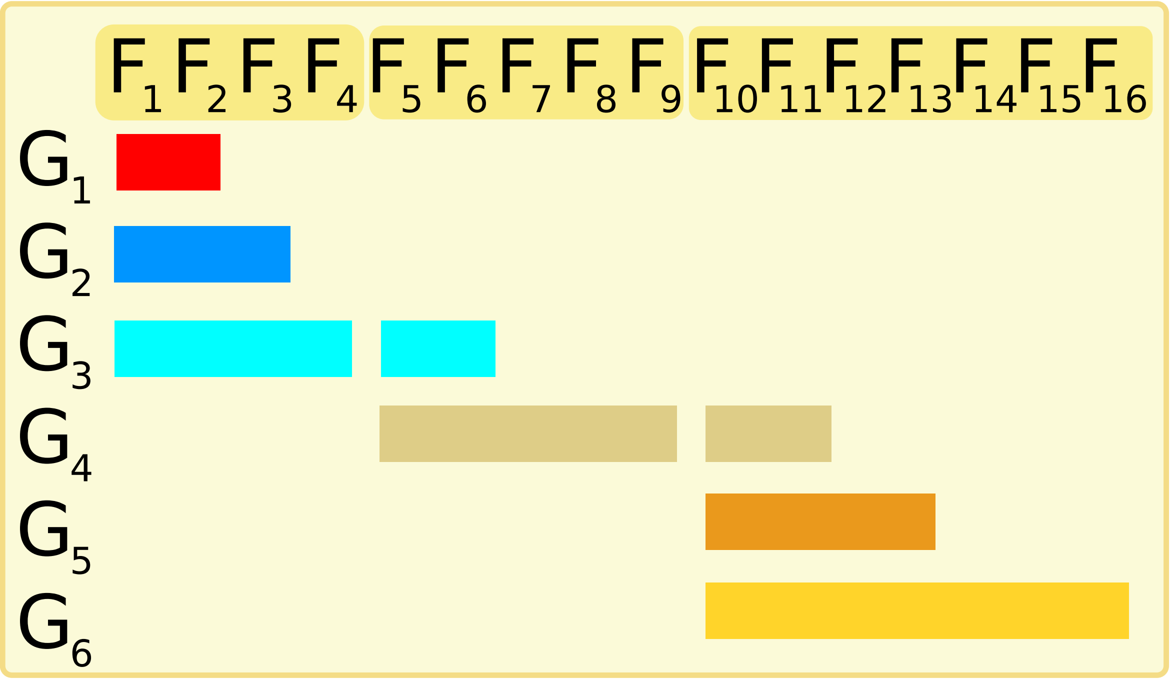


Figure 3.5: The Chow-Liu tree in the first stage is used to cluster the features and a new cluster tree is built that maintains the similar edge structure as the Chow-Liu tree maintained at the root nodes of the cluster. The traversal on the tree is depth first order so that clusters having nodes with high connectivity are visited first. In order to ensure that the algorithm does not get quickly over-confident we maintain the Gaussian hypotheses for a longer period by keeping a very small threshold for pruning.

perform matching cluster-by-cluster. The N3M algorithm [Hinterstoisser et al., 2007] defines groups of nearby features which have one more than the minimum number of feature members needed to offer their own consistent pose estimate, but these definitions are not as rigorously founded as our information theoretical measures. [Chum and Matas, 2005] in their PROSAC provide an approach to speed up the standard RANSAC based matching by randomly sampling from set of features selected based on the quality scores. All features are ranked based on their quality score. Under the assumption that features with high quality score are more likely to be correct, PROSAC begins with a smaller set of these features and progressively adds in more features so that high quality sets are tested earlier to speed up RANSAC. GroupSAC [Ni et al., 2009], on the other hand, clusters candidate features based on cues such as similar optical flow vectors. However, such clustering relies on exactly the kind of blanket image processing which we wish to side-step in our method. Instead, we show that useful clusters for matching can be determined just from matching priors, before the image data has even been accessed. Such sort of clustering captures the essence of dependencies of features within and among them. Below we describe our SubAM algorithm in the form of pseudo code and outline the various different stages it goes through:

```
SubAM($\mathbf{G}_1$)
mixture = $[[1, \mathbf{G}_1]]$  (each entry is a $[\lambda_i, \mathbf{G}_i]$ tuple)
T = find_tree_of_subsets($\mathbf{G}_1$)
$\mathbf{V}$ = []  (to hold all subsets visited by SubAM)
for $\forall\ \mathbf{s}_i \in$ T  (selecting $\mathbf{s}_i$ in a depth-first manner)
    mixture = AM(mixture, $\mathbf{s}_i$)
    $\mathbf{V}$ = append($\mathbf{s}_i$, $\mathbf{V}$)
    $\mathbf{G}_{best}$ = get_most_probable_$\mathbf{G}$(mixture)
    while $\exists\ \mathbf{f} \in \mathbf{V}$: is_unmeasured($\mathbf{f}$,$\mathbf{G}_{best}$)
        $\mathbf{s}_j$ = get_subset_of_$\mathbf{f}$($\mathbf{f}$)
        mixture = AM(mixture, $\mathbf{s}_j$)
        $\mathbf{G}_{best}$ = get_most_probable_$\mathbf{G}$(mixture)
    end while
end for
return $\mathbf{G}_{best}$
```

Having formed the tree T of subsets $\mathbf{s}_i$ (preserving the same hierarchy as in the CL tree) we attempt matching of subsets progressively postponing the propagation of updates until a given cluster is measured. Starting with the root subset, we traverse T in a depth first manner and perform full AM but **o**nly limited to features in the examined $\mathbf{s}_i$. This means that while each Gaussian in the mixture has a representation for every feature in the image, one AM process is only allowed to operate within the part of each Gaussian corresponding to the features in $\mathbf{s}_i$. Following an AM step, the most probable Gaussian of the mixture $\mathbf{G}_{best}$ is checked for any unmeasured features belonging to already visited subsets $\mathbf{V}$.If $\mathbf{G}_{best}$ has all visited features measured, then we can confidently propagate the probabilistic state to the next subset, otherwise the algorithm seeks to measure the features until all the features in all the visited subsets have been measured. Note that in the latter case, the nature of the matches obtained might reveal a different Gaussian as the most probable one, leading to a reassignment of $\mathbf{G}_{best}$. Figure 3.6 shows the propagation of Gaussians among clusters. Gaussians $\mathbf{G}_2$ and $\mathbf{G}_3$ are spawned by Gaussian $\mathbf{G}_1$ and as a result inherit all the matches obtained so far by $\mathbf{G}_1$. Gaussian $\mathbf{G}_3$ gets propagated to the next cluster and spawns Gaussian $\mathbf{G}_4$ that finds all the matches in the cluster and also inherits the matches of $\mathbf{G}_3$ which in turn inherits the matches of $\mathbf{G}_2$ and $\mathbf{G}_1$. Figure 3.7 shows another example where a previously low-weighted hypothesis gains weight and that leads to going back to measuring clusters using that Gaussian.



F_1 F_2 F_3 F_4
F_5 F_6 F_7 F_8 F_9
F_{10} F_{11} F_{12} F_{13} F_{14} F_{15} F_{16}
G_1
G_2
G_3
G_4
G_5
G_6

Figure 3.6: Gaussians in SubAM get propagated amongst clusters by initialising an Active Matching instance with those Gaussians as inputs. As matching progresses, weights of these Gaussians adapt accordingly with new Gaussians spawned when if necessary.

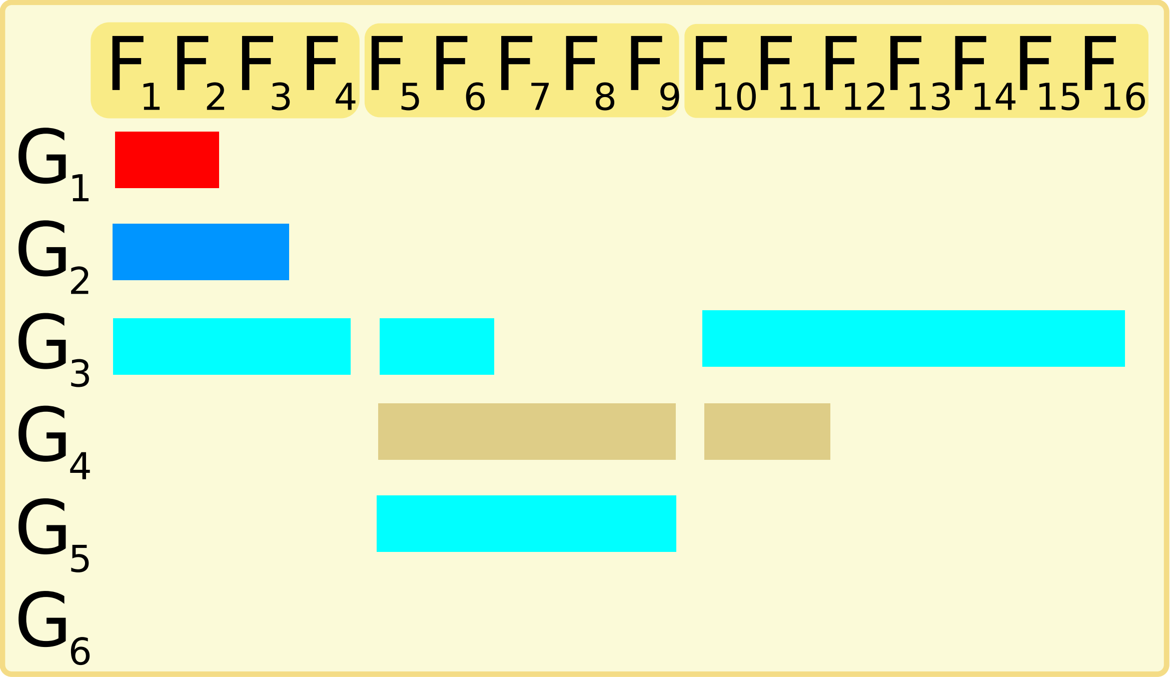
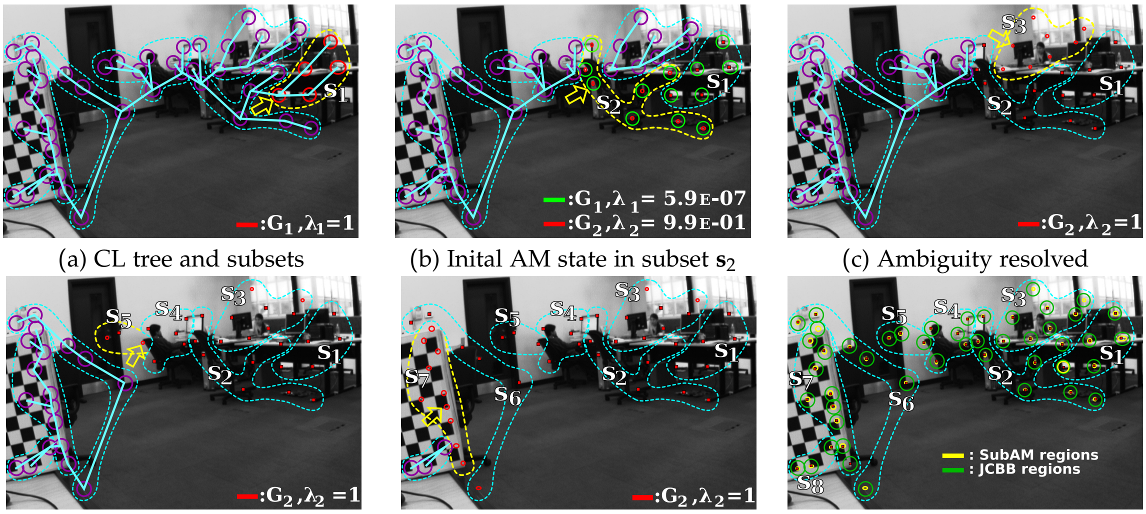

F_1 F_2 F_3 F_4 F_5 F_6 F_7 F_8 F_9 F_{10} F_{11} F_{12} F_{13} F_{14} F_{15} F_{16}
G_1
G_2
G_3
G_4
G_5
G_6

Figure 3.7: An instance when the a weak Gaussian hypothesis gains confidence and leads to propagating the matches backwards in the visited cluster and completes the matching in that cluster.

3.6.2 SubAM: A Step-By-Step Example

Figure 3.8 illustrates a step-by-step example of SubAM in action. The CL tree and the subset structure for this frame is projected in Figure (a). Following the application of AM on subset $\mathbf{s}_1$, the resulting mixture comprises the input to the new AM process to operate on $\mathbf{s}_2$ as demonstrated in (b). By the time $\mathbf{s}_3$ is visited, the mixture contains a single Gaussian projected as small search-regions for the features in $\mathbf{s}_3$ in (c). Since subsets are visited sequentially, their state is updated on demand so any yet-unmeasured subsets retain their original search-state. Finally, in (f) we superimpose the area searched by SubAM, with the area that conventional methods would apply look for matches. It is worth noting that the bigger the subsets, the better the approximation but also the more time AM needs to complete. Moreover, if subsets are very small then it becomes more likely to generate erroneous hypotheses, so one has to select a suitable c to compromise the desirable speed with the quality preserved.



(a) CL tree and subsets (b) Inital AM state in subset $\mathbf{s}_2$ (c) Ambiguity resolved

(d) AM in $\mathbf{s}_4$ (e) AM in $\mathbf{s}_7$ (f) Searched area

Figure 3.8: Matching using SubAM. The prior is projected in (a) together with the CL tree and the partition into subsets. The mixture resulting from AM in $\mathbf{s}_1$ is projected to $\mathbf{s}_2$ in (b) where a new AM process is initialised. In (c) the ambiguity is resolved and AM is attempted in $\mathbf{s}_3$. In (d) and (e) AM is applied to subsequent subsets until all features are matched. In (f) we superimpose the regions searched by SubAM with the initial regions that conventional methods like JCBB need to search.

3.7 Results

To test the capabilities of the CLAM and SubAM algorithms, we have generated a test-bed of matching scenarios spanning different camera dynamics and numbers of features. Since probabilistic filter-based camera trackers such as [Davison et al., 2007] are unsuitable for processing the number of correspondences which we aim at here, we have based our experiments on a new camera tracking system using keyframe optimisation, following very much the design of PTAM [Klein and Murray, 2007]. In all experiments presented we detect FAST features [Rosten and Drummond, 2006] as the only blanket pre-processing, and save the 24×24 surrounding image patches as descriptors. Following the low-cost detection of FAST peaks in a given image (around 2ms for a 640×480 image) we check for template matches of features using ZNCC within the search-regions determined by the matching algorithm. We evaluate the performance of CLAM and SubAM with respect to AM by feeding exactly the same input predictions to all three algorithms.

3.7.1 Obtaining Matching Priors from Optimisation-based Camera Tracking

While matching priors are straightforwardly obtained from the innovation covariance matrix $\mathtt{S}$ calculated at every step in filtering-based camera trackers such as MonoSLAM [Davison et al., 2007], we need to work a little to obtain them from the alternative keyframe optimisation trackers in the style of PTAM [Klein and Murray, 2007] which we use in our experiments.

Such a camera tracker does not store distributions over feature positions due to prohibitive computational cost. However, uncertainty in feature positions has a relatively small effect on matching priors, since it tends to be aligned with the camera's viewing direction in monocular SLAM. Instead, the main uncertainty in image space comes from the unknown motion which is described by a probabilistic motion model with process noise $\mathtt{Q}$. Since the pose of the previous frame is already optimised with respect to the 3D map, we are only interested in the relative uncertainty $\mathtt{P}_{\mathbf{x}_v}^{(\mathrm{rel})}$ between the previous and the current frame: $\mathtt{P}_{\mathbf{x}_v}^{(\mathrm{rel})} = \mathtt{Q}$. Projecting $\mathtt{P}_{\mathbf{x}_v}^{(\mathrm{rel})}$ to the current image, we can compute $\mathtt{S}$:

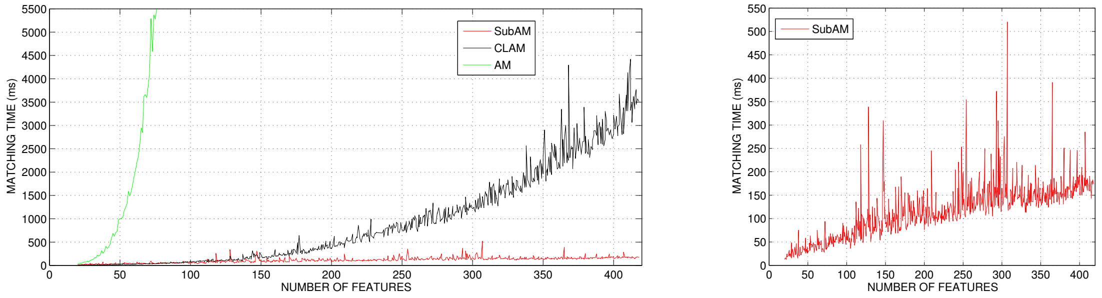
$$\mathtt{S} = \frac{\partial \mathbf{h}(\mathbf{y}_{1:n})}{\partial \mathbf{x}_v} \mathtt{P}_{\mathbf{x}_v}^{(\mathrm{rel})} \frac{\partial \mathbf{h}(\mathbf{y}_{1:n})}{\partial \mathbf{x}_v}^T + \mathtt{R} \,, \tag{3.16}$$

where $\mathbf{h}$ is the projection function of map features $\mathbf{y}_i$, $\mathbf{x}_v$ is the camera pose and $\mathtt{R}$ is a block-diagonal measurement noise matrix. The resulting $\mathtt{S}$ is dense whereas the inter-feature covariances only come from the motion uncertainty.

3.7.2 Time Requirements

Aiming to move towards matching large number of features, we have tested our scalable algorithms with variable number of features. Figure 3.9 illustrates the time required to perform matching using AM, CLAM and SubAM for frames where the number of features are predicted to be visible range from 20 up to 420. As suggested in [Chli and Davison, 2009a], it is evident that AM is not suitable for real-time matching of more than 50 features per frame. CLAM on the other hand exhibits vast reductions in processing time, however its scalability is also affected by the number of matched features — although in a far less severe rate than that of AM (as an indication, AM needs 50 mins for 300 features).

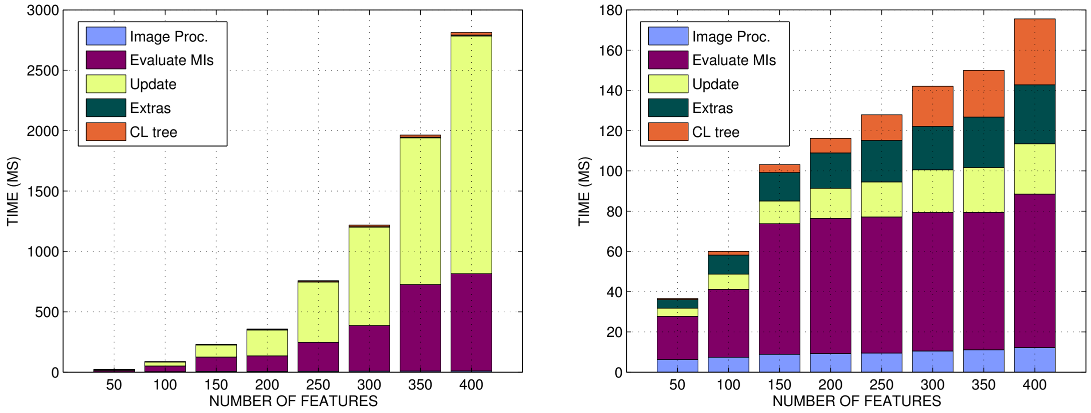
Interestingly, SubAM demonstrates nearly constant runtime across the range of numbers of features achieving matching of 400 features in a record time of 170ms. Up until matching



(a) CLAM vs SubAM *w.r.t.* AM

(b) SubAM close-up

Figure 3.9: Absolute processing time requirements per frame for CLAM and SubAM as a function of features matched per frame. The processing time for standard AM is displayed in (a) for comparison, but its use becomes computationally unfeasible beyond 76 features. While CLAM demonstrates vast a speed improvement over AM, its use is expensive beyond around 200 features. SubAM on the other hand, in (a) and the close-up in (b) exhibits much more scalable performance achieving matching of 420 features in only 170ms.



(a) CLAM timings

(b) SubAM timings

Figure 3.10: Breakdown of the average processing time for CLAM (a) and SubAM (b) with respect to the number of features searched in the individual stages of the algorithms (‘Extras’ includes initialisation of data structures). It is evident that the update of the mixture of Gaussians takes up most of the processing time in CLAM, while the evaluation of MIs is the dominant factor in SubAM. Note that the Image Processing, Extras and CL tree building stages consume comparable time in both methods.

150 features, both AM variants are comparable but as shown in Figure 3.10(a) both the Update and the Evaluation of MIs stages consume increasing processing time in CLAM. As explained earlier in Section 3.5 this is due to the maintenance costs of the tree representation, which on one hand gets partitioned into smaller subtrees we do not explicitly force balanced partitions — this decision is instead driven by MI. The timings breakdown for SubAM in Figure 3.10(b) suggests that the most significant factor then is the Evaluation of MIs. This

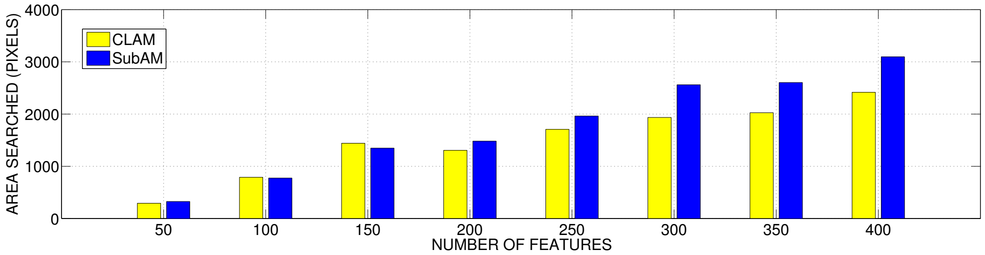


Figure 3.11: The number of pixels searched for ZNCC matches with respect to the input uncertainty regions. While conventional methods like JCBB and RANSAC need to look for matches within the regions corresponding to the input prior, AM and variants exploit correlations of features to reduce this as shown in Figures 3.4(f) and 3.8(f).

is expected as SubAM performs full AM on small subsets of features we have already seen that this is the most expensive step in AM [Chli and Davison, 2009a].

On a small note, the 'rough' nature of the time records in both Figures 3.9 and 3.10 is to be attained to matching scenarios of different levels of difficulty. This is either due to the encounter of incompatible matches or the absence of enough matches to quickly rule out any ambiguity. While the input to all matching algorithms is exactly the same, the order of measurement of features differs which can lead to disparity in timings for the same frame.

3.7.3 Area Searched and Matches Found

Matching a growing number of features per frame with conventional methods, increases the image processing time since more pixels need to be tested for a template match which increases the likelihood of false positives. However, AM exploits the priors in such a way that it reduces the area searched for matches. However, despite CLAM and SubAM being approximations of AM they still reduce the searched areas significantly as shown in Figures 3.4(f) and 3.8(f), which is to be accredited to the use of the CL tree to identify highly correlated links to preserve. Figure 3.11 superimposes the area searched for matches using CLAM and SubAM.

It is worth noting that the matches accepted using AM and both variants are in agreement with the reference result provided by an independent matcher. In some cases, AM rejects some of the matches that CLAM and SubAM seem to accept (comprising no more than 6% of the features matched). This is to be attained to the strict confidence of AM in the linearisation of the distribution of features by a singe Gaussian in the input prior, whereas both CLAM and SubAM relax this distribution allowing some extra (conservative) freedom in the expected configuration of matches.

3.8 Conclusions

We have explored possibilities of conservative approximations to probabilistic priors used in matching, and proposed two fast matching algorithms: CLAM considers the prior distribution as a tree of features while SubAM partitions the matching problem into a tree of subsets of features.

Exploiting the power of probabilistic priors and the insights of Mutual Information to drive decision making, both algorithms have been demonstrated to achieve dramatically low processing time. In fact, our SubAM method is able complete matching of 400 features in 170ms which to our knowledge, is faster than what any other fully probabilistic method has ever achieved. This was an attempt towards making tracking super-fast. However, the kind of motion we want to track remains still a futuristic possibility with such approaches that spend a lot of time on using the prior information. A natural question is that whether these methods can scale better for super-dense online matching. We expect that there is a point where the intermediate updates performed in a top-down approach will come at diminishing returns, which raises the fundamental question of top-down versus bottom-up methods. In the next chapter, we look at other techniques for tracking that can be scaled well on a parallel hardware. Our next chapter debates about feature based and direct parametric based methods and throws insights into why if the aim is to do super-dense matching for 6DoF camera pose estimation, one should move towards dense direct methods.

Chapter 4

Dense Visual Tracking

Contents

4.1 Introduction

Dense image matching forms another associative family of approaches to image registration that use all the potential data available in the image. While feature matching approaches rely on specific image regions based on some saliency measure, dense visual matching on the other hand gleans the necessary information from all the pixels in the image to find the transformation that registers two images. However, the main idea remains the same which is to obtain the transformation by establishing correspondences.

The manner in which this transformation is obtained signifies the major difference between the two approaches. Feature based methods, in most cases, obtain this transformation in the style of a 'match first analyse later' strategy where feature locations are matched first using descriptors and then only the locations of those features are used to obtain the transformation. On the contrary, dense methods either directly obtain the transformation from image gradients using a parametric model or register images in an optic flow style technique. In this framework, the transformation parameters relate the two images in a cost function that encodes the degree of similarity between a pixel and its correspondence in the other image. The alignment starts with an appropriate initial estimate of the transformation and an update is obtained using a gradient descent based technique which is employed on the linearised form of this cost function to trace a point where the gradient no longer changes i.e. an extrema of the cost function. This update is subsumed into the initial alignment transform and the linearise-register-update rule is applied iteratively until some stopping criteria is achieved. The final transformation then represents a convergent set of parameters that best possibly bring the two images in alignment satisfying the constraints set initially in the cost function. Therefore, the two-stage approach which is generally adopted by feature based methods, is combined together to put the search for correspondence task within the loop of finding the global transformation. It is therefore, the paradigm shift in dense matching approaches, 'use all the available data at once', that underscores the strong pull over feature based approaches, and makes these methods quite appealing and attractive.

There have been a multitude of papers that debate these contrasting methods and compare them in an application-specific task to highlight where one betters the other. We appeal to [Irani and Anandan, 1999] and [Triggs et al., 2000] for excellent discussion and their personal opinions on these two approaches to image registration. The starting point of all this debate is that feature based methods depend on some blanket image processing techniques that tend to use arbitrary thresholds to sift points of interest in an image that are considered salient based on their gradient information and the local structures around them. The repeatability of detected features is then affected not just by occlusions and other scene related vagaries but also by these arbitrary thresholds. Detection is followed by augmentation of these points of interest with descriptors that encode the local structure around them to aid matching in the respective images to establish correspondences. The *de facto* two stage procedure of finding a registration transformation treats the matches provided by the first stage as hard constraints and it is the next stage that decides, by using a voting scheme like RANSAC, the ownership of the match to the transformation.

The approach that dense whole image alignment takes is minimising a continuous cost function of intensity differences of pixel locations in an image with warped location in the reference image obtained by applying the transformation that is being sought. The formulation also allows a natural way to weigh the contributions of some pixels over the other by looking at the local gradient information and not ignoring them completely unlike feature based approaches. These weights act as soft constraints and therefore a weighted combination of gradients gives an estimate of the transformation. However, arguments against dense matching greatly profess the inability of the dense approaches to find a global minimum since the matching works on a linearised form of the initial non-convex similarity function. Also, the standard hierarchical coarse to fine framework adopted to further constrain the search space of the parameters can sway the optimisation and direct the solution towards local minima. Moreover, dense approaches, by virtue of the way they work, are naturally ill-suited for wide baseline image matching.

Even with their accompanying disadvantages, in the context of real-time image matching task, the small frame-to-frame distance serves as a really good prior that aids the dense methods. This small frame-to-frame distance naturally supports the linearisation assumptions used in the cost functions and as a result, the solutions obtained are very near to the global minima. In the following, we outline the general benefits that make dense approaches very attractive choice over feature based approaches again within the context of real-time image matching in SLAM type systems which is the focus of this thesis.

No Binary Data Association Most feature matching approaches used in SLAM *e.g.* [Davison, 2003] and [Klein and Murray, 2007] for data association work in a two-stage process where putative matches are found first and then used to obtain the global transformation that relates two images. Relying mostly on the local structure around a feature, the matching operates by searching for a correspondence of each feature independently of the others. Matching decisions on the features correspondences then are only binary: either the match is found or not found. Features for which matches are not found are not used at all to update camera pose. The feature matching also suffers at the hands of fixed thresholds that are used to decide the matching. On the other hand, tracking all-pixel methods are able to weigh contributions of each pixel according to the local structure around them without throwing them based on arbitrary thresholds and hence find out optimal transformation to register the observed image.

Active Search Already Embedded Another difference with dense methods is that feature based approaches are not endowed already with active search embedded in the framework that greatly reduces the correspondence task as the baseline between images decreases. Sparse feature based SLAM systems [Davison, 2003] that keep an uncertainty model on the 3D position of the feature project this uncertainty to find the corresponding search region in the image space where the feature is expected to lie. It is only due to the model that these methods are able to constrain the search space as the frame-rate increases. Dense visual matching approaches on the other hand are very adaptable to changing baselines and naturally support the reduced computational overhead involved with decreasing baseline without having to maintain an explicit uncertainty model. This property is especially very desirable in high frame-rate sequences because there is already a good initial guess available from the previous frame to start the matching that these approaches benefit greatly from.

Graceful Degradation With Deteriorating Visual Conditions Motion blur is one of the main artefacts that arise in images when a camera is undergoing rapid motion. The severity with which the alignment process gets affected is manifested more in feature matching based approaches. The local gradient information gets washed out due to motion blur, leading to the detectors not finding any meaningful points of interest. Dense matching approaches are still able to glean information and show graceful degradation as visual conditions deteriorate. DTAM [Newcombe et al., 2011b] show how tracking that works by matching sparse features as done in [Klein and Murray, 2007] is quite brittle while dense tracking still persists when the camera is undergoing very rapid and shaky motion as highlighted in [Park et al., 2009] and [Park et al., 2012]. This adds a substantial layer of stability and robustness to the system that uses dense matching.

No Absolute Dependence on Texture and Thresholds Dense approaches do not necessarily demand the scene to be well textured which is a prerequisite for any feature based approach. This is again partly due to the fixed thresholds that are used in the detector that fires up the salient feature locations. In fact, when changing frame-rates matching images using sparse features would require feature detection thresholds be altered due to the change in the image brightness levels, while dense matching as such would work without any hassles to change thresholds.

Real-time systems for dense tracking Dense matching approaches come already with their concomitant amenability towards a parallel hardware. Local image gradients directly relate to the transformation being sought and depend only on the local image structure. All the image data can be used instead of discarding majority of pixels as done in sparse methods. Practical systems like [Lovegrove and Davison, 2010] and

[Meilland and Comport, 2012] have shown the parallel implementation of dense tracking.

4.2 Background

The genesis of dense methods is credited to the early works of [Lucas and Kanade, 1981] and [Horn and Schunck, 1981] that stand out for introducing dense matching. Both worked on similar problems in two parallel but different streams to determine correspondences for all the pixels in an image. [Lucas and Kanade, 1981] described a way to globally register two images using a parametric transformation applied to all pixel locations in the image. On the other hand, [Horn and Schunck, 1981] introduced a method to find the correspondence of each pixel, *optical flow*, without imposing any parametric model on the motion or flow of pixels in two images. They both used the concept of sum of squared differences energy minimisation to guide the matching and assumed the Lambertian surface model to derive a constraint that relates image brightness of a pixel in two images. This constraint has been popularly come to be known as *brightness constancy* [Horn and Schunck, 1981] while [Lucas and Kanade, 1981] termed their approach as *method of differences* alluding to the cost function that was being minimised. The least square based energy formulation with gradient descent type optimisations that are common to both, have now become a staple of all the image registration algorithms. The diverse array of places where these formulations have been applied provides for a taxonomy of these image registration algorithms [Bergen et al., 1992]

Fully Parametric where the motion of the pixels can be parametrised using a global motion model that is applied to all the pixels. They include affine or quadratic flow.

Quasi-Parametric They represent the flow field of pixel as a combination of globally parametric component and a local component that varies from pixel to pixel. They include the rigid body motion model where the extrinsic camera parameters form the global parametric component while the depth required to register the images varies with pixel and is the local component.

Non Parametric where the motion cannot be easily parametrised. They include the standard optical flow where each pixel has its own flow vector.

The optimisations used in fully parametric as well as quasi-parametric models reduce to solving "ordinary" least square optimisation while the non-parametric image registrations are as such under-constrained so they require an explicit smoothness term to model the variations of flow vectors in the neighbourhood and are solved using variational methods. On the other hand, fully parametric and non-parametric based registrations usually operate only in image space (*e.g.* 2D template tracking) while the quasi-parametric registration involves a three-dimensional model anchored to image against which the alignment operates *à la* $2\frac{1}{2}$D tracking implemented using Lucas-Kanade *method of differences*.

One of the most interesting properties of fully as well as quasi parametric tracking is that they add the information of the motion model within the framework. We briefly outline the popular tracking methods used in the computer vision community and later turn our attention to 6DoF Rigid body motion model which is the focus of this thesis.

2D Tracking In this type of tracking, the task is to align a template to an image that is undergoing only 2D transformations. This includes the popular work of [Hager and Belhumeur, 1998] who used various motion models to track a template across video frames *e.g.* popular pure translation model [Lucas and Kanade, 1981], affine model [Shi and Tomasi, 1994] and scaled 2D rigid motion model *i.e.* similarity transform. In fact, they also use some standard non-linear models *e.g.* constant acceleration. Tracking in 2D can also benefit computationally. For instance, [Jurie and Dhome, 2001] show how compactly representing change in time derivatives of an image resulting from small changes in the model parameters in a learning stage can result in quick look-ups for updates while doing online tracking.

3D Tracking This includes tracking using constraints derived from the geometry of the object or surface in three-dimensional space, that is to be tracked over a sequence of video frames. Tracking using Active appearance models [Cootes et al., 2001], 3D parametric articulated models with more than six degrees of freedom [Lowe, 1991] and later [Bregler and Malik, 1998], CAD models based *àla* ACRONYM [Brooks, 1981], HYPER [Ayache and Faugeras, 1986], SCERPO [Lowe, 1987] and RAPiD [Harris and Stennet, 1990], [Harris, 1992], and models using dense point surfaces [Newcombe et al., 2011b] in the style of Lucas-Kanade $2\frac{1}{2}$D [Baker et al., 2004b] is considered to be 3D tracking as all these methods add the information pertaining to the 3D existence of the object into the registration.

There has also been work on over-parametrising optical flow, notably the work of [Nir et al., 2008], where each pixel has more than just two parameters in the conventional form and there is still smoothness that operates on gradients of motion parameters. Though such kind of over-parametrised registration methods do perform better than the non-parametric methods but they still solve the problem in image domain and do not impose a global model on all the pixels. In the context of sequential camera tracking where a three-dimensional surface is registered against an incoming image, registration in the style of quasi-parametric 3D-2D image association forms the focus of attention in this thesis. The following section traces the roots of various formulations and approaches to track a moving camera undergoing a rigid body motion observing a three-dimensional scene that cannot necessarily be easily parametrised.

4.2.1 Rigid Body Motion

Given a three-dimensional model of the scene, it is possible to obtain the motion field that relates the motion of a scene point in the image as a function of its position and the translatory and rotary motion the camera is undergoing. This was first shown in [Higgins and Prazdny, 1980]. Considerable attention has been devoted to the problem of recovering camera motion parameters using image data from the point of view of tasks that involve robot navigation. [Bruss and Horn, 1983] provided a way to recover this by minimising the difference between the optical flow and the motion field predicted in the image. However, their method split the task into two steps: computing optic flow first and using that to obtain the parameters. No experimental results were given and only theoretical insights were presented. Computing the motion parameters using an optical flow pre-processing stage has motivated many other researchers too *e.g.* [Adiv, 1985], [Heeger and Jepson, 1992] and [Negahdaripour and Lee, 1992].

It is worth mentioning that attempts to use optical flow as a front end to the rigid body motion estimation process [Bruss and Horn, 1983] have not been greeted well in the community. It has been long argued that using optical flow as the first stage for motion estimation is no different to using another blanket image processing technique similar to using feature extraction matching. The results are heavily dependent on the first stage and moreover, computing optical flow is already an ill-posed problem that yields often inaccurate flow vectors at the image boundaries and depth discontinuities. Translation estimates are severely affected since they rely on the motion parallax at the depth discontinuities more than rotation. In their retrospective view on optical flow [Horn and Schunck, 1993] also point out how using optical flow makes the problem of obtaining camera parameters even more difficult. Rigid body constraints the motion field of all the scene points in only six parameters while optical flow has twice the number of parameters as the size of image. Therefore, they professed the superiority of using rigid body constraints directly within the image intensity based error function.

Later [Lucas and Kanade, 1985] used their original *method of differences* to obtain general camera motion assuming the depth was already given (*e.g.* from stereo). They showed real experiments on the Stanford cart [Moravec, 1980a, Moravec, 1980b]. Their method that was not fully dense and that few selected points were used and correspondences were obtained by hand nonetheless the optimisation was still carried out in an iterative dense matching style in image instead of using pixel locations to obtain the parameters as done in [Moravec, 1980a]. Interestingly, in the same conference [Negahdaripour and Horn, 1985a, Negahdaripour and Horn, 1985b] also gave a closed form solution to recovering rigid body motion parameters as well as 3D plane parameters for a moving camera observing a planar scene, directly from image brightness gradients. In contrast to Lucas's method, they demonstrated the applicability of their method using only a synthetic experiment. However, they did not require any selected pixels and instead used the whole image.

[Horn and Weldon, 1988] present a comprehensive analysis of direct approach to recovering camera motion parameters when it is undergoing pure rotation, pure translation or when the rotation is known. [Negahdaripour and Horn, 1989] propose a direct method to recover the focus of expansion (FoE) using a positiveness of depth constraint within the formulation. They present insights into the uniqueness of the FoE and show how stationary points (the points corresponding to zero temporal gradient) can be used to obtain the translation. [Kumar and Hanson, 1989] show how estimating the rotation and translation simultaneously performs better than splitting the task of estimating each one independently when the data is noisy and corrupted with outliers. [Taalebinezhaad, 1992b, Taalebinezhaad, 1992a] recover the rotation and *fixation velocity* in the image induced by purely rotating camera motion using the standard brightness constancy constraint.

A different approach is also proposed by [Tomasi and Shi, 1993] to recover the translation of a moving camera that introduces motion parallax. The rate of change of angle subtended by two rays emanating from two image points as the camera moves gives a bilinear constraint on the translation and depth of the points. All point pairs give such bilinear constraints that can be used to solve for translation and depth of these points using a variable projection method.

[Hanna, 1991] use a multi-resolution framework in a direct estimation of camera motion parameters with a planar scene assumption by iteratively alternating between ego-motion and scene structure. A taxonomy of various different motion models used in image registration is summarised in [Bergen et al., 1992] where they also use the standard coarse-to-fine strategy to obtain the parameterised motion of pixels. This multi-resolution framework embedded in the direct estimation of motion has become a well known concept from then onwards. [Irani et al., 1994] provide a solution to the same problem by first registering the images based on a 2D parametric motion assumption that cancels the rotational component. The resulting registered images have only 3D translation which is solved by finding the FoE. They use optical flow to find the FoE and state that the overdetermined solution space enables robust computation of FoE even when the flow is inaccurate at the motion boundaries. The 3D rotation components of the motion can then be obtained once the translation is known. However, they do not recover the structure information.

Multiview approaches later found a huge surge in applications that also required to reconstruct the scene. In particular, mosaicing and building panoramas from purely rotating cameras or observing planar scenes were very popular since they did not require a full 3D reconstruction and registration could be done using a rigid body assumption. Such multiview approaches are able to extract information optimally from all the images and give a high resolution representation of the scene.

The plane + parallax [1] approach [Sawhney, 1994], gained popularity as the registration of images under a dominant planar motion assumption also offered a means to recover the 3D structure i.e. the depth of the points with respect to the planar surface. The two view approach of [Kumar et al., 1994] used to recover heights of objects on ground using aerial images found extensions later in [Irani et al., 1999] and [Irani et al., 2000] for multiple views for resolving ambiguities in structure estimation and improved signal-to-noise ratio performance all using direct methods.

[Kumar et al., 1995] review the mosaic reconstructions that use direct image matching and later [Irani et al., 1996] provide an excellent taxonomy of different mosaics that can be created using images obtained from a video sequence. They term *static mosaic* as the ef-

[1] A more clear visual exposition of the concept is detailed in [Irani et al., 1998]. Image motion of points that lie on a planar surface can be trivially explained by plane-induced homography while the motion of any 3D point not on a plane can be decomposed into two components: the homography induced and the residual motion which is called parallax due to plane and hence the name of the approach.

ficient representation of a scene background without moving objects. The *dynamic mosaic* corresponds to the representation of mosaic that is dynamically updated based on the current view and only shows the part of mosaic that is within the vicinity of the current view. The mosaic reconstructions precede with image alignment using a 2D parametric quadratic flow or 3D plane parallax representation. The plane parallax approach is able to rectify the mosaic reconstruction of the parts of the scene that do not belong to planar surface. They also show various applications of mosaics specially in visual enhancement, video compression and video indexing. [Irani and Anandan, 1998] detect moving objects as a byproduct of plane parallax alignment based image registration. Moving objects are visibly highlighted as outliers that do not obey the transformation.

[Sawhney et al., 1995] provided a catalogue of 2D and 3D motion model based registration techniques in the context of efficient visual representations of the scene via mosaicing. They showed a robust estimator driven outlier rejection approach to discard regions containing moving objects that do not obey the dominant motion assumption. [Szeliski and Kang, 1995] propose a direct method for homography based image registration to create mosaics of whiteboards, desktops or other planar surfaces and recovering projective depth. Later, [Shum and Szeliski, 1997, Szeliski and Shum, 1997, Shum and Szeliski, 1998] show building panoramas from a rotating camera by using patch based alignment as an efficient substitute to aligning images using only pixels. It works by defining the motion of patch locally and then using patches instead of pixels in the full registration framework. They also propose ways to reduce the effects of registration using block adjustment and local refinement for visually appealing panoramic reconstructions.

Direct estimation process has also been embedded in Kalman Filter based recursive estimation of structure and motion [Matthies et al., 1989] and [Heel, 1990]. Later [Dellaert et al., 1998] and [Dellaert and Collins, 1999] also show the direct estimation process coupled with planar scene reconstruction in Kalman Filter based recursive estimation framework. [Jin et al., 2003] also register sparse patches in 3D against an incoming image using a direct approach within Kalman filter recursive structure and motion estimation.

Lately, [Horn et al., 2007] show the benefits of using direct methods to reliably estimate the time to contact for a vehicle relative to a planar surface. [Dame and Marchand, 2010] and [Panin and Knoll, 2008] use MI information instead of pixel intensity difference in the framework to do tracking.

In a concurrent stream, there has been a confluence of methods that use a CAD model to obtain the rigid body motion of the camera. The landmark paper from [Harris, 1992] was one of the first marker-less 3D model based real-time rigid body camera tracking system called RAPiD. It used the edge information to register CAD model of a 3D object to an observed image. Searching in one dimension along the normal direction to the control points sampled on the predicted model edge it finds out the closest edge in the image that brings the model in alignment to the observed image projection of the object. Though such methods do not necessarily use the whole image but the idea of obtaining the motion parameters through a least square optimisation using a small motion assumption are in the same vein. The simplicity and the efficiency of this method has given birth to many further extensions. In particular, [Lowe, 1991] propose a least square method of fitting parametrised three-dimensional models with more than standard six degrees of freedom to images and use a stabilisation technique (a form of regularisation prior) to obtain the parameters even when the problem is under-constrained. Also, [Drummond et al., 2002] track a wire-frame CAD model of the object using a robust least square approach in a RAPID style matching framework with binary space partitioning (BSP) tree based visibility reasoning. A practical real-time (25Hz) system is implemented which uses the tracker to servo a robot to a particular position. They also present an extension to multi-camera and multi-target tracking scenarios.

4.2.2 Summary

In 1980s the rigid body tracking research revolved around theoretical investigation of the problem and bringing forward the concept of direct tracking. In early 90s direct tracking using a multi-resolution approach was a well established concept to do image matching. There were also investigations of using robust penalty functions to preclude the contributions of outliers in the framework. Most of the direct approaches began to find their applications in mosaicing and building panoramas and object segmentation where the structure was either not required or planar though it was clear that a 3D model would greatly simplify and improve tracking. The idea of tracking from a general 3D surface was still in its nascent stages even though 3D reconstruction from multiple views was beginning to mature. In the parallel stream, there was a great activity in tracking camera poses using simplified 3D surfaces in the form of CAD models. It was a great success and provided a platform to robust model based tracking. The simplicity, fidelity and robustness of the method was received with aplomb. Many further extensions led to the application of the idea to different streams mainly in augmented reality and visual servoing. The parallel track on 3D reconstruction advanced enormously and boasted big numbers in the scale of reconstruction and lately with variational methods and GPU on the modern hardware it is possible to have the tracking from a general 3D model that is being built online than just using a simplified CAD model.

4.3 Camera Motion Parametrisation

Much of the previous research has only focussed on estimating the rigid body parameters with little attention to the actual parametrisation used to represent rotation and translation. They are either represented independently or instead the rotational and translation velocity components are estimated which do not suffer from the singularities introduced by notations. Although [Taylor and Kriegman, 1994] had used SO(3) based minimisation to obtain iterative rotational updates, [Bregler and Malik, 1998] were the first to introduce twists and exponential maps for tracking articulated human body pose under orthographic projection. Later, [Drummond and Cipolla, 1999b] show the use of Lie Algebra for direct visual tracking to control the movements of a robotic arm. The nice properties of Lie Algebra make them an obvious choice of representation in our tracking. We summarise few of them below:

- SE3 provides a compact representation of rotation and translation in only 6 parameters compared to representing them directly with matrices. There is very simple mapping from these 6 parameters onto a lie-manifold similar to a general point-curve locus relationship in euclidean space.

- Rotation and translation can be represented together in one matrix instead of representing them independently and this helps when linearsing them around a given point.

- The Adjoint of the Lie group can be used to represent the inverse of the transformation very easily.

Such Lie Algebra based parameterisation for motion has become a commonplace in representing the rigid body rotation and translation. [Klein and Drummond, 2003], [Bayro-Corrochano and Ortegon-Aguilar, 2004], [Klein and Murray, 2007] and the recent work of [Lovegrove and Davison, 2010], [Newcombe et al., 2011b] and [Meilland and Comport, 2012] all choose lie algebra parameterisation.

4.4 Direct Parametric Tracking

An image coordinate is represented by a homogenised vector x, the location of the homogenised scene point, $\dot{\mathrm{x}}$, pierced at the image plane by the ray joining camera optic center and the point. This scene point is represented in camera coordinate system and its depth from the image plane is abbreviated by d. The camera internal parameters are described by the K matrix where as π is the projection operator to de-homogenise a vector.

$$\mathrm{x} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}, \quad \dot{\mathrm{x}} = \begin{pmatrix} d\mathrm{K}^{-1}\mathrm{x} \\ 1 \end{pmatrix}, \tag{4.1}$$

$$\mathrm{K} = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \pi\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} \frac{u}{w} \\ \frac{v}{w} \end{pmatrix}. \tag{4.2}$$

The matrix T_{lr} represents the transformation between the camera reference frame and the live frame, where the subscript "lr" is read as reference to live. Under the assumption that the three-dimensional surface being viewed in the image has Lambertian properties, the standard *brightness constancy* constraint then relates the image intensities of the same scene point when observed from two different views. The photometric cost function is defined as the sum of the intensity differences between reference image and live image aligned to the reference image coordinate system.

4.4.1 Pure Rotation Tracking: SO(3) Tracker

$$\psi \;=\; \sum_{\mathrm{x}\in\mathrm{I}} \Big(\mathrm{I}_r(\mathrm{x}) - \mathrm{I}_l(\pi(\mathrm{K}\mathrm{R}_{lr}\dot{\mathrm{x}}))\Big)^2 . \tag{4.3}$$

A general **SE**(3) tracker optimises jointly over translation and rotation. The mathematical optimisation framework is explained below for **SE**(3) tracker however, similar optimisation scheme can be derived for pure rotation tracker with transformation replaced by rotation.

4.4.2 Translation and Rotation Tracking: SE(3) Tracker

$$\psi \;=\; \sum_{\mathrm{x}\in\mathrm{I}} \Big(\mathrm{I}_r(\mathrm{x}) - \mathrm{I}_l(\pi(\mathrm{K}\mathrm{T}_{lr}\dot{\mathrm{x}}))\Big)^2 . \tag{4.4}$$

T_{lr} is the transformation that is being sought via minimisation of this cost function. Given the high non-linearity associated with the cost function, it is linearised around an already available estimate $\hat{\mathrm{T}}_{lr}$. Using the lie algebra properties the transformation can be rewritten as

$$\mathrm{T}_{lr} = \hat{\mathrm{T}}_{lr} \exp(\delta \mathrm{u}) \ . \tag{4.5}$$

As a result, the cost function becomes parametrised by this small update δu

$$\psi(\delta \mathrm{u}) \ = \ \sum_{\mathrm{x} \in \mathrm{I}} \left(\mathrm{I}_r(\mathrm{x}) - \mathrm{I}_l(\pi(\mathrm{K}\hat{\mathrm{T}}_{lr} \exp(\delta \mathrm{u})\dot{\mathrm{x}})) \right)^2 , \tag{4.6}$$

Using the Taylor Series, the live image is linearised around the current estimate

$$\psi \ = \ \sum_{\mathrm{x} \in \mathrm{I}} \left(\mathrm{I}_r(\mathrm{x}) - (\mathrm{I}_l(\pi(\mathrm{K}\hat{\mathrm{T}}_{lr} \exp(0)\dot{\mathrm{x}})) + \frac{\partial \mathrm{I}_l(\pi(\mathrm{K}\hat{\mathrm{T}}_{lr} \exp(\delta \mathrm{u})\dot{\mathrm{x}}))}{\partial(\delta \mathrm{u})}|_{\delta \mathrm{u}=0} \delta \mathrm{u}) \right)^2 . \tag{4.7}$$

Denoting $\mathrm{e_x}$ as the residual obtained from the previous estimate and $\mathrm{J_x}$ as the Jacobian resulting from the Taylor series expansion, the above expression can be simplified as

$$\begin{aligned} \mathrm{e_x} \ &= \ \mathrm{I}_r(\mathrm{x}) - \mathrm{I}_l(\pi(\mathrm{K}\hat{\mathrm{T}}_{lr} \exp(0)\dot{\mathrm{x}}) && (4.8) \\ \mathrm{J_x} \ &= \ \frac{\partial \mathrm{I}_l(\pi(\mathrm{K}\hat{\mathrm{T}}_{lr} \exp(\delta \mathrm{u})\dot{\mathrm{x}}))}{\partial(\delta \mathrm{u})}|_{\delta \mathrm{u}=0} && (4.9) \\ \implies \psi \ &= \ \sum_{\mathrm{x} \in \mathrm{I}} (\mathrm{e_x} - \mathrm{J_x}\delta \mathrm{u})^2 \ . && (4.10) \end{aligned}$$

The $\mathrm{J_x}$ is a 1 × 3 vector in case of pure rotation and 1 × 6 for full 6DoF parameterisation. The δu is 3 × 1 and 6 × 1 vector respectively. The Jacobian is computed as

$$\begin{aligned} \mathrm{J_x} \ &= \ \frac{\partial \mathrm{I}_l(\pi(\mathrm{K}\hat{\mathrm{T}}_{lr} \exp(\delta \mathrm{u})\dot{\mathrm{x}}))}{\partial(\delta \mathrm{u})}|_{\delta \mathrm{u}=0} && (4.11) \\ \mathrm{J_x} \ &= \ \frac{\partial \mathrm{I}_l(\pi(\mathrm{K}\hat{\mathrm{T}}_{lr} \exp(\delta \mathrm{u})\dot{\mathrm{x}}))}{\partial(\pi(\mathrm{K}\hat{\mathrm{T}}_{lr} \exp(\delta \mathrm{u})\dot{\mathrm{x}}))}|_{\delta \mathrm{u}=0} \times \frac{\partial \pi(\mathrm{K}\hat{\mathrm{T}}_{lr} \exp(\delta \mathrm{u})\dot{\mathrm{x}})}{\partial(\delta \mathrm{u})}|_{\delta \mathrm{u}=0} \ . && (4.12) \end{aligned}$$

Using the chain-rule, this is further expanded to

$$\mathrm{J_x} = \left(\frac{\partial \mathrm{I}_l(\pi(\mathrm{K}\hat{\mathrm{T}}_{lr} \exp(\delta \mathrm{u})\dot{\mathrm{x}}))}{\partial \pi(\mathrm{K}\hat{\mathrm{T}}_{lr} \exp(\delta \mathrm{u})\dot{\mathrm{x}})} \times \frac{\partial \pi(\mathrm{K}\hat{\mathrm{T}}_{lr} \exp(\delta \mathrm{u})\dot{\mathrm{x}})}{\partial \mathrm{K}\hat{\mathrm{T}}_{lr} \exp(\delta \mathrm{u})\dot{\mathrm{x}}} \times \frac{\partial \mathrm{K}\hat{\mathrm{T}}_{lr} \exp(\delta \mathrm{u})\dot{\mathrm{x}}}{\partial(\delta \mathrm{u})} \right)|_{\delta \mathrm{u}=0} \ . \tag{4.13}$$

The first derivative in the chain is the partial derivative of the warped image with respect to warped locations and is a 1 × 2 matrix. Elements of this matrix are obtained via a finite difference stencil (preferably a higher order than the standard 2-point stencil). The next term is the derivative of projection operator with respect to its parameters and in general form is a 2 × 3 matrix written as:

$$\frac{\partial\pi\begin{pmatrix}u\\v\\w\end{pmatrix}}{\partial\begin{pmatrix}u\\v\\w\end{pmatrix}}=\frac{\partial\begin{pmatrix}\frac{u}{w}\\\frac{v}{w}\end{pmatrix}}{\partial\begin{pmatrix}u\\v\\w\end{pmatrix}}=\begin{pmatrix}\frac{1}{w}&0&-\frac{u}{w^2}\\0&\frac{1}{w}&-\frac{v}{w^2}\end{pmatrix}. \tag{4.14}$$

The last derivative in the chain can be obtained as

$$\frac{\partial\mathrm{K}\hat{\mathrm{T}}_{lr}\exp(\delta\mathrm{u})\dot{\mathrm{x}}}{\partial(\delta\mathrm{u})}|_{\delta\mathrm{u}=0}=\mathrm{K}\hat{\mathrm{T}}_{lr}\left(\frac{\partial\exp(\delta\mathrm{u})}{\partial(\delta\mathrm{u})}|_{\delta\mathrm{u}=0}\right)\dot{\mathrm{x}}\,. \tag{4.15}$$

The expression $\frac{\partial\exp(\delta\mathrm{u})}{\partial(\delta\mathrm{u})}|_{\delta\mathrm{u}=0}$ yields a generator of the Lie algebra corresponding to the parameter. As a result the expression expands to a 3 × 6 matrix, i.e.

$$\mathrm{K}\hat{\mathrm{T}}_{lr}\frac{\partial\exp(\delta\mathrm{u})}{\partial(\delta\mathrm{u})}|_{\delta\mathrm{u}=0}\dot{\mathrm{x}}=\mathrm{K}\hat{\mathrm{T}}_{lr}\begin{pmatrix}\mathrm{G}_{\mathrm{t_x}}&\mathrm{G}_{\mathrm{t_y}}&\mathrm{G}_{\mathrm{t_z}}&\mathrm{G}_{\theta_\mathrm{x}}&\mathrm{G}_{\theta_\mathrm{y}}&\mathrm{G}_{\theta_\mathrm{z}}\end{pmatrix}\dot{\mathrm{x}}\,. \tag{4.16}$$

The Generators when multiplied with $\dot{\mathrm{x}}$ produce the following 4 × 6 matrix:

$$\begin{pmatrix}1&0&0&0&z&-y\\0&1&0&-z&0&x\\0&0&1&y&-x&0\\0&0&0&0&0&0\end{pmatrix}. \tag{4.17}$$

If $\delta\mathrm{u}=(\mathrm{t_x},\mathrm{t_y},\mathrm{t_z},\omega_\mathrm{x},\omega_\mathrm{y},\omega_\mathrm{z})^T$ represents the 6 parameters over which the optimisations runs, then $\mathrm{J_x}$ yields a element-wise derivative with respect these 6 parameters,

$$\mathrm{J_x}=\begin{pmatrix}\mathrm{J}_{\mathrm{t_x}}&\mathrm{J}_{\mathrm{t_y}}&\mathrm{J}_{\mathrm{t_z}}&\mathrm{J}_{\omega_\mathrm{x}}&\mathrm{J}_{\omega_\mathrm{y}}&\mathrm{J}_{\omega_\mathrm{z}}\end{pmatrix}$$

. The cost function is then minimised with respect to $\delta\mathrm{u}$ leading to solutions to the "ordinary" least squares problem, called the normal equations:

$$\begin{aligned}\frac{\partial\psi}{\partial(\delta\mathrm{u})}&=\sum_{\mathrm{x}\in\mathrm{I}}-2\mathrm{J_x}^T(\mathrm{e_x}-\mathrm{J_x}\delta\mathrm{u})=0 &(4.18)\\\left(\sum_{\mathrm{x}\in\mathrm{I}}\mathrm{J_x}^T\mathrm{J_x}\right)\delta\mathrm{u}&=\sum_{\mathrm{x}\in\mathrm{I}}\mathrm{J_x}^T\mathrm{e_x} &(4.19)\\\delta\mathrm{u}&=\left(\sum_{\mathrm{x}\in\mathrm{I}}\mathrm{J_x}^T\mathrm{J_x}\right)^{-1}\sum_{\mathrm{x}\in\mathrm{I}}\mathrm{J_x}^T\mathrm{e_x}\,. &(4.20)\end{aligned}$$

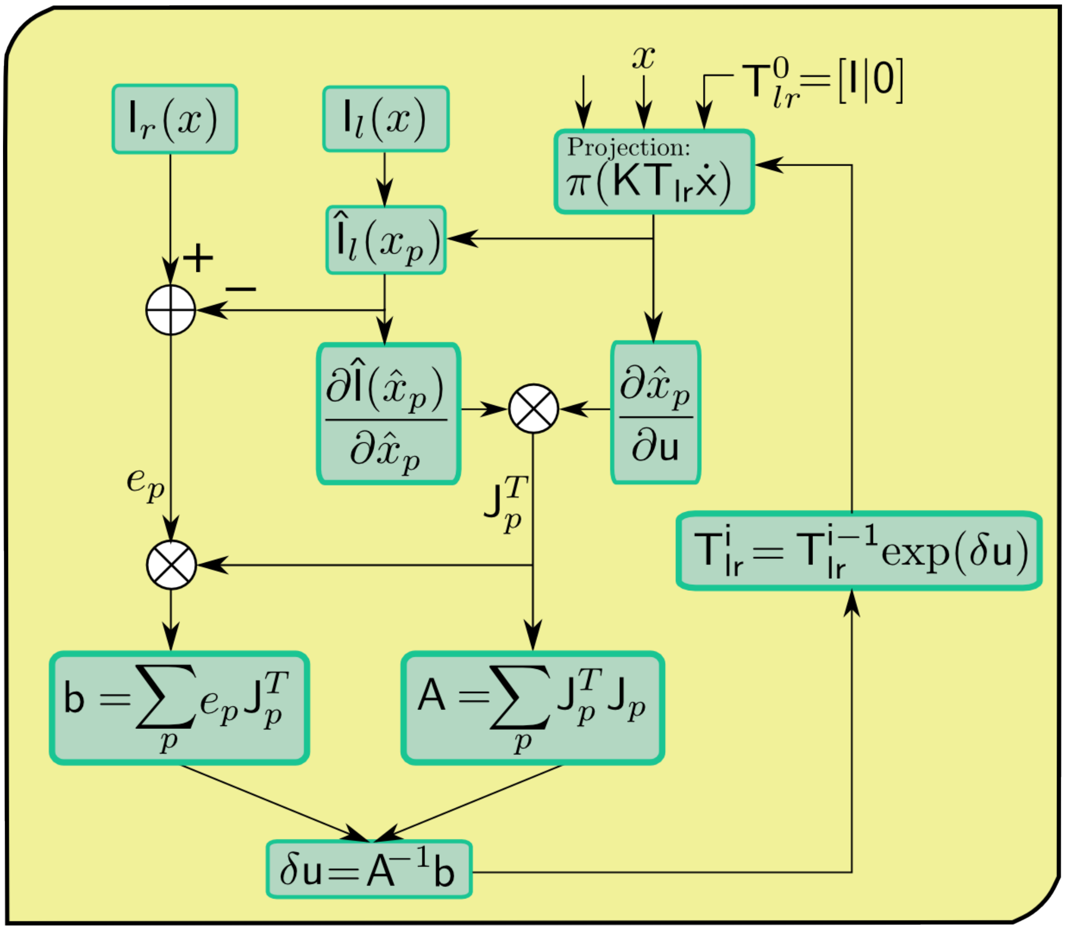


Figure 4.1: Flow chart of the algorithmic components of direct image alignment tracking.

4.5 Robust Cost Functions: M-Estimators

The cost function employed in the formulation implicitly already carries the notion of inlier-outlier model present in the data *e.g.* using a Gaussian model means that data sample lying outside the 3σ is likely to be an outlier. The impact of an outlier sample in the optimisation then is greatly determined by how forgiving is the cost function to the outlier; this is termed as *robustness* of a cost function. It is measured in terms of *breaking point* (it is the % of bad samples that break the model underlying the cost function being used) and by the *influence function* [Hampel et al., 1986] (how much weight does the cost function give to the sample) of a distribution.

The standard quadratic cost function used most often assumes an additive Gaussian noise distribution. The special properties to do with linearity and the ease in manipulating them in optimisations make Gaussian distributions very much welcomed in optimisation theory. However, a Gaussian distribution in its pure form rarely fits well with real data and hence it is not very robust to outliers; the cost function grows without bound as the error increases and the associated weighting function in the minimisation assigns same weights to all residuals irrespective of whether the sample is an outlier or inlier to the distribution. Therefore, any inclusions of outliers in the estimation can greatly hamper the minimisation scheme. This is more evident when occlusions, specular reflections and other distractions (*e.g.* non-rigid objects or repetitive scene structures) are present in the scene that tend to misguide the optimisation. What is needed is an implicit way to discard outliers in a manner similar to how humans are able to sift the inliers by just looking at data. A cost function that increases less rapidly compared to the quadratic function and that the associated weighing function gives less or zero weight to outliers that violate the Gaussian assumption, has the desirable property of discarding outliers. How exactly then should a cost function that is more immune to the outliers present in the data be chosen? One way would be to plot the statistics of residuals similar to [Huang and Mumford, 1999], from imagery taken with a real camera. However, this is only ever possible when we know the alignment transformation and even then the residual statistics vary a lot with scenes.

Robust statistics and outlier rejection offer a plethora of choices of cost functions that are resilient to outliers. Among the first to introduce statistical methods in the least square formulations in computer vision were [Gruen, 1985], [Förstner, 1987], [Black and Anandan, 1991], and later, [Black, 1992], [Black and Anandan, 1993] and [Black and Rangarajan, 1996] pioneered the use of robust cost functions to estimate optical flow at the image boundaries. [Stewart, 1999] provide an excellent compendium of various cost functions that are used in computer vision for parameter estimation. [Fitzgibbon, 2001] also use similar cost functions to robustly register point sets in an iterative closest point (ICP) framework.

Akin to the Gaussian distribution standard deviation that decides the spread of the function, an equivalent associated scale factor determines the cut-off point beyond which the contribution of a data sample begins to diminish. By tuning this scale factor, one is able to control the effect of a data sample into the minimisation. The standard least square optimisation is then turned into a weighted least square optimisation where weights are obtained from the distribution that is being used to model the data. Denoting $\mathrm{r_x}$ as the residual at pixel location x and ρ is any robust cost function being used to model the distribution of the residuals.

$$\begin{aligned}
\mathrm{r_x} &= \mathrm{I}_r(\mathrm{x}) - \mathrm{I}_l(\pi(\mathrm{KT}_{lr}\dot{\mathrm{x}})) && (4.21)\\
\psi &= \sum_{x\in \mathrm{I}} \rho(\mathrm{r_x}) && (4.22)\\
\frac{\partial\psi}{\partial(\delta\mathrm{u})} &= \sum_{x\in \mathrm{I}} \rho'(\mathrm{r_x})\frac{\partial \mathrm{r_x}}{\partial(\delta\mathrm{u})} && (4.23)\\
\frac{\partial\psi}{\partial(\delta\mathrm{u})} &= \sum_{x\in \mathrm{I}} \frac{\rho'(\mathrm{r_x})}{\mathrm{r_x}}\mathrm{r_x}\frac{\partial \mathrm{r_x}}{\partial(\delta\mathrm{u})} && (4.24)\\
\frac{\partial\psi}{\partial(\delta\mathrm{u})} &= \sum_{x\in \mathrm{I}} w(\mathrm{r_x})\mathrm{r_x}\frac{\partial \mathrm{r_x}}{\partial(\delta\mathrm{u})} && (4.25)\\
\implies \psi &= \sum_{x\in \mathrm{I}} w(\mathrm{r_x})\mathrm{r_x^2}\,. && (4.26)
\end{aligned}$$

Therefore the update is obtained from this weighted least square function as

$$\begin{aligned}
\frac{\partial\psi}{\partial(\delta\mathrm{u})} &= \sum_{x\in \mathrm{I}} \frac{\partial}{\partial(\delta\mathrm{u})} w(\mathrm{r_x})(\mathrm{e_x} - \mathrm{J_x}\delta\mathrm{u})^2 = 0 && (4.27)\\
\implies \delta\mathrm{u} &= \left(\sum_{\mathrm{x}\in \mathrm{I}} w(\mathrm{r_x})\mathrm{J_x}^T\mathrm{J_x}\right)^{-1} \sum_{\mathrm{x}\in \mathrm{I}}\left(w(\mathrm{r_x})\mathrm{J_x}^T\mathrm{e_x}\right)\,. && (4.28)
\end{aligned}$$

Below we catalogue some of the robust cost functions that have been used primarily in computer vision:

Quadratic Function

$$\rho(x) = x^2 \tag{4.29}$$

Truncated Quadratic

$$\rho(x,\sigma) = \begin{cases} x^2 & |x| \le \sigma \\ \sigma^2 & \text{otherwise} \end{cases} \tag{4.30}$$

Huber

$$\rho(x,\sigma) = \begin{cases} \frac{x^2}{2} & |x| \le \sigma \\ \frac{1}{2}\sigma(2|x| - \sigma) & \text{otherwise} \end{cases} \tag{4.31}$$

Geman and McClure

$$\rho(x,\sigma) = \frac{x^2}{\sigma^2 + x^2} \tag{4.32}$$

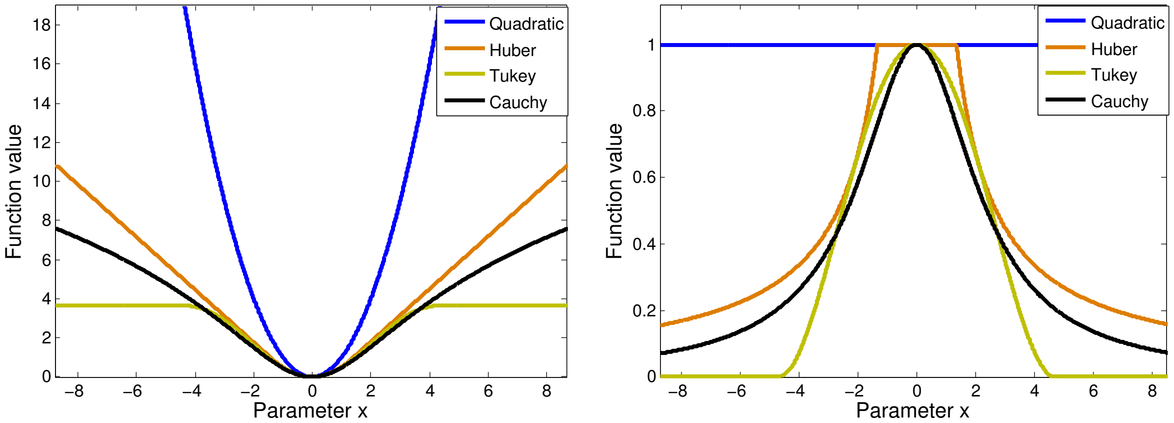


Figure 4.2: Different Robust M-Estimators and their corresponding influence functions shown side by side. All the functions are plotted with their scale parameters tuned to 95% statistical efficiency of standard normal Gaussian distribution [Zhang, 1997]. It is worth noting that Tukey function gives zero weight to any data sample that lies outside the range set by the scale-parameter and is able to reject completely the contribution of that sample in comparison to other robust cost functions that assign low weight.

Lorentzian/Cauchy

$$\rho(x,\sigma) = \frac{\sigma^2}{2}\log\left(1+\frac{x^2}{\sigma^2}\right) \tag{4.33}$$

Tukey Biweight Function

$$\rho(x,\sigma) = \begin{cases} \frac{\sigma^2}{6}\left(1-\left(1-\left(\frac{x}{\sigma}\right)^2\right)^3\right) & |x| \leq \sigma \\ \frac{\sigma^2}{6} & \text{otherwise} \end{cases} \tag{4.34}$$

Each of these distributions has a scalar parameter that is tunable to change the spread of the distribution. An estimate of this value obtained from data is more desirable than setting an absolute value. Therefore, a lot of attention has been devoted to finding the optimal scale parameter. [Sawhney et al., 1995] model the residuals using a contaminated Gaussian distribution and obtain a robust estimate of the standard deviation, σ of the distribution using the residual data as:

$$\sigma = 1.4286\mathsf{median}_i|r_i| \ .$$

They discard the residuals from the optimisation that exceed the threshold which is empirically set to be 2.5σ. Later, [Black et al., 1998] discuss various robust cost functions and draw connections of these robust functions with anisotropic diffusion. They use a similar robust

σ estimation but from median of absolute deviations (MAD) [Rousseeuw and Leroy, 1987] (note that there is no $|.|$ in the inner median) as:

$$\sigma_e = 1.4286 \mathsf{median}_i |r_i - \mathsf{median}_i r_i| \ .$$

They obtain the appropriate scale parameter for any distribution by finding a point at which the derivative of the function vanishes and setting that to σ_e to obtain the respective value of the scale parameter. This is a point beyond which the influence of the outliers begins to decrease. For instance, the Lorentzian function yields $x = \sqrt{2}\sigma$ as the point at which the derivative of the function becomes zero and as result the σ for a Lorentzian function can be obtained as $\sigma = \frac{\sigma_e}{\sqrt{2}}$. In particular, they talk about the "edge-stopping" ability of a Tukey function that leads to preventing the spillage of the estimates of a denoised image across image boundaries. As a result, a relatively sharper denoised image can be obtained and this also provides a way to detect edges that are outliers to the robust cost function being used. Similar estimate of scale parameter has also been used by [Tommasini et al., 1998], [Fusiello et al., 1999], [Toldo and Fusiello, 2009] and [Moschini and Fusiello, 2009] who employ the $X84$ rejection rule described in [Hampel et al., 1986] that discards residuals with $|r_i - \mathsf{median}_i r_i| > 3.5\sigma_e$. [Comport et al., 2006] use LSMed to obtain the estimate of the scale parameter and use the Tukey function with scale factor $\sigma = 4.6851\sigma_e$ and Huber function $\sigma = 1.345\sigma_e$.

This scale parameter is then obtained afresh every iteration of the optimisation and therefore determines the weights for each pixel. The optimisation begins with an initial estimate of the parameters obtained using a standard un-weighted least-square approach with all weights set to unity. This provides a way to obtain the residuals and thereby the scale parameter. A robust cost function can be used then for later iterations [Huber, 1981] and the whole process is repeated until convergence.

[Hager and Belhumeur, 1998] use a somewhat different form of IRLS originally proposed by Dutter and Huber [Dutter and Huber, 1981]. They run few more inner iterations obtaining updates where weights are computed afresh every inner iteration keeping the Jacobians, residuals and the scale factor constant but for outer iterations.

4.6 Minimisation Schemes

The normal equation 4.20 has a nice structure of the matrix A that it is symmetric positive definite. Therefore, the solution can be obtained without explicitly computing the inverse of the A matrix. There are many different matrix factorisations tools that can used to solve the system of linear equations, *e.g.* LU decomposition, SVD, QR factorisation and *Cholesky*. However, only *Cholesky* decomposition [Press et al., 1992, Shum and Szeliski, 1997] exploits the positive definite property of matrix A and is nearly twice as efficient as LU decomposition.

It is important to note that the underlying optimisation method that solves the normal equations is approximating the Hessian of the function that is being minimised with the square of Jacobians of the function and is called the **Gauss-Newton** method. **Gauss-Newton** can perform poorly if the approximated-Hessian matrix is ill-conditioned or the residuals are not "small". This is a clear indicator that the approximation of the Hessian matrix is ill-suited for the system of linear equations. This can lead to either convergence to a unexpectedly wrong estimate or leading the solution to bounce up and down around an estimate.

While **Gauss-Newton** style optimisations are well-suited when initialised close to a local minimum where quadratic approximations are good, there are many other different variations to the **Gauss-Newton** approximation that work better if the solution is far away from the minimum *e.g.* **steepest descent** actually computes the full Hessian rather than just an approximation it. However, a major concern with all these optimisations is that there is no feedback associated with the process that is driving the optimisation to a minima. Theoretically speaking, the error must decrease as the optimisation is iterating. As a result the iterative procedure involved is blindly optimistic that the solution will get to the minimum. Therefore, there is no guarantee that energy will decrease due to numerical inaccuracies, interpolations and other discrete approximations to the continuous domain being used in the optimisation.

An optimisation scheme that monitors the change in the error as the iterations progress is **Levenberg-Marquardt**. It allows to combine the best of both worlds, the **Gauss-Newton** and **Line-Search**. The normal equations are then modified:

$$\mathrm{Ax} = \mathrm{b} \quad (GN) \tag{4.35}$$

$$(\mathrm{A} + \lambda \mathrm{I})\mathrm{x} = \mathrm{b} \quad (LO) \tag{4.36}$$

$$(\mathrm{A} + \lambda \mathrm{diag}(\mathrm{A}))\mathrm{x} = \mathrm{b} \quad (LM) \tag{4.37}$$

Levenberg (LO) and Levengberg-Marquardt (LM) optimisations add a stabilising or damping factor, λ, to ensure convergence [Eade, 2009]. To control the parameter λ, a schedule to change its values is recommended in [Press et al., 1992, p. 684], [Szeliski and Coughlan, 1997] and [Baker and Matthews, 2004]. It is initialised with a value of 0.01 and if the residual error has decreased after the iteration then λ is decreased as $\lambda/10$ ([Eade, 2009] recommend halving it) and the new update is performed while an increase in error means that update obtained is erroneous and must not be used as a result, the parameters are restored to their values before the iteration and λ is increased as $\lambda \times 10$ and a new iterative update is sought.

It is also worth mentioning that the inverse compositional approaches [Baker and Matthews, 2004] that benefit from the computational savings that can be made by reversing the role of 2D template and the observed current image, and their further even more efficient second order extensions like Efficient Second Order Minimisation (ESM) [Benhimane and Malis, 2004] do not find equivalent analogue inverse compositional $2\frac{1}{2}$D tracking or 3D-ESM when tracking from a $2\frac{1}{2}$D surface [Baker et al., 2004b]. The template image is anchored to the structure of the surface and that switching roles of template and observed image would mean we know the depth in the observed frame which is not true because this is what we want to know by optimising over the transformation parameters.

4.7 Coarse to Fine Pyramid Hierarchy

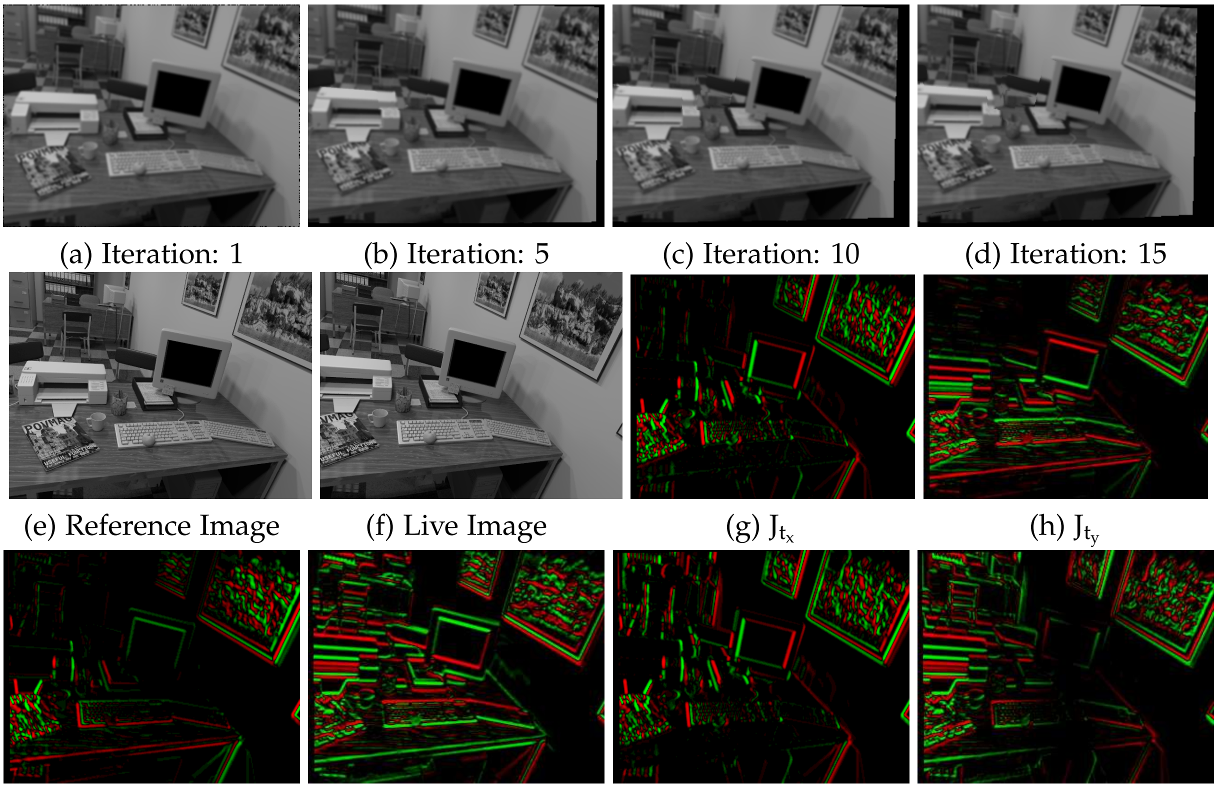
Most all-pixel dense approaches that work by linearising the cost function are not suitable for recovering large motion estimates at a standard resolution of 640 × 480 due to the fact that linearisations hold true only within a narrow range of motion and as a result their radius of convergence is small. In order to track bigger motion, the standard technique is to work on a coarser resolution where observed motion in the image would be considerably smaller than the higher resolution. Such coarser resolution would support the linearisation assumptions made in the trackers and the resulting solution obtained at this level provides a very good initial guess to instantiate the tracking at the higher resolution. Paradoxically, the number of coarser resolutions vary with the extent of the motion present in the image.

In many standard vision applications, a hierarchy of pyramid levels is used where each pyramid level corresponds to a progressive scale by half resolution image. The coarsest pyramid level then initiates the matching and provides a good initial guess to the next level in the hierarchy that keeps the immediate higher resolution. The solution constrains the search space for motion parameters and the whole process is repeated until the pyramid level containing the highest resolution. [Quam, 1984] proposed a hierarchical coarse-to-fine stereo matching algorithm. [Anandan, 1987, Anandan, 1989] and later [Bergen et al., 1992], introduced a framework to measure large displacement motion using the standard coarse-to-fine strategy. Today this strategic hierarchical framework has become a vital ingredient to large displacement motion estimation processes.

It should be noted that care must be taken to ensure that the images are filtered using low-pass filters to wipe out the high frequencies that may introduce aliasing while down-sampling to obtain a coarser resolution.

4.8 Instructive Example

An example of **SE(3)** tracker is shown in Figure 4.3 where different iterations of the tracker running at 320×240 are tiled. As iterations progress, reference image is aligned onto the live image. Different Jacobian images are also shown where a small movement on each of the 6 dimensions leads to changes in the gradients in the image intensities.



(a) Iteration: 1 (b) Iteration: 5 (c) Iteration: 10 (d) Iteration: 15

(e) Reference Image (f) Live Image (g) J_{t_x} (h) J_{t_y}

(i) J_{t_z} (j) J_{ω_x} (k) J_{ω_y} (l) J_{ω_z}

Figure 4.3: Top row shows reference image warped onto live image at various different iterations. Jacobian images for all different parameters that are optimised over are show in the next rows. The colour coding has green for negative gradient, black for zero while red means a positive valued gradient. Only as an instructive example, we have shown the results at resolution 320×240 otherwise a standard coarse to fine pyramid scheme is used.

Chapter 5

Evaluation of Tracking and Synthetic Test-Bed

Contents

5.1 What Questions would we Like to Answer?

Our key goal in this work is to analyse the performance of 6DoF camera tracking in a known 3D rigid scene under varying camera frame-rate (or strictly exposure time in computational photography parlance). This is mostly driven by our intuition that high frame-rate should be better because image motion between consecutive frames reduces considerably when the frame-rate setting of the camera is turned up. Any tracking algorithm that is aimed towards real-time is more likely to maintain the performance if the computations performed per unit of time decrease — this is most likely to happen at high frame-rates where motion between two consecutive images is relatively small. Many direct tracking algorithms that work on linearising the cost function to obtain a convex approximation, the linearisations become increasingly more valid because of small motion assumption at high frame-rates and in feature based systems we get a similar benefit from smaller prediction regions. Therefore, we would like to examine the effects of high frame-rates on the tracking performance and answer the following very simple questions:

- If we have a limited computational budget available on our processor, what is the optimal frame-rate for tracking?
- If we want to work on a given frame-rate, what kind of processor should we use?

However, when doing so, we quickly realise that there are few more parameters that we can change that can affect the performance of a tracker. These parameters are intertwined with frame-rate when it comes to performance evaluation. An immediate extra parameter that can be used in the analysis could be image resolution. Therefore given additional parameters, being more specific we would like to find out answers to the following altered questions

- If we have a limited computational budget available on our processor, what is the optimal frame-rate and image resolution for tracking?
- If we want to work at a given frame-rate and image resolution, what kind of processor should we use?
- If processing budget allows, we can run more iterations of any tracking algorithm X we are using to obtain more accurate results. So how many more iterations can we run?

This is all aimed towards answering questions as a tracking expert/consultant: a user can fix any subset of parameters and we will give range of optimal settings of the output they have.

5.2 Need for Synthetic Test-Bed

In order to be able to vary all the parameters continuously and compare the performance against a perfect ground-truth, we would need a framework that allows us to do this to judge which frame-rate is optimal? We believe, for this analysis there are strong arguments against doing it accurately with a real camera system. Therefore, we appeal to a synthetic framework for the following reasons:

- We cannot obtain perfect ground truth depth-map as well ground truth camera poses even with using an expensive equipment for our analysis.
- We would like to vary frame-rate and image resolution continuously and most cameras offer standard image resolutions. Also, since we would like to also obtain images at frame-rates as high as 200–400Hz, we then need a sensor that can sample as high as 400Hz to give us ground truth.
- We also realise that in real world scenes, lighting cannot remain the same all the time and therefore we need a way to control scene lighting.
- We also need a repeatable motion of the camera so that all frame-rates can be obtained with the same camera trajectory. This demands a mechanism that can give us repeatable motion which we can mount our camera on.

However, all these concentric views on using a synthetic framework do not mean that the real experiments are not possible. We do verify the conclusions from our synthetic test-bed against a well controlled real experiment.

5.3 Choosing a Tracking Algorithm

We choose to focus our analysis on dense tracking algorithms that use all the pixels in the image. The task of image alignment in tracking is posed as locating the lowest point on a surface covered in hills and valleys by doing an explicit ”gradient-descent”; one starts from a predicted location and looks around to find a lower spot and keeps on iterating this procedure until one reaches a point where one cannot go downhill any more.

Dense gradient-descent based tracking algorithms implement active processing implicitly, since if run from the starting point of a good prediction they will require fewer iterations to converge to a minimum. Given an increase in frame-rate, we would expect that from one frame to the next the optimisation will converge faster and with less likelihood of gross failure as inter-frame motion decreases and the linearisation at the heart of Lucas-Kanade tracking becomes increasingly valid. Specifically then, besides the point that we are now seeing increasing practical use of dense tracking methods, we have chosen such a framework within which to perform our experimental study on tracking performance because iterative gradient-based image alignment aims in a direct, pure way at the best possible tracking performance (since it aims to align *all* of the data in an image against the scene model), and makes automatic the alterations in per-frame performance (computation, accuracy and robustness) we expect to see with changing frame-rate. Any feature-based method we might have instead chosen places an abstraction (feature selection and description) between image data and tracking performance which is different for every feature type and would lead us to question whether we were discovering fundamental properties of tracking performance or those of the features used. Feature algorithms have discrete, tuned parameters and thresholds. Further, as we will see in our experiments, there is a complicated interaction between physical image formation blur and noise effects and frame-rate. A dense tracking framework allows an analysis to be made on such degraded images without altering algorithm parameters that might be necessary for feature-based methods.

5.4 How do We Evaluate a Tracker?

When we talk about assessing the performance of a tracker, there are several measures that can be used — a single measure is not always very informative. We define the performance in terms of three different metrics namely accuracy, robustness and computational cost. However, it is important to remember that all three metrics have different dimensions and units and as such cannot be combined like

$$\text{Performance} \neq \frac{\text{Accuracy}}{\text{Computational Cost}} + \text{Robustness} \tag{5.1}$$

An appropriate weighted combination could be a way to combine these metrics. However, these weights are not very trivial to obtain and only serve for somewhat ad-hoc way to combine them which is again not very adequate. Therefore, we consider instead a multi-objective criteria, Pareto Front [1], a multi-dimensional axes vector to judge the performance.

[1] Pareto Front is a popular technique used in economics and described in detail here: http://en.wikipedia.org/wiki/Pareto_efficiency

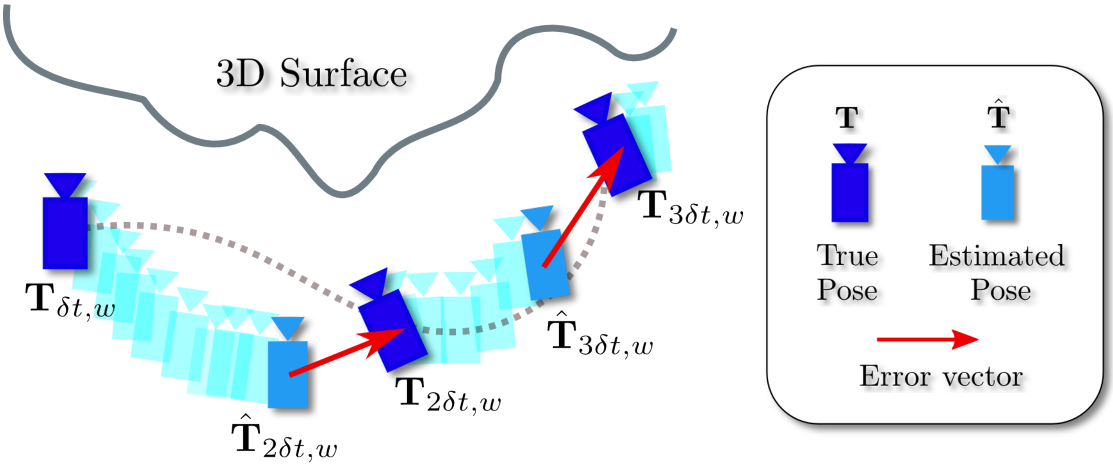

3D Surface
$\mathbf{T}_{\delta t,w}$
$\hat{\mathbf{T}}_{2\delta t,w}$
$\mathbf{T}_{2\delta t,w}$
$\hat{\mathbf{T}}_{3\delta t,w}$
$\mathbf{T}_{3\delta t,w}$
$\mathbf{T}$
$\hat{\mathbf{T}}$
True Pose
Estimated Pose
Error vector

Figure 5.1: Tracking experiment configuration and evaluation. At each frame of our synthesized sequence, after tracking we record the translational distance between pose estimate $\hat{\mathbf{T}}_{t,w}$ and ground truth $\mathbf{T}_{t,w}$; an average of these distances over the sequence forms our accuracy measure. We then use the ground truth pose as the starting point for the next frame's tracking step.

We detail our performance metrics below and later we show how to use them in a multi-objective performance evaluation.

5.4.1 Accuracy

Accuracy captures the degree of proximity of the obtained result to a known ground truth value. Most trackers operating in the regime where they can perform without failures provide estimates that can be used to judge the *accuracy* of a tracker. Thus, it is this measure that separates an *accurate* tracker from an *inaccurate* one.

We define the accuracy of tracker as the euclidean distance between the estimated translation and the ground truth translation. Figure 5.1 shows in how the accuracy measure is calculated. Such measure captures the straightforward deviation from the true value per frame.

Another accuracy measure could be to find the deviation at the last frame by letting the tracker run on its own for a given number of frames, without re-intialising at every frame in between.

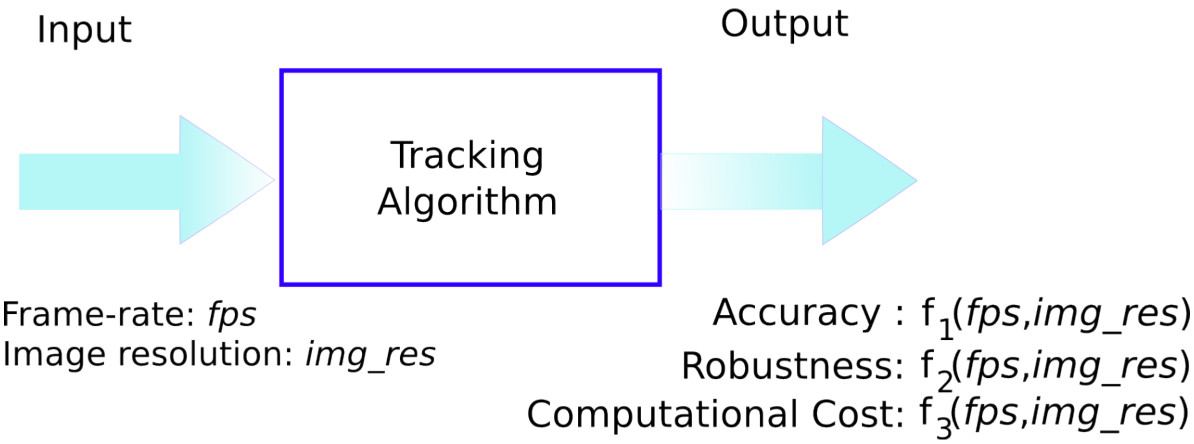

Input
Output
Tracking Algorithm
Frame-rate: *fps*
Image resolution: *img_res*
Accuracy : f_1(*fps*,*img_res*)
Robustness: f_2(*fps*,*img_res*)
Computational Cost: f_3(*fps*,*img_res*)

Figure 5.2: Input-Output parameter space for our tracker. We rate the performance of a tracker based on three paramters namely *Accuracy*, *Computational Cost* and *Robustness*. *Accuracy* captures the degree of correctness of a tracker, *Robustness* is defined as the number of times tracker works without gross failures while *Computational Cost* denotes the underlying processing demands per second.

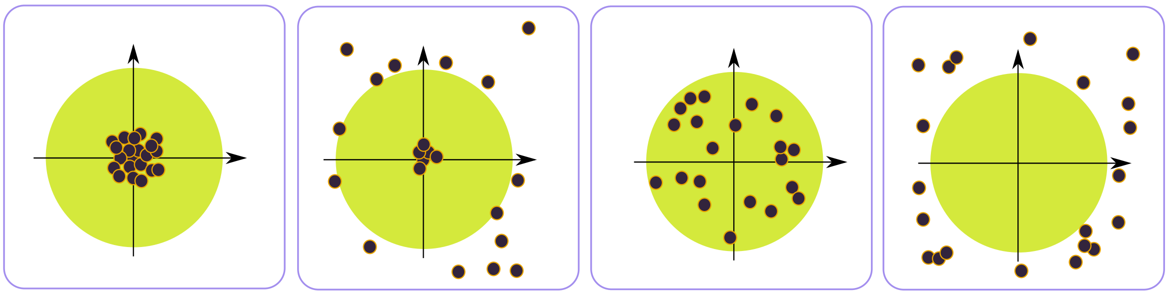


Figure 5.3: Accuracy and Robustness are somewhat tied together but the subtle difference is shown in the figure where we describe four different kind of trackers, (a) Accurate and Robust (b) Accurate but not Robust (c) Robust but not Accurate (d) Neither Robust nor Accurate. The trackers are expected to return a value very close to the origin and the data points represent the samples over which the performance of a tracker is evaluated.

5.4.2 Robustness

The ability of a tracker to remain immune to varying degrees of degradation applied to input data and still give consistently *good* estimates (in a statistical sense) determines the robustness of a tracker. The degradations may include changing illumination in the scene, noise in the image and/or occlusions and rapid and abrupt camera motion producing blurry images. Working in real world conditions means that such degradations are expectedly common and greatly thwart the performance of a tracker, making it brittle. A tracker that works accurately in normal controlled and tuned conditions is hardly of any use in scenarios where long term performance is crucial, if it cannot withstand these challenges and fails catastrophically. For instance, a tracker when given perfect input gives the best possible results but breaks irresistibly against even a small degradation introduced in the images cannot be put to use in real world applications.

Accuracy alone then is an impoverished measure to judge the performance of a tracker because there is an inherent assumption underpinning it that the tracker is working and that accuracy can be computed. *What it does not tell us if the tracker fails or if the tracker works only in a limited scope or it fails more often than it works*. A robustness metric is therefore necessary then to quantify the long term performance of a tracker and compare different trackers under a standard performance metric. If accuracy reports the statistical measure of the degree of correctness of a tracker, robustness is the statistical consistency of the tracker to give correct estimates. Figure 5.6 shows the subtle difference between Accuracy and Robustness.

Robustness, unlike accuracy, requires a large amount of data that captures all the degrees of degradations observed in the image to obtain a statistically meaningful number that reflects the success-failure ratio. Most tracking algorithms used in SLAM only define robustness qualitatively showing either the success or failure of the used tracker against the degradations; the enormity of data required prohibits quantitatively evaluation of robustness. So far, only [Coffin et al., 2010b] and [Coffin et al., 2010a] analyse the performance of 3DoF camera orientation tracking system and propose a robustness metric to quantify the performance. They organise the tracking into three different categories: acceptable, recoverable and irrecoverable tracking region based on fixed thresholds (that are domain specific) they determine. Robustness is then expressed as a linear combination of the frequency of the time the tracker spends in each of these regions. Expert users are asked to rate the tracking subjectively and based on the obtained statistics they minimise the sum of square differences of quantified robustness and the statistics of robustness obtained from expert users to determine the weights. They obtain the weights from the results of three of the four methods they use to perform tracking and verify the accuracy of the metric by being able to correctly predict the robustness of the fourth.

We obtain similar robustness statistics by quantifying the success/failure and calculating the frequency of success of the tracker. This is also illustrated in Figure 5.6 where any data point inside the green circle means that the tracker is successful. The success/failure is determined by a threshold.

5.4.3 Computational cost

Our computational cost model evaluates the total occupancy ratio: the ratio of the amount of time a tracker keeps the processor busy to strict timing demands *i.e.* $\frac{1}{fps}$, expected for the tracker to finish. This is formally defined as

$$\begin{aligned} \textbf{Computational Cost} &= \frac{\textbf{time taken per frame}}{\frac{1}{\textbf{fps}}} && (5.2) \\ &= \textbf{time taken per frame} \times \textbf{fps} && (5.3) \end{aligned}$$

Since the computational cost is a dimensionless quantity, in other words it can also be interpreted as the total computational power spent in running the tracker on the number of frames provided by operating at a given frame-rate. For instance, if we are using 200Hz camera, it means the computational power spent in processing 200 frames in total.

Figure 5.2 summarises the input-output parameter space for our tracking evaluation. Having identified various performance metrics which seem important in evaluation of tracking, we detail on how do we put them together in the next section.

5.5 Multi-Objective Cost Functions: Pareto Fronts

Performance evaluation and assessment demand a criteria or metric to judge the performance of a system when subjected to different input functions it depends on. Not only does it highlight a set of various operating points where the system performs without failures, but also provides a standardised scale on which the performance of two different operating points can be compared objectively. This metric could be accuracy or a combination of metrics that returns a scalar quantity to merit the performance of the system. Although a single metric, e.g. accuracy could be appealingly useful, most assessment and evaluation methodologies require more than just one metric. For instance, one may also be interested in weighing the computational cost (a penalty) associated with attaining a given accuracy figure. Therefore, the performance should be collectively assessed based on multi-objective criteria.

5.5.1 Formal Definition

Our assessment function here has three different criteria over we wish to assess the performance of a tracker. Formally, the assessment function for tracker can be written in terms of these criteria as follows:

$$f(\mathbf{x}) \quad = \quad \begin{cases} \min \mathcal{E}(\mathbf{x}) & \textbf{(Error / Accuracy)} \\ \min \mathcal{C}(\mathbf{x}) & \textbf{(Computational cost)} \\ \max \mathcal{R}(\mathbf{x}) & \textbf{(Robustness)} \end{cases} \tag{5.4}$$

$\mathcal{E}(\mathbf{x})$ is the measure of error described as the deviation from its true value, $\mathcal{C}(\mathbf{x})$ denotes the cost associated with running a tracker measured by the amount of time it takes until convergence and $\mathcal{R}(\mathbf{x})$ is the third assessment of the quality of a tracker judged in terms of the frequency of failures of a tracker over a given period of time.

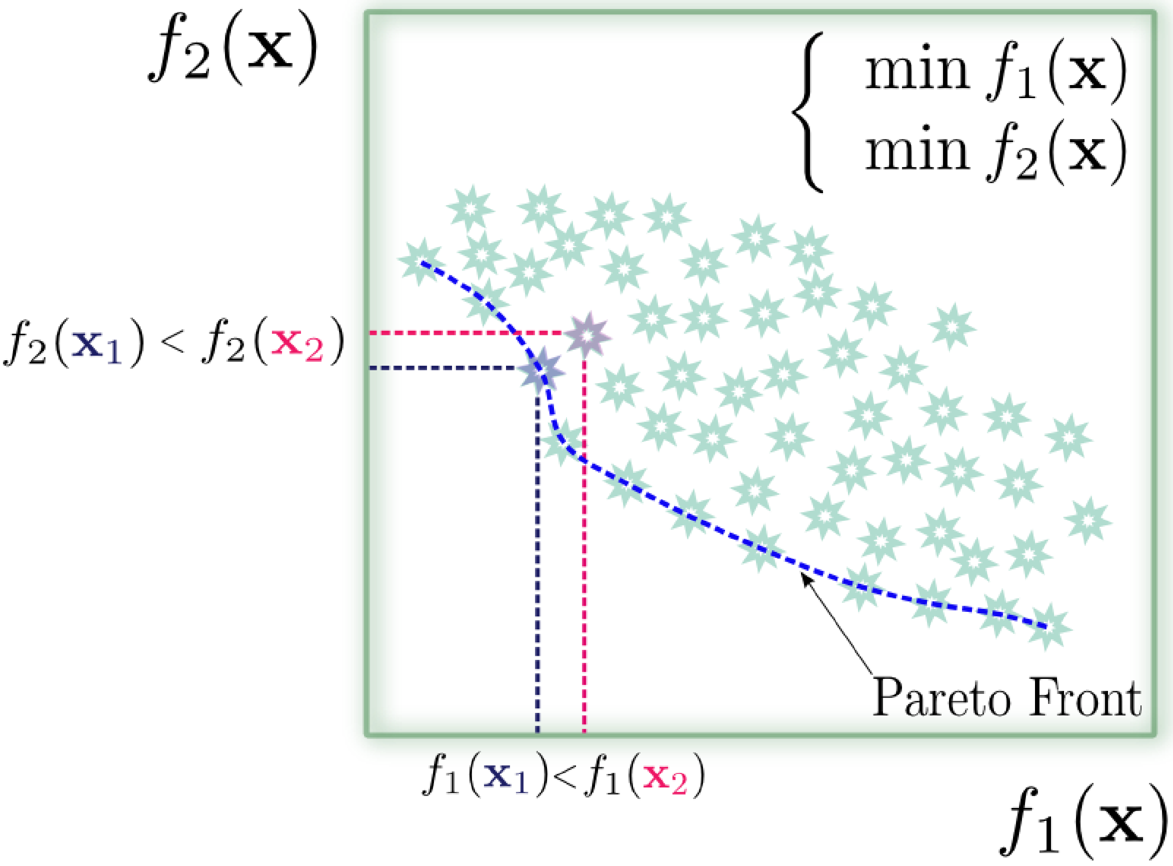
Since all the three performance measures are dimensionally different to each other, it is not straightforward to combine (e.g. weighted combination) them to obtain a single scalar quantity that sums up the final assessment. Therefore, a single global optimal solution does not exist. Rather there is multiple choice solution that is considered dominant. The dominant solution is termed as Pareto optimal solution while the envelope of all dominant solutions over a given set of possible ranges forms the Pareto Front as shown in Figure 5.4. The user desirable output parameters allow to slide over the Pareto Front to obtain the corresponding input parameters that yield Pareto Optimal solutions — an envelope of operating points from which a user might sensibly choose.

Pareto Fronts are an old idea from economics. In computer vision, evaluation of algorithms and systems using multi-objective Pareto Fronts has been previously visited by [Everingham et al., 2006] in the context of image segmentation, [Calonder et al., 2010] for interest point, [Mayol et al., 2002] for wearable camera placement, [Dunn et al., 2004] for sensor planning in active vision system.

5.5.2 Pareto Optimality

An illustration of Pareto Optimality is given in Figure 5.4. A Pareto Front forms the lower envelope of evaluation parameters plotted one against the other. The Figure shows the Pareto analysis of the following evaluation

$$\begin{cases} \min f_1(\mathbf{x}) \\ \min f_2(\mathbf{x}) \end{cases} \tag{5.5}$$



$f_2(\mathbf{x})$
$\begin{cases} \min f_1(\mathbf{x}) \\ \min f_2(\mathbf{x}) \end{cases}$
$f_2(\mathbf{x}_1) < f_2(\mathbf{x}_2)$
Pareto Front
$f_1(\mathbf{x}_1) < f_1(\mathbf{x}_2)$
$f_1(\mathbf{x})$

Figure 5.4: Pareto Front of the feasible operating points. The assessment criteria seeks the minimum of two both objective functions $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$. Solution $\mathbf{x}_1$ dominates $\mathbf{x}_2$ i.e. $\mathbf{x}_1 \succ \mathbf{x}_2$ because for both objective functions $\mathbf{x}_1$ yields the optimal results. The dashed blue line forms the lower envelope of the scattered operating points dotting the plot of the objective functions.

Similarly for our analysis, if for some $\mathbf{x}_1$ and $\mathbf{x}_2$, there exists the following

$$\begin{aligned}
&\mathcal{E}(\mathbf{x}_1) \leq \mathcal{E}(\mathbf{x}_2) \text{ and } \mathcal{C}(\mathbf{x}_1) < \mathcal{C}(\mathbf{x}_2) \text{ and } \mathcal{R}(\mathbf{x}_1) > \mathcal{R}(\mathbf{x}_2) \text{ or}\\
&\mathcal{E}(\mathbf{x}_1) < \mathcal{E}(\mathbf{x}_2) \text{ and } \mathcal{C}(\mathbf{x}_1) \leq \mathcal{C}(\mathbf{x}_2) \text{ and } \mathcal{R}(\mathbf{x}_1) > \mathcal{R}(\mathbf{x}_2) \text{ or}\\
&\mathcal{E}(\mathbf{x}_1) < \mathcal{E}(\mathbf{x}_2) \text{ and } \mathcal{C}(\mathbf{x}_1) < \mathcal{C}(\mathbf{x}_2) \text{ and } \mathcal{R}(\mathbf{x}_1) \geq \mathcal{R}(\mathbf{x}_2)
\end{aligned}$$

then the solution $\mathbf{x}_1$ is said to *dominate* $\mathbf{x}_2$ i.e. $\mathbf{x}_1 \succ \mathbf{x}_2$. It shows that an improvement can be made without making the other objective functions worse. An optimal solution then is the one that is not dominated by any other solution. This permits us to answer questions of the form: *What is the best choice of input parameters for a tracker that gives an accuracy of* **A% but**

Elegant Practical Fancy Cheap

Figure 5.5: Eating out in a restaurant can be considered as an engineering problem. The choice of a restaurant greatly depends on multiple criteria *e.g.* good ambience, low price, high quality etc. One restaurant may not fit all the criteria, instead it may reflect a fair balance of them and it is upon the person to choose sensibly. The images are obtained from `http://www.die.unipd.it/~alotto/didattica/corsi/Elettrotecnica%20computazionale/pareto.pdf`

runs at a speed of **T***ms* **and** *works* **R%** *of the time without failures.*

5.5.3 Toy Example: Eating Out in a Restaurant

Eating out in a restaurant can be considered as multi-objective optimisation where the decision to go to a restaurant is governed by many objectives, *e.g.* the ambience, the price and the quality rating of the restaurant as shown in Figure 5.5. However, the objectives can be conflicting that results in multiple choices offered to the person instead of a single choice. Table 5.1 summarises the multiple objectives with the choices of restaurants. It is clear that although a weighted combination of these objectives will provide us with a single scalar number which we can use to judge the best choice however, it is biased towards the combination of weights used.

Restaurant	$\mathbf{f_1}$**: Price**	$\mathbf{f_2}$**: Quality [1=max,5=min]**	$\mathbf{f = af_1 + bf_2}$ a=$\frac{1}{60}$, b=$\frac{1}{5}$	$\mathbf{f = af_1 + bf_2}$ a=120, b=$\frac{1}{5}$
Joe's Juicy Stake	30	2	0.9(2)	0.65(1)
Pizza Pazza	15	4	1.05(3)	0.93(4)
Mc Duck's chicken	10	3	0.77(1)	0.68(2)
The Golden Shrimp	60	1	1.2(5)	0.7(3)
The Cold Soup	5	5	1.08(4)	1.04(5)

Table 5.1: The choice of a restaurant is governed by somewhat conflicting objectives, price and the quality. The cons of weighted combination are evident that it although yields a single scalar value but the choice of best restaurant depends on the weighting used.

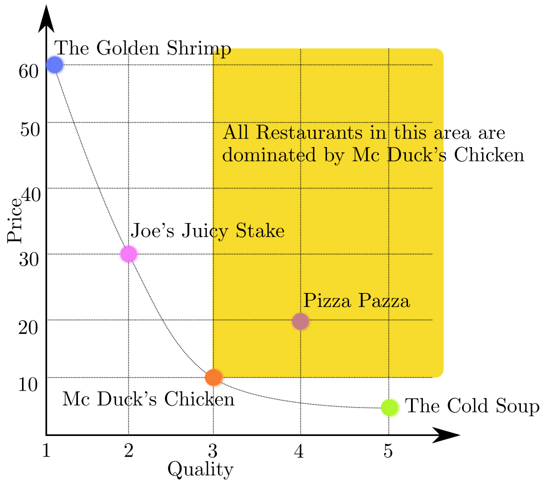


Figure 5.6: A Pareto Front on the other hand does not make any strict decision based on a scalar number instead it provides a set of optimal choices which are considered to be dominant.

5.6 Synthetic Image Generation via Ray Tracing

Motivated by the need for a synthetic framework to allow full control of the necessary parameters, we have followed the recent advancements in the computer graphics community. There has been a huge proliferation of 3D modellers and software during the past decades that allow one to create a synthetic scene with very realistic elements. Ray tracing lies at the heart of many of these projects e.g. POVRay [2], Blender [3], OptiX [4], Radiance [5] and Sunflow [6] to name a few, and allows to define the scene geometry and model all the real world effects and lighting interactions. A typical ray tracer has the type of camera and a 3D scene defined usually in either a software-specific language or some intermediary tools. Scene used in a ray tracer is composed of objects of various different shapes and sizes. Rendering then, is a procedural way of projecting the complex 3D world defined in terms of simple primitive shapes e.g. cube, sphere etc. in an image by tracing path of rays that hit every pixel in it in the same style as a real camera would do. Figure 5.7 shows a glimpse of the advancements made in ray tracing and that it is not difficult anymore to create realistic scenes. If we look at the latest sci-fi and action movies, new video games and architectural and engineering

[2] www.poray.org
[3] www.blender.org
[4] http://www.nvidia.com/object/optix.html
[5] http://radsite.lbl.gov/radiance/
[6] http://sunflow.sourceforge.net/

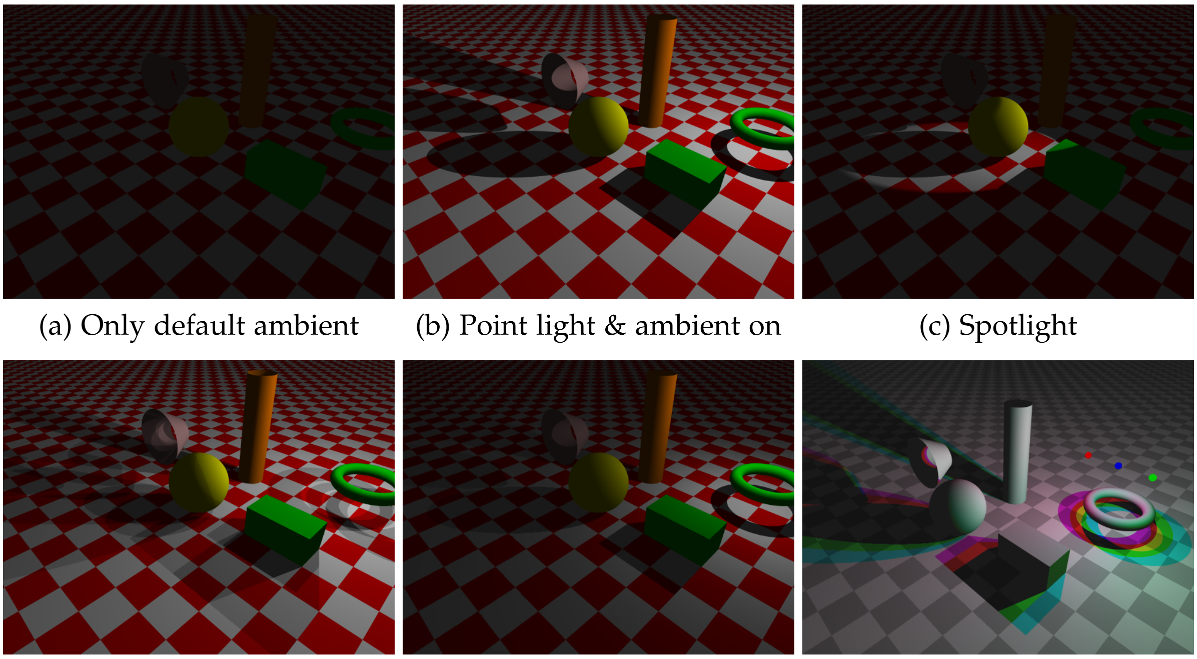
designs such as airplanes, cars, and houses etc., we can easily pin point the degree of realism in them. This is largely again due to the advancements made in the ray-tracing community.

1996 2000 2006

Figure 5.7: Images selected for first prize in the IRTC (Internet RayTracing Competition) in the years marked below each image.

Ray Tracing models the complex light interactions with the objects in the scene to create an image. A ray tracer, in principle, works as a real camera but however, the difference is that the it works in reverse to a real camera i.e. rays are projected back from their point of reception (pixels) to their point of origin (light source). When light rays travel in real world, many complex interactions take place e.g. diffusion, reflection, refraction etc. before they hit the pixel location on a camera. In this process, some rays never reach the camera and it is for this reason that a ray tracer works only on the rays that hit the camera by tracing them backwards and saves the unprecedented amount of calculations involving infinite number of rays it would have to make otherwise. Figure 5.8 and 5.9 highlight various complex real-world effects that can be modelled with a ray-tracer.

A *primary ray* emerging from the camera center of projection through pixel is traced to the scene. The ray is tested for intersection with each object and the nearest intersection is identified. A *shadow ray* is then traced backwards towards the light source. If the ray hits another object on the way, the ray bounces to a different direction and therefore depending on the fixed number of bounces, it either reaches the light source or not. If the ray reaches the light source, the contribution of the colour at that pixel then is obtained by applying the standard optics laws of reflection (and refraction if there is any) on all the intermediate paths the ray takes before reaching the light source. Therefore, ray tracing is a *view-dependent* rendering technique.



(a) Only default ambient (b) Point light & ambient on (c) Spotlight

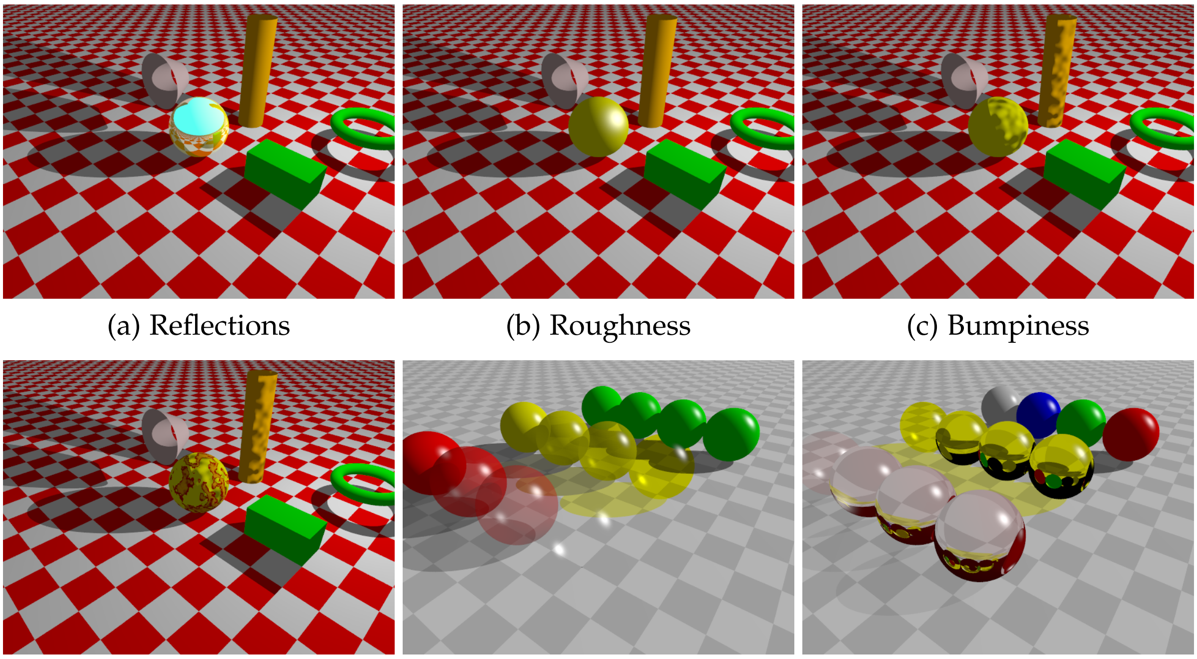
(d) Area light (e) Fading light (f) Lightbulbs

Figure 5.8: A simple scene comprising primitive shapes (cylinder, cuboid, sphere etc.). The same scene is rendered under varying lighting conditions. Figure (a) is rendered with no light sources and default ambient lighting while (b) shows default lighting with a point source that has infinite power and shines in all directions casting hard shadows, (c) shows rendering under a spotlight; (d) is rendered using a 2 × 2 area matrix of light bulbs lying on the vertical plane (e) shows a render with fading light property and (f) is rendered with three different lightbulbs (red, green and blue show in the image). The images are reproduced from the POVRay scripts available at the website `http://xahlee.info/3d/povray-lighting.html`

But this cannot undermine the time a ray tracer still spends in rendering an image. For instance, to obtain a 640 × 480 image, it sends out (one or more) rays for every pixel step by step and renders it. Increasing image size, adding more realistic effects e.g. shadows, reflections etc. and sending more rays in the scene only encumbers the ray tracer and makes it spend more effort and time.

5.6.1 Radiosity

A standard ray-tracing simulator only performs reflections of lights coming from direct illumination. If an object or a part of the scene is being blocked by another object then there is no direct light transport. However, in real world light transport, there are inter-reflections of light among different surfaces and this is exactly what radiosity aims to simulate. This



(a) Reflections (b) Roughness (c) Bumpiness

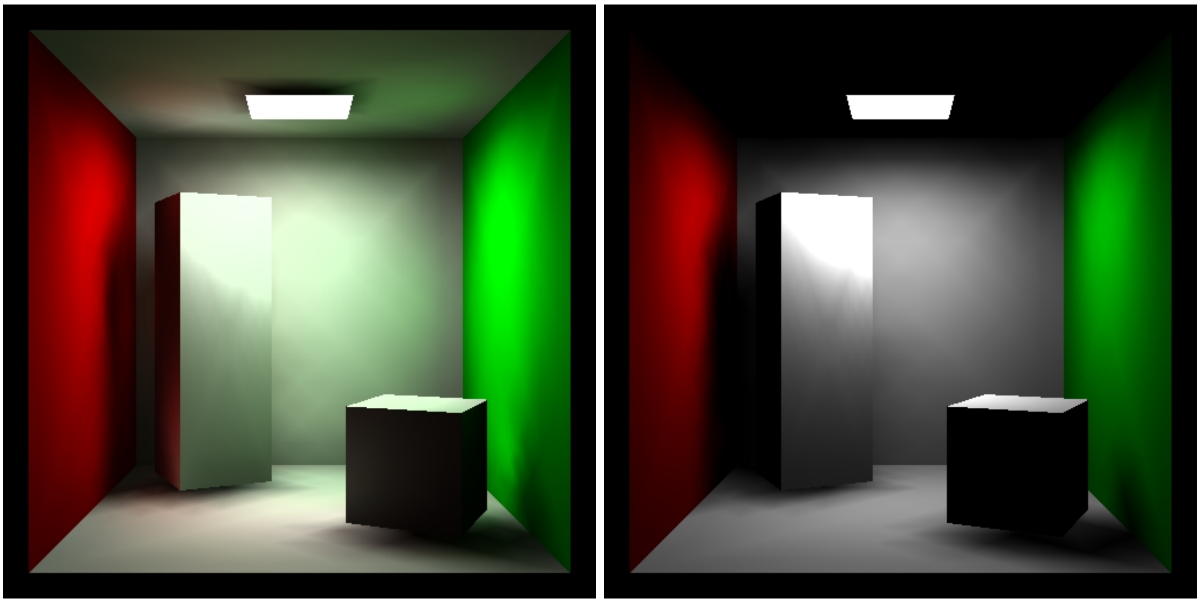
(d) Texturing (e) Transparency (f) Refraction

Figure 5.9: These images are again reproduced from the POVRay scripts available at the website `http://xahlee.info/3d/povray-lighting.html`

gives rise to important realistic visual features in the image e.g. soft shadows and colour bleeding.

Radiosity information is computed between all the diffuse surfaces in the scene based on the visibility form factors i.e. the area of a surface that can be viewed from a different surface which is what contributes to the radiance transfer. Radiosity is *view-independent* and therefore all the computations can be done for the whole scene at once. After computing the radiosity of the whole scene, novel view renderings can be done interactively. Figure 5.10 shows the remarkable difference radiosity brings to the synthetic scene. The front box shows areas that are completely dark due to the lack of direct light transport but with radiosity it is able to obtain a greenish tinge from the green wall to the right. Similar effects are observed at the side of the box, at the wall at the back and the ceiling.

The cost incurred in achieving the realism is huge both in terms of time and memory requirements. The algorithms require the scene to be decomposed into small patches so that radiance transfer can be easily calculated from the *form-factors*. This is quadratic in the number of patches and hence a massive increase in complexity.



(a) (b)

Figure 5.10: The classic Cornell Box, Figure (a) is rendered with radiosity setting **on** while Figure (b) is using **only direct illumination and no global illumination**. How much difference diffuse inter-reflections can make to the scene is clearly evident in the images. Note the colour bleeding at the back wall. A standard ray tracer cannot simulate the light inter-reflections between diffuse surfaces. Therefore *radiosity* is a technique that is often used to recover the diffuse inter-reflections. The images are downloaded from the website http://www.cs.technion.ac.il/~cs234326/projects/ray/ray.htm

For our experiments, we have used the open-source ray tracer POV-Ray as our main tool for video synthesis. POV-Ray has been used previously for ground truth generation [Funke and Pietzsch, 2009] with regard to performance evaluation of a feature-based SLAM system, but with very simple non-realistic scenes. We have instead used a publicly available indoor office scene model [7] ($800 \times 500 \times 250cm^3$). The scene is appealingly similar to a normal office scene with deskstops, keyboards, tables, chairs and paintings. It is rendered in two passes; the first pass saves all the radiosity settings and the second pass uses the computations saved in the first pass and injects specular reflections and other real scene features to give a realistic appearance. Figure 5.11 shows an image from the scene with the recovered depth-map. We generate videos for different frame-rates by inserting trajectories in the scene and then add real-camera artefacts e.g. noise, to the image. The following subsequent sections look at the trajectory generation and adding calibrated image noise processes.

[7] http://www.ignorancia.org/en/index.php?page=The_office

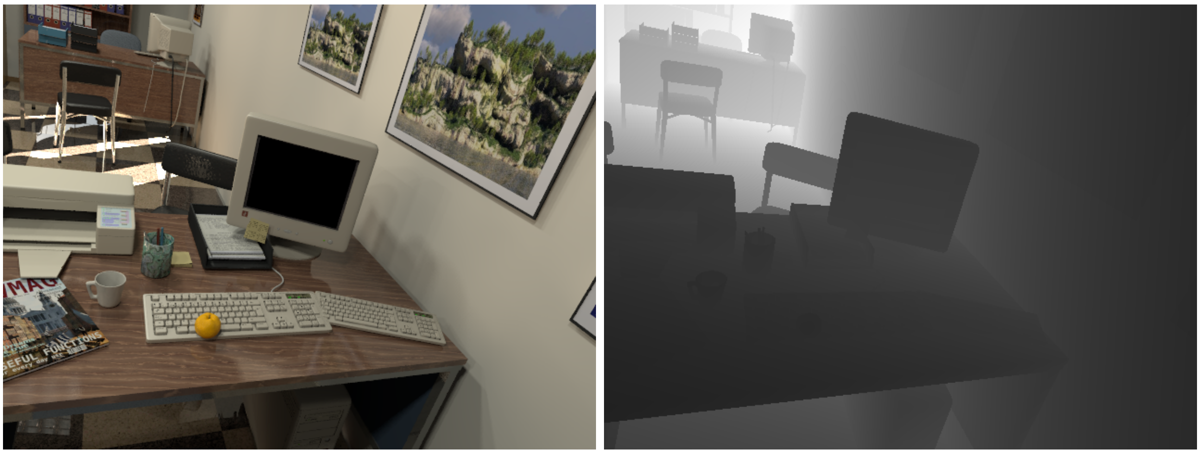


Figure 5.11: A 'pure' ray-traced image with no blur or noise effects. Each such image takes 30–60mins to render in two passes on an 8 core Intel i7 3.20GHz machine. The associated planar depth map also generated is shown alongside.

5.7 Adding Photo-Realistic Image Effects to Synthetic Images

Pure ray-tracing allows one to add realistic effects that are observed as light transports in the real-world. Camera noise that arises due in low exposure images taken at high frame-rates and image blur that is observed at low frame-rates are not easily modelled in this ray-tracing — a ray-tracer would have to send more rays and average the results which is again quite time consuming.

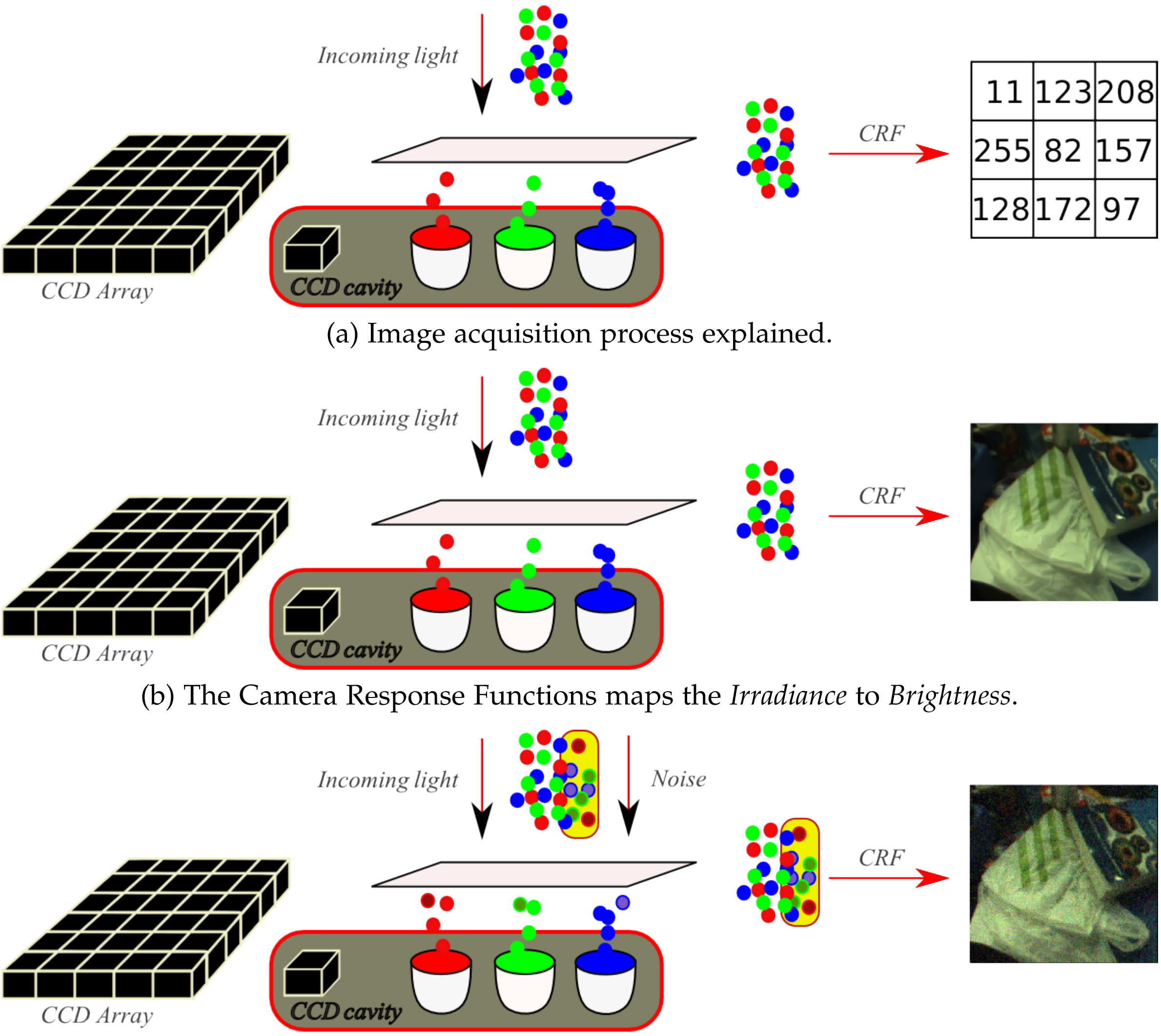
These artefacts are quite an essential part of our experiments where we would like to show the degradation in a tracker's performance with noisy and blurry images. We obtain the camera noise parameters from a real camera and use those parameters to post-process the ray-traced images and add noise to them. On the other hand, we average the irradiance values to obtain the motion blurred image. Below we detail the various noise sources that arise in the camera image capture and show how to model them and later we describe the procedure to add blur to images.

5.7.1 Adding Image Noise

The number of photons collected by a CCD element per unit time is called Irradiance and is defined as:

$$E = \frac{(\pi d^2)\cos^4 \beta L}{c^2}, \tag{5.6}$$

where d the radius of aperture, β is the angle subtended by the principal ray from the optic axis, L is the radiance of the scene and c is the focal length of the camera. Figure 5.12



11	123	208
255	82	157
128	172	97

(a) Image acquisition process explained.

(b) The Camera Response Functions maps the *Irradiance* to *Brightness*.

(c) The incoming photons are also accompanied with noise.

Figure 5.12: The image acquisition process for a global shutter CCD camera. Each CCD cavity collects the photons very analogous to a bucket collecting falling rain drops. The number of photons collected by each bucket gives the pixel its colour value. The process of photon collection to its conversion to a standard 8-bit value is modelled by the camera response function, CRF. The modelling process involves various non-linear transformations e.g. white balance, tone mapping, gamma correction which the value undergoes before being quantised to a computer readable 8-bit number. The photon collection process also comes with accompanying extra photons which is termed as noise. The noise follows a *Poisson distribution* but can be approximated by a Gaussian distribution model if the number of expected photons exceed 10.

shows how and where noise gets added during the imaging process. There are mainly four different types of noise in CCD cameras that corrupt the signal before it is digitised to a value observed on a computer. They are photon noise, dark current noise, read out noise, and quantisation noise. We expand on each of the noise in the following.

Photon noise

A typical CCD site on the sensor array acts as a bucket that collects photons from the photon rain falling from a light source. The more the photons collected, the brighter the pixel value is at that location. However these photons are emitted at random times meaning that at every time the bucket collects photons, it is not guaranteed to collect the same number. In physics of photon counting this is termed *photon noise* or *shot noise*.

The rate at which photons arrive at the CCD site follows a *Poisson distribution*. Photon noise depends on the square root of the number of photons collected [Hasinoff, 2012]. The number of photons N measured by a given sensor cavity over a time of t seconds can be described by the following distribution

$$\Pr(N = p) \quad = \quad \frac{e^{-\lambda}(\lambda)^{p}}{p!} \,, \tag{5.7}$$

where $\lambda = \lambda_t t$ and λ_t is the expected number of photons per unit time and is proportional to the incident scene irradiance. Both the mean and the variance of the distribution are $\lambda_t t$, i.e. the standard deviation of the noise grows with the square root of the signal. For small shutter speed, photon noise is swamped by the signal while for large shutter speeds the effect of photon noise begins to corrupt the signal profusely. However, it is important to remember that this noise has nothing to do with the camera and that it is an inherent property of light source.

In most practical systems, the Poisson noise can conveniently be approximated by Gaussian noise. Figure 5.13 shows how a Poisson distribution turns Gaussian as λ increases beyond 10. This makes it easy to use the appealing properties of a Gaussian distribution while modelling and measuring the amount of noise in images. Understanding the process of photon noise enables us to model the effects of noise that are related to the variations of light entering the sensor, e.g. changing exposure time and camera gain.

Dark current noise

The collected photons in the CCD are then converted into electrons via the photoelectric effect. The excited electrons leave the molecules of the material and transfer the charge when external voltage is applied. This charge is collected by the capacitors which then convert it back to a voltage for the Analog-to-Digital converter. However, these electrons can also get excited by heat in the sensor which leads to what is known as dark current noise. It is essential to remember that this type of noise emanates due to the property of the material of the sensor used in the CCD and it has nothing to do with the light. Even if the sensor does not receive any light, it is still possible to have this noise in the image because it depends on the temperature of the sensor.

Read out noise

On converting the charge to voltage by capacitor, this voltage is amplified and the noise resulting from the amplification is termed read out noise.

Quantisation noise

The noise resulting from the conversion of voltage to a digital value by the Analog-to-Digital converter is termed quantisation noise. A given n-bit ADC has a limit to which it can measure the voltage accurately and convert into a digital value.

The digital pixel value often in most cameras also goes through various non-linear transformations before it gets converted to a value a human visual photometer can contrast easily. These transformations may include gamma correction, white balance and tone mapping etc. A camera response function (CRF) then allows us to model these variations to provide an abstract relationship between input *Irradiance* and output *Brightness*. This response function is usually monotonic and invertible and reflects the modelling variations typical of the camera used. [Grossberg and Nayar, 2003] provide a database of camera response functions for different consumer cameras.

In order to set realistic rendering parameters for our synthetic sequences, we have performed experiments with a real high frame-rate camera to determine its camera response function (CRF) and noise characteristics. When a pixel on a camera's sensor chip is illuminated with light of irradiance E, during shutter time Δt it captures energy per unit area $E\Delta t$. This is then turned into a pixel brightness value B. We model this process as in [Liu et al., 2006]:

$$B = f(\mathrm{E}\Delta t + n_s(\Delta t) + n_c) + n_q \ . \tag{5.8}$$

Here, the CRF f is the essential mapping between irradiance and brightness, monotonic and invertible. n_s is a shot noise random variable with zero mean and variance Var($n_s(\Delta t)$)= $E\Delta t\sigma_s^2$; and n_c is camera noise with zero mean and variance Var(n_c)=σ_c^2. We assume that quantisation noise n_q is negligible. Our choice of modelling the shot noise with Gaussian distribution is more driven by fact that the number of photons captured by the CCD cavity for even very low lighting conditions are more than 10. Table 5.2 summarises the number of photons obtained for different light sources. We observe that even in the low lighting conditions of a living room, the number of photons received exceed the minimum number of photons required to model the *shot noise* with Gaussian noise. Given the formal relationship between *Brightness* and *Irradiance* this also allows us to obtain the variation of the brightness value observed as a function of various noise. Using first order Taylor series expansion of the function we can easily obtain the standard deviation of the brightness:

$$\mathrm{B} \;=\; f(\mathrm{I}_0) + \left.\frac{\partial f(\mathrm{I})}{\partial \mathrm{I}}\right|_{\mathrm{I}=\mathrm{I}_0} (\mathrm{I}-\mathrm{I}_0)\;. \tag{5.9}$$

Representing I as the input *Irradiance*, variance in *Irradiance* can be obtained as:

$$\mathrm{I} = \mathrm{E}\Delta t + n_s + n_c \qquad \Longrightarrow \sigma_{\mathrm{I}}^2 = \mathrm{E}\Delta t\sigma_s^2 + \sigma_c^2\;. \tag{5.10}$$

The variance of image brightness can be obtained by covariance propagation:

$$\sigma_{\mathrm{B}}^2 \;=\; \left(\left.\frac{\partial f(\mathrm{I})}{\partial \mathrm{I}}\right|_{\mathrm{I}=\mathrm{I}_0}\right)^2 \sigma_{\mathrm{I}}^2\;, \tag{5.11}$$

Therefore, the standard deviation in the observed brightness can be explained as

$$\sigma_{\mathrm{B}} \;=\; \left(\left.\frac{\partial f(\mathrm{I})}{\partial \mathrm{I}}\right|_{\mathrm{I}=\mathrm{I}_0}\right)\sqrt{\mathrm{E}\Delta t\sigma_s^2 + \sigma_c^2}\;. \tag{5.12}$$

We obtain the variance of noise using the method of [Liu et al., 2006] plotting mean brightness against the standard deviation by taking snaphots of a static scene with a still camera. Pixel value at each location in the image stack is averaged to obtain the mean and the standard deviation. Parameters that fit well, as defined by the sum of squares cost function, to the lower envelope of the plot are finally used to add synthetic image noise.

We used the Matlab optimisation toolbox and standard functions `fmincon` (with constraints $\sigma_s \geq 0$ and $\sigma_c \geq 0$) and `fminunc` to obtain the optimal values of the parameters σ_s and σ_c. Optimisation results are overlaid on the observed NLFs from the images for each channel in Figure 5.15.

Light Source	**Lumens/m^2**	**Photons/μm^2 /s**	**Photons**
Starlight	10^{-4}	5	< 1
Full Moon	1	5×10^4	19
Living Room	50	2.5×10^6	965
Office Lighting	4×10^2	2×10^7	7.7×10^3
Overcast Day	10^3	5×10^7	1.9×10^4
Daylight	10^4	5×10^8	1.9×10^5
Direct Sun	10^5	5×10^9	1.9×10^6

Table 5.2: The numbers and figures are obtained from [Cossairt, 2011, p. 150]. These numbers correspond to exposure time of $\frac{1}{50}ms$.

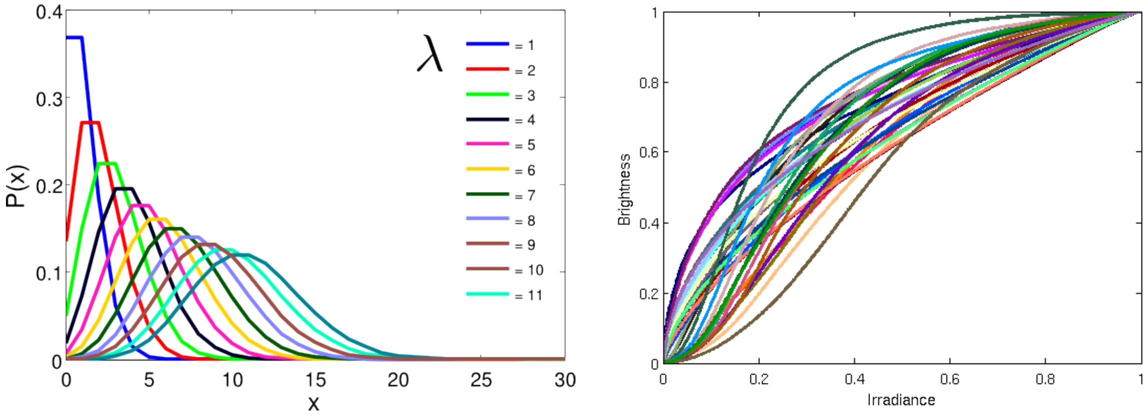


Figure 5.13: (a) Poisson distribution as a function of the parameter λ. The distribution can be approximated by Gaussian distribution for $\lambda \geq 10$. (b) Different camera response functions obtained from the popular Columbia camera response function database http://www.cs.columbia.edu/CAVE/software/softlib/dorf.php

5.7.2 Adding Motion Blur

Motion blur is a major artefact observed in images when the camera is undergoing rapid shaky motion. It occurs due to the way the sensor works by integrating the light intensity over a period of time. Therefore, when a camera (or object in the scene) moves faster than the sampling rate of the image capture, images tend to have motion streaks in them. In image space, this means each pixel value is obtained from the combination of the pixels along the arc of image motion induced by the camera trajectory. In most applications it is reasonable to average the image brightness levels to create a blurry looking image. However, a real camera does not average the *Brightness*, it in fact averages the *Irradiance* that is incident on the sensor since the CCD cavity collects a number of photons. Therefore injecting synthetic

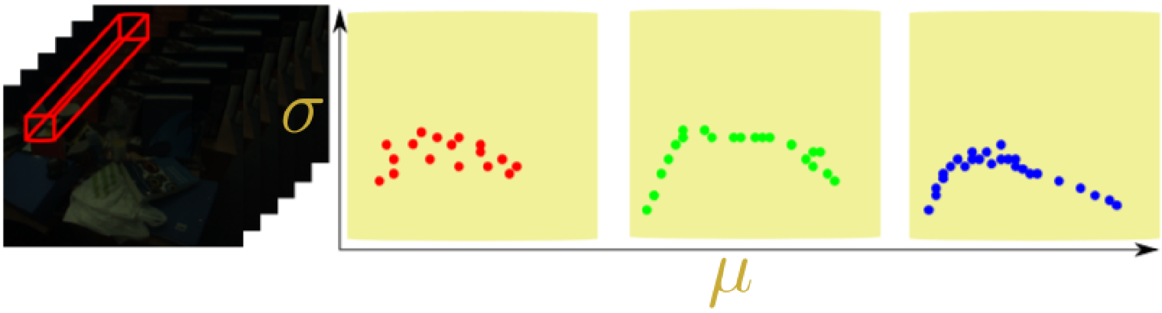
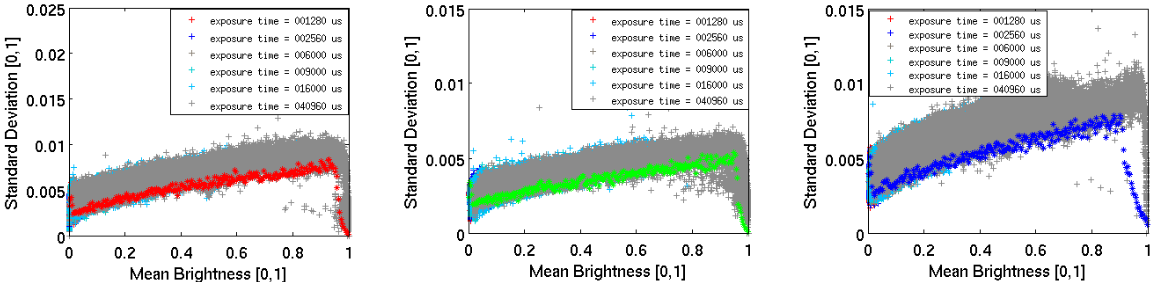


Figure 5.14: An illustration of how the noise level function is plotted. Different channels have means μ plotted against the standard deviation σ obtained from the temporal stack of images of a static scene taken with a stationary camera.



Red Green Blue

Figure 5.15: Data and results for experimental Noise Level Function (NLF) calibration for each channel of our real camera shown as scatter plots. Overlaid on each of the plots is the brightness-dependent NLF computed using the minimisation technique formulated in [Liu et al., 2006]. Note that the green channel has significantly lower noise than red and blue due to there being twice as many green pixels in our camera's Bayer pattern. The optimal sigmas obtained for R are (σ_s = 0.0104, σ_c = 0.0045), for G are (σ_s = 0.0066,σ_c = 0.0038) and for B are (σ_s = 0.0107, σ_c = 0.0053). We also minimised the noise level energy function with R, G and B together and obtained optimal values being ($\sigma_s = 0.0103, \sigma_c = 0.005$).

motion blur in images requires the image acquisition process be modelled carefully.

We have followed the work of [Debevec and Malik, 1997] and recently [Lin and Chang, 2006] for generating motion blur in synthetic images. Both choose to average values in the *Irradiance* space by projecting the *Brightness* to the *Irradiance* plane via the camera response function and then converting the averaged *Irradiance* back to *Brightness* plane for the corresponding blurry value. [Klein and Murray, 2010] also use a similar model to create realistic motion blur.

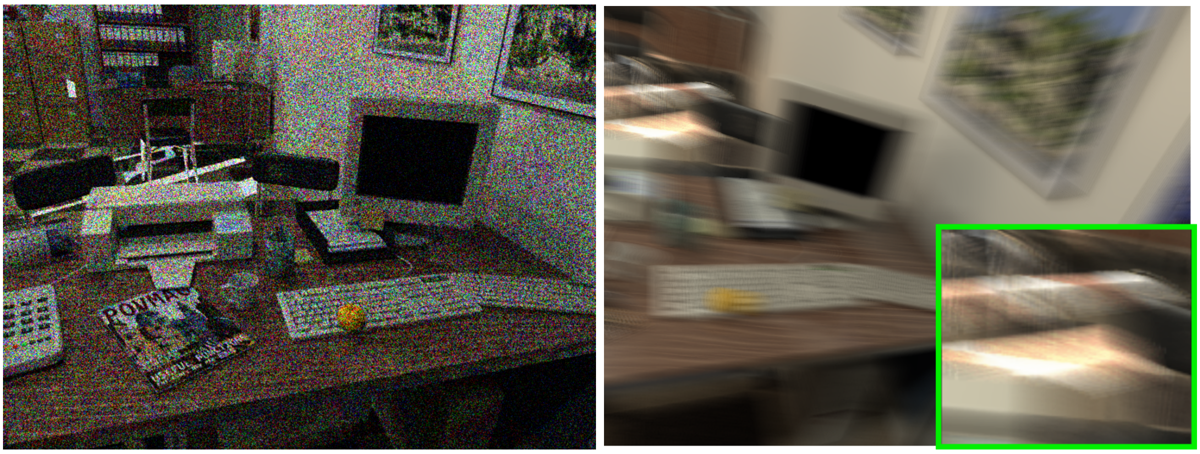


Figure 5.16: Synthetic photo-realistic images at shutter timings set to half of the maximum at a given frame-rate. Left: at 100Hz, images have very little blur but are dark and noisy. The image shown has brightness values rescaled for viewing purposes only. Right: at 20Hz, motion blur dominates. The cutout highlights our correct handling of image saturation by averaging for blur in Irradiance space.

Since we do not have absolute scale for the *Irradiance* values obtained from the CRF, we cannot define scene illumination in absolute terms. Therefore, the *Irradiance* values obtained from POVRay are normalised to obtain the corresponding brightness values. We render extra samples at the either side of the trajectory pose over a given interval which give us *Irradiance* values that can be averaged. We can also vary scene illumination in the synthetic scene and obtain images for varying degree of lighting conditions. We define a new constant

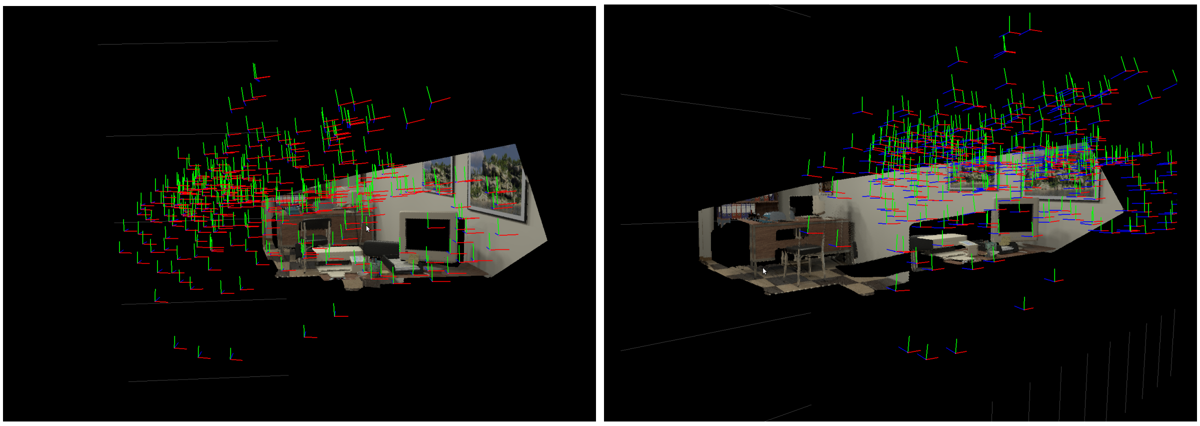


Figure 5.17: A sample trajectory displayed with a section of one of the 3D scenes used in the analysis. Red, Green and Blue denote the X,Y and Z axes respectively.

α representing overall scene brightness:

$$\mathrm{B} = f(\alpha \mathrm{E} \Delta t) \; . \tag{5.13}$$

In our experiments we have values $\alpha \in \{1, 10, 20, 40\}$, with increasing scene illumination leading to better image SNR. The reference E for each pixel is set to the base pixel value obtained from POV-Ray. Therefore, the overall process of blur generation for a given trajectory pose can be summarised by the following equation:

$$\mathrm{B}_{\mathrm{avg}} = f\left(\frac{\alpha \Delta t}{\mathrm{N}} \sum_{i=1}^{N} \mathrm{E}_i + n_s + n_c\right) \; . \tag{5.14}$$

We apply the image noise first because noise comes with incoming photons, before averaging the *Irradiance*. Finally, we quantise the obtained brightness function into an 8-bit/16-bit per channel colour image. Figure 5.16 gives examples for simlar motion at 20 and 100Hz.

5.7.3 Different Camera Response Functions

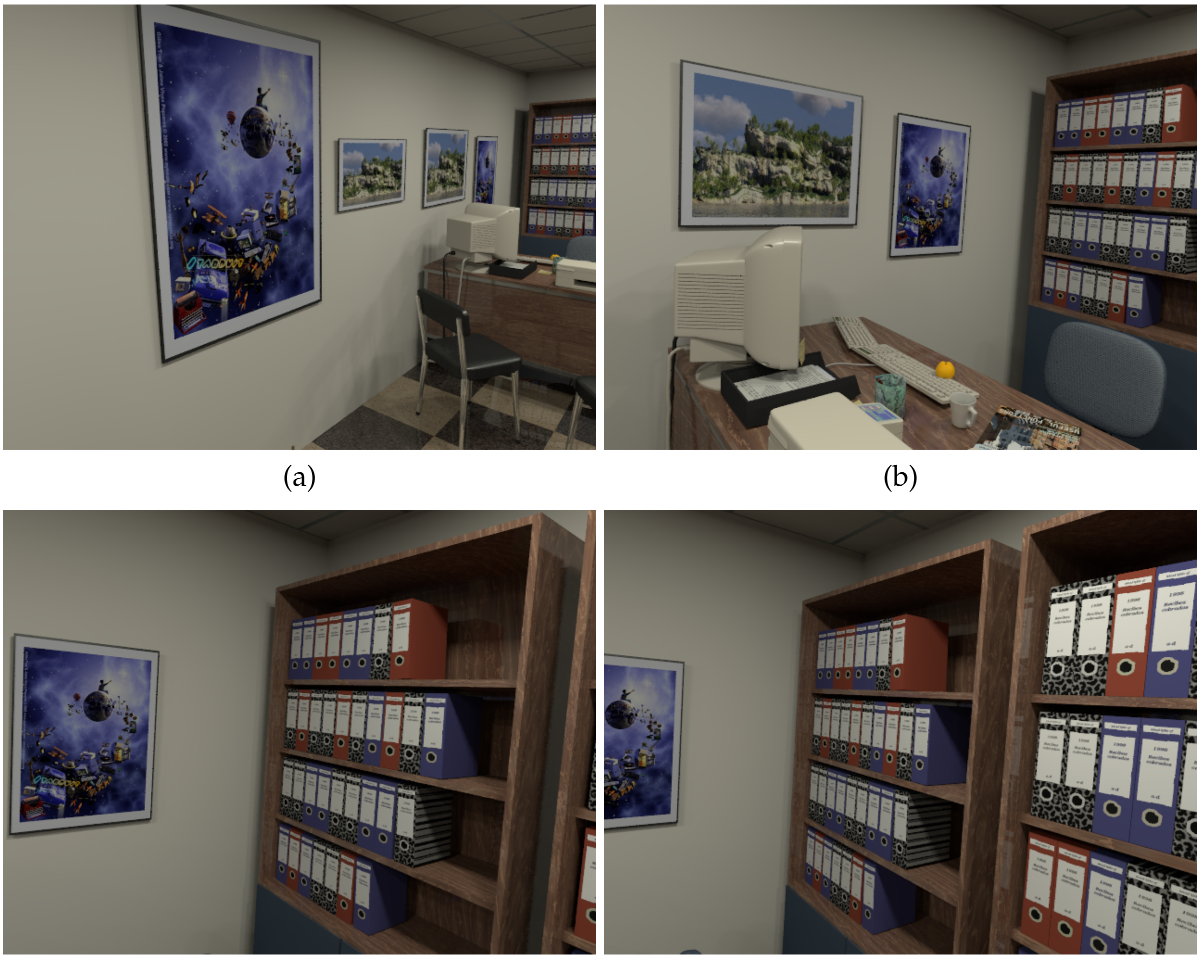
Rendering of images using different camera response functions can be obtained similarly. Figure 5.13 shows different camera response functions that are obtained from popular Columbia CRF dataset [Grossberg and Nayar, 2003].

5.8 Gathering Synthetic Data for Different Frame-Rates

Obtaining a sequence requires a camera trajectory that has all the characteristic properties of real hand-held motion; it should vary smoothly between consecutive trajectory points and should not be different from the motion a user generally creates while moving the camera. Any system that allows the capture the hand-held motion can provide a trajectory we need that can be easily ported to POVRay. Given that the trajectory obtained from the system is in the frame of reference of that system, the following equation must be used to map it to the reference frame of POVRay:

$$\mathrm{T}_{pov\_cam} = \mathrm{T}_{pov\_system} \, . \mathrm{T}_{system\_cam} \tag{5.15}$$

T_{system_cam} records the camera poses represented in the world coordinate frame of the system. T_{pov_system} is what we seek to transform the system poses to their corresponding poses that are represented in world coordinate frame of POVRay, T_{pov_cam}.



(a) (b) (c) (d)

Figure 5.18: Figures (a) and (b) show different views of the office desk while (c) and (d) view at the back of the scene.

We obtain T_{system_cam} by running DTAM [Newcombe et al., 2011b] on the hand-held camera. Dense systems like DTAM directly work on the image pixel values to obtain camera motion by aligning the incoming image to one obtained from the model. The redundancy and over-constrainedness of the data for estimation of camera motion makes DTAM relatively immune to shaky and rapid motion. Hence, it is possible to obtain camera poses for the type of motions that we would like to analyse.

The poses obtained from the system are in the world coordinate frame of the system. In order to be able to relate them to the poses that the raytracer can use, we need to transform these poses in the world coordinate frame of POVRay as described in Equation 5.15. Given the intial pose where we want the trajectory in POVRay to start from, we can map the initial pose given by DTAM and obtain the transformation T_{pov_system} via standard inverse transformation *i.e.*

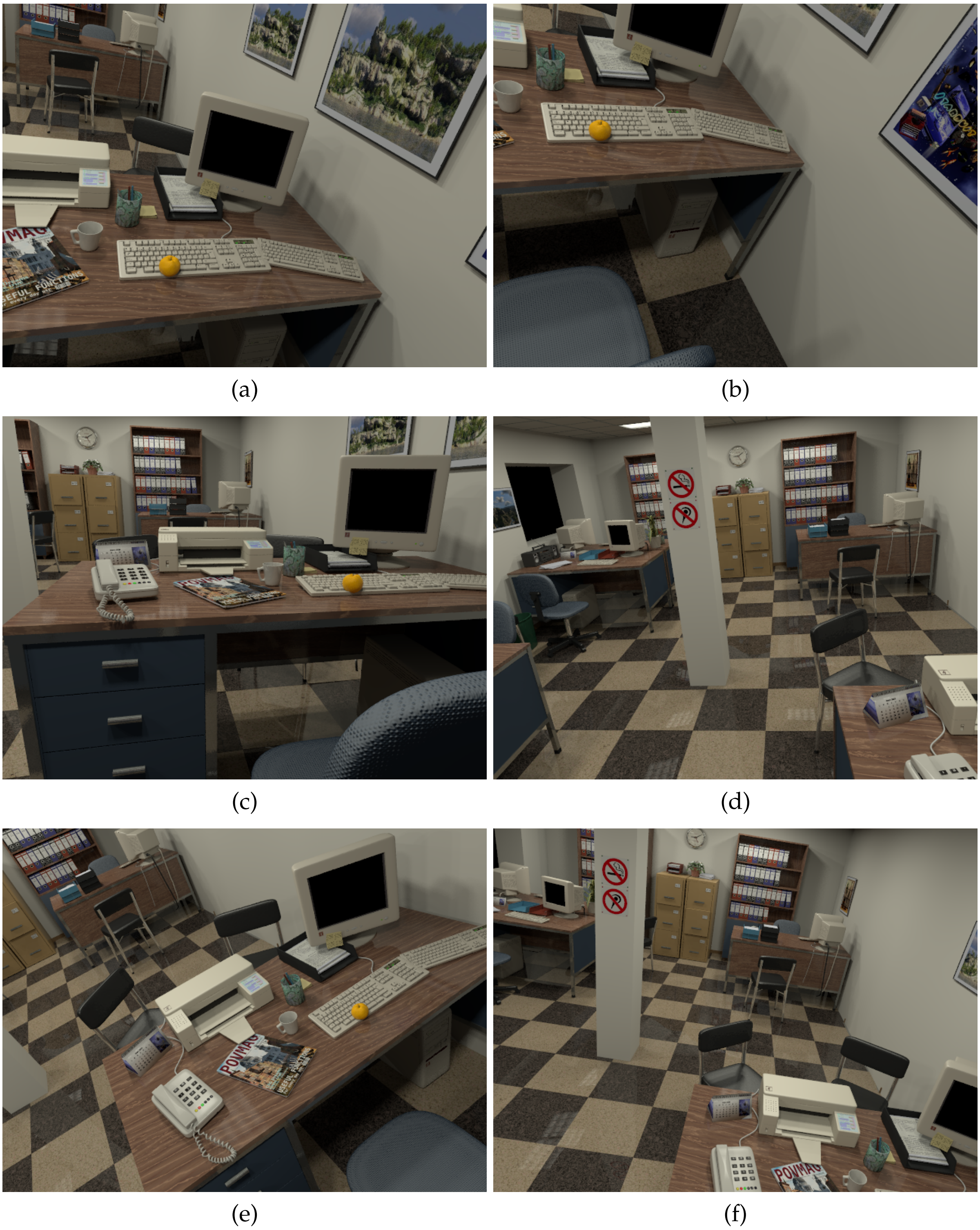
$$\mathsf{T}^{start}_{pov\_cam} \cdot \left(\mathsf{T}^{start}_{system\_cam}\right)^{-1} = \mathsf{T}_{pov\_system} \; . \tag{5.16}$$

The obtained transformation is kept fixed and other poses, T_{system_cam} , of the trajectory can be trivially mapped to their corresponding poses in the POVRay system using Equation 5.15.

Going from still images to video requires a camera trajectory for the simulated camera to follow through the scene. After initially experimenting with various methods for sythesizing trajectories and finding these unsatisfactory, we decided to capture a trajectory from a real camera tracking experiment. Using DTAM [Newcombe et al., 2011b] we tracked an extreme hand-held shaky motion which was at the limits of DTAM's state of the art tracking capability with a 30Hz camera. These poses are then transformed using a single similarity transform into the POV-Ray frame of reference to similarly render the synthetic scene. To obtain images for any frame-rate, the poses were interpolated, using cubic interpolation for translation and slerp for rotations. At our lowest experimental frame-rate of 20Hz, we see a maximum observed horizontal frame-to-frame motion of 260 pixels, nearly half of the whole 640 pixel wide image, indicating the rapidity of the motion. Figure 5.18 and 5.19 show some of the sample images we obtain in the sequence.

5.9 Do We Think these Images are Realistic?

Figure 5.20 shows how far computer graphics has matured today that it is impossibly to distinguish between a synthetically generated photo-realistic image and a photograph. To begin with, assessing the realism of a synthetically generated image requires understanding the response of the spectrum of the scene to the human visual system, *psychophysics investigation*, because it is human in the end who is going to decide *whether the image is realistic or not*, not a machine. Perceiving realism of synthetic images has been visited far back by [Meyer et al., 1986] when ray-tracing was an emerging concept and later by [Rushmeier et al., 1995] who present three different metrics inspired from disciplines of image compression to quantify the realism. They both carefully replicate the scene characteristics of a well controlled real scene in a synthetic ray-tracer. Thus, they are able to compare the images from both settings under the defined metrics and report the accuracy and realism. However, the scenes they use are quite simple with not very complex texture



(a)

(b)

(c)

(d)

(e)

(f)

Figure 5.19: Figures (a) and (b) focus on the desk showing various objects that are on the table. Reflections and shadows are clearly visible, while (c) and (d) show the front view of the desk and the view of the whole scene and (e) and (f) show images taken at various different camera poses.

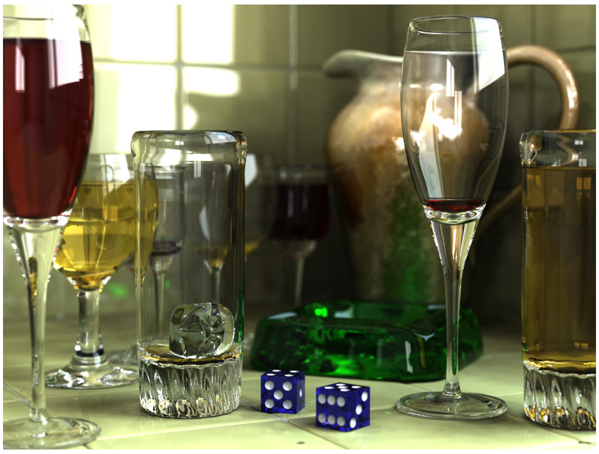


Figure 5.20: By just pure visual inspection at the image can we tell *whether the image is real or not?*

and very small number of objects and light sources. The kind of scene we want to target in our analysis is a complex and cluttered scene with varying surface properties, textures and lighting and illuminations and to be able to replicate all that in a controlled real scene is well-nigh impossible. Inverse Global Illumination [Yu et al., 1999] is another technique

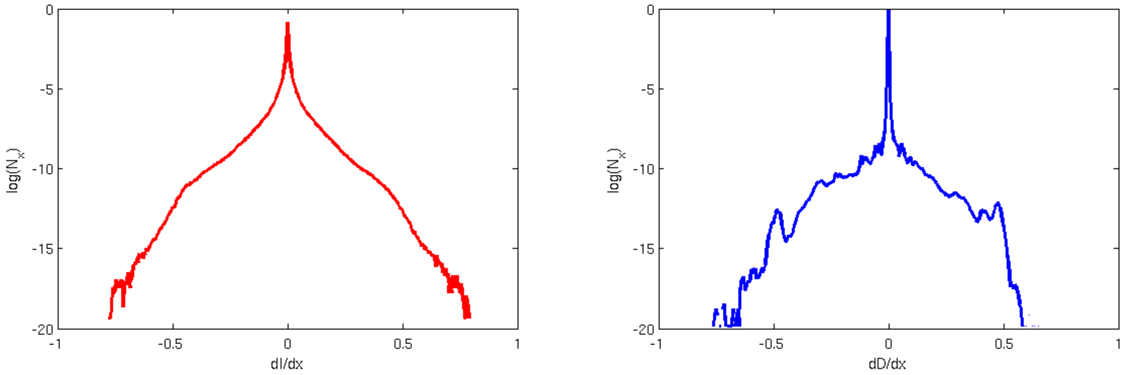


Figure 5.21: Statistics of the derivatives of the image intensities (a) and depth (b). The derivative statistics of depth are sharply peaked than the corresponding image derivatives.

that given scene geometry, the radiance information, position of light sources and known illumination recovers the diffuse and low dimensional parametrised surface reflectance of the objects in the scene. The surfaces are assumed to be have isotropic and time invariant reflectance properties. The obtained data is used in a standard ray tracer that harnesses the information of the real scene to obtain its synthetic near-replica. This also allows to insert new synthetic objects to the scene that can camouflage the light transport and appear to be a part of the scene.

Training and classification based approaches have also found their place in this analysis. [Ng and Chang, 2004] and [Lyu and Farid, 2005] present a machine learning based approach to train a classifier to later distinguish between the photographic and photo-realistic image. However, such an approach is quite labourious and therefore not always possible.

[Reinhard et al., 2001] show an interesting fact about the second order statistics of the synthetic scenes. They observe that like natural images the spectrum of the scenes also follows the power law (with the exponent near to 2) and these statistics remain invariant to various rendering transformations e.g. reflections, anti-aliasing etc. applied to the scene. It shows that geometric and modelling properties of the scene can be separately studied from rendering properties by looking at second-order statistics.

An interesting aspect of the dense alignment algorithm is that the transformations that are applied by a rendering system *e.g.* reflections and shadows to a large extent break the standard Lambertian surface assumption anyway and may not be going to be used in the estimation of camera pose. A natural question then is *Is it worthwhile spending the effort on quantifying realism?* As long as we know the images we are going to use have a tinge of realism in them, it is sufficient to treat them as photo-realistic images. Therefore, while rendering the images, we also conducted a regular passive experiment asking users to qualify the image as *real* or *unreal*. The experiment is passive because users were not told beforehand that the images were synthetic. We found in most of the cases the photo-realistic effects in the scene elicited similar response to the human visual system as a regular photograph duping the users to believe that the image was *real*. Similar passive experiment has also been done by [Rademacher et al., 2001] in their study of measuring the perception of visual realism in images. We found that to be a satisfactory criteria to assess the realism in images.

5.10 Novelties of Our Dataset

Our multi frame-rate dataset significantly differs from many existing datasets that aim to provide ground truth for evaluation of various standard algorithms in computer vision. We outline the differences below:

Longer sequences Many existing datasets *e.g.* Middlebury dataset for evaluation of optical flow [Baker et al., 2011] and stereo [Scharstein and Szeliski, 2001] contain only a handful of image pairs. However, as [Geiger et al., 2012] and [Butler et al., 2012] also point out, this often leads to biased evaluation and overfitting to a small set of images. A longer sequence is required then to properly evaluate an algorithm for different measurements of the same scene. We render 5 second videos for different frame-rates. This means 1000 images captured at 200Hz and 100 for 20Hz.

Similar opinions have also come from TUM-RGBD[8] dataset [Sturm et al., 2012], KITTI[9] dataset [Geiger et al., 2012] and SINTEL[10] dataset [Butler et al., 2012] who collect indoor and outdoor sequences respectively with an aim towards evaluation of an algorithm over a trajectory. However, both these datasets contain real-imagery and their ground truth is obtained from a better quality sensor and as a result it is not exactly 100% ground truth. On the other hand, our dataset has exact ground truth obtained from synthetic framework but special care is taken to ensure that the images are as photo-realistic as possible. The closest dataset to ours is the dataset of TSUKUBA [Peris et al., 2012] and [Martull et al., 2012] where they create a synthetic scene[11] built on real-world scene features and materials to enable its photo-realistic version on a computer using CG rendering.

Careful replication of real-world camera settings Although datasets [Peris et al., 2012] and [Martull et al., 2012] have replicated material properties from the real world scenes, they do not replicate the camera settings *e.g.* motion blur and camera noise that greatly affect the camera image capture. Our dataset has modelled it carefully by calibrating the camera response function of the camera. Moreover, our dataset also provides images for different lighting conditions like [Martull et al., 2012].

Data collection for different frame-rates Given our focus on camera tracking at high frame-rates, our dataset is the first of its kind that has images obtained on the same trajectory for different frame-rates.

The availability of datasets with accurate ground truth greatly assists in evaluation of algorithms as has popularly been done by Middlebury datasets for optical flow and stereo estimation. However, the overall goal of the datasets is to enable fair judgement and comparison of an algorithm and not lead to over-fitting. Therefore, it is necessary that the datasets

[8] http://vision.in.tum.de/data/datasets/rgbd-dataset

[9] www.cvlibs.net/datasets/kitti

[10] http://sintel.is.tue.mpg.de/

[11] A webpage for this dataset is set up here at http://cvlab-home.blogspot.jp/2012/05/h2fecha-2581457116665894170-displaynone.html

contain a variety of difficult cases that appear in real world scenes. Our dataset focusses on these issues and we add lot of real world artefacts that break the general assumptions used by algorithms.

Chapter 6

Tracking Analysis: Synthetic Experiments

Contents

A part of the work described in this chapter led to the following publication:

Ankur Handa, Richard A. Newcombe, Adrien Angeli, Andrew J. Davison (2012). Real-Time Camera Tracking: When Is High Frame-Rate Best? *In Proceedings of the IEEE European Conference on Computer Vision (ECCV).*

6.1 An Experimental Evaluation of Dense 3D Tracking

Our primary interest in this analysis is to understand the implications of using high frame-rate images on the performance of 6DoF camera tracking from a known 3D model. Although there are natural and obvious benefits of increasing frame-rate — the reduction in baseline means that tracking should be much easier and robust but *is it affordable to work on high frame-rates that put an increasingly stringent demands on the processing available?* This is something we would like to know.

Moreover, the camera tracking we are interested in, is a function of the dense 3D structure, the scene texture and the scene lighting. Putting all these things together and varying them independently on each axis provides for another route to test the suitability of given frame-rate under user provided computational demands. However, we have only considered varying frame-rate and computational budget and therefore via this analysis we would like to be able to say that a frame-rate operation is only possible when a given processor is available. Alternatively, a given processor can only afford to allow tracking to run at a given frame-rate.

Our understanding of the standard pipeline of the camera tracking algorithm reveals another parameter, image resolution, that also contributes to our final conclusions. This is obviously an independent parameter of the analysis that does not depend on the frame-rate but given the typical coarse-to-fine pyramid strategy used in the tracking, different image resolutions offer a different level of accuracy. Bringing image resolution in the analysis provides even more insights and that we can have different cross overs e.g. *a combination of low frame-rate but higher resolution can provide more accuracy than working on a combination of low resolution and high frame-rate at a given computational budget*. In the standard coarse-to-fine pyramid structure a divide-by-2 reduction ratio is used and that increasing the image resolution to the next level quadruples the number of pixels. This increase in the number of pixels is likely to provide more robustness to the tracker as well as offer more information about the camera pose till a certain point. It is important to remember that there are other possible choices of subsampling instead of divide-by-2 reduction ratio. One may want to increase the reduction-ratio to preserve more information content in the image at a given pyramid level. However, this comes at an expense of increased number of pyramid levels which greatly affects the real-time operatability of the tracker. Therefore, divide-by-2 is a fair balance of preserving image information at lower pyramid as well as allowing a real-time tracking operation.

In our experiments, we have analysed data where the motions are such that gross tracking failure is rare. Further, we also consider a full investigation of the robustness of tracking which requires orders of magnitude more data so that meaningful statistics on tracking failures can be obtained. We have revisited this issue in our real experiments, because one would guess that one of the main advantages of high frame-rate is improved *per second* robustness. We therefore focus here on two measures in our synthetic results: accuracy during normal tracking, and computational cost. Our main results are in the form of bi-objective plots, with Pareto Fronts suggesting application-dependent operating points a user might choose.

Taking into account the full image formation process, there are many other parameters that can be varied – camera aperture controls the light entering in the sensor, camera focus controls the depth-of-field of image, the point spread function of the camera controls the sharpness of the pixel, the pixel size of the sensor controls again the light as well as the camera noise, the quantum efficiency of the sensor controls the SNR of the image, other camera artefacts like vignetting, blooming and dithering also play a significant role in the real image formation. However, we want to clarify that we have not analysed the effect of these parameters in our experiments since they do not arise from the change of frame-rate or shutter time of the camera. They remain virtually constant for all settings of frame-rate as they relate to either the manufacturing of camera or lens. Given that we analyse the effect of frame-rate with respect to the changing computational cost, these parameters are insignificant.

6.1.1 Highlights of the Results

Most significantly, our experiments have highlighted the extreme importance of proper consideration of scene lighting conditions in evaluating tracking; even in a sophisticated and expensive experimental set-up with motion capture or a robot to track camera motion, and laser scanning to capture scene geometry, controlled lighting and light measurement apparatus would also be needed to ensure repeatability across different camera settings. As described in the previous chapter, we have gone to extreme lengths in endeavouring to produce experimental video which is not just as photo-realistic as possible, but is based on parameters of a real camera and motion and realistic typical 3D scene. This is highlighted in the samples from our dataset showing the effect of changing frame-rate leading to varying brightness levels and noise in the images. The effects of scene lighting are largely neglected in most tracking algorithms which perhaps is of significant importance in this analysis because it is something that directly affects the images obtained at different frame-rates. Also, we study the effects of lighting, baseline and motion blur independently where these three are tightly entangled in the real world scenarios and perhaps inseparable.

6.2 How do We Interpret the Graphs?

Given the current parameters of our tracker (the frame-rate, the resolution and cost), we would locate the corresponding operating points on the Pareto Front (provided scene lighting conditions we choose the relevant Pareto Front), and then given the increased computational budget, we would slide along the Front in the style of looking up values in a look-up table or chart of operating points, to find out our next best possible choice of frame-rate and resolution for a given camera motion and camera response function model. We would like to mention that we have only provided details of the results pertaining to a linear camera response function. However, the results will generalise for different camera response function models.

6.3 Assumptions

We would like to review the assumptions that we are using in our experiments to clarify the settings under which our analysis would hold sensible.

1. We are tracking from a known rigid 3D surface model with texture. This is assumed to have been obtained by running a depth-estimation multi-view stereo based approach that provides the geometry and the texture. However, we have assumed that geometry remains unchanged while texture that is obtained from the images changes as the frame-rate changes — different frame-rates provide different contrasts of textures obtained. Important thing to remember is that when we increase the frame-rate, matching operates on an observed image obtained from the camera which is noisy while the prediction comes from the 3D model which provides a texture that has the noise averaged out. For low frame-rates we pre-blur the prediction with *a priori* knowledge of the camera motion from the previous frame to obtain a texture which reflects the amount of blur observed in that frame and match against current observation that has blur depending on the how rapid the current motion is. Consequently, we blur the depth-map too.

2. Although the resolution and frame-rate are limited by the bandwidth of the bus that is being used to transfer the data from camera to the desktop (or other) hardware, we assume in our analysis that the bandwidth of bus is unlimited and that it allows the camera to give the highest resolution at the highest allowable frame-rate we experi-

ment with. However, in real-world conditions we would have limited bandwidth and this would mean if the Pareto Front shows a list of choices that can not be afforded due to limited bandwidth, we can still find out the optimal choice of parameters by precluding the Fronts beyond the limits of the camera being used.

3. The highest resolution used in our tracking is the standard 640×480. Although we have used this as the highest resolution, we believe our results generalise well with increasing resolutions provided the camera motion is such that working on higher resolutions still furnishes considerable gains in accuracy. One would guess that at some stage increasing resolution beyond a limit will only provide diminishing returns.

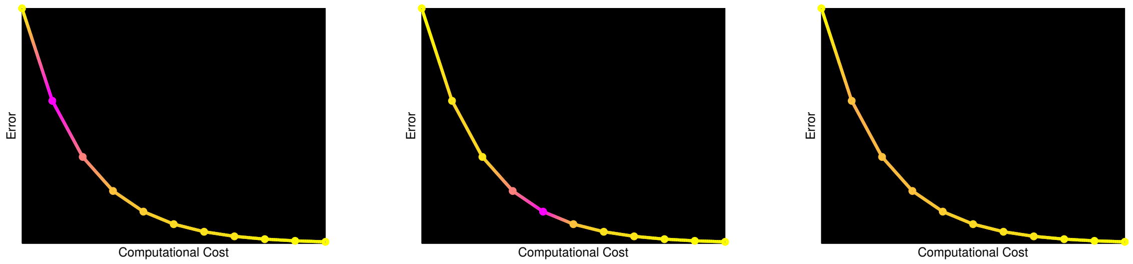
4. We have also assumed that the time-lag between current frame and the next frame is of no importance to our analysis and that the images are ready available on the hard disk of the computer.

6.4 Characterisation of Experiments

We characterise our experiments into different motions and the scene lighting. The image motion obtained is a direct function of the sampling rate at which images are captured and as a result affects the performance of tracker — faster motion would need higher frame-rate to allow more robust tracking while it would be easy to track even at low frame-rates for slow motion. Scene lighting on the other hand, affects the contrast and signal-to-noise (SNR) of the image. Tracking suffers when images are relatively darker and noisier which is quite often the case when working in low-lighting conditions.

6.4.1 Camera Motion

Our motion characterisation allows to examine the performance of a tracker at different motions. It is obvious that for faster motion, we need higher frame-rate to be able to track robustly. However, whether it is possible to work at higher frame-rates under the desired computational budget, this is what we examine. Our experiments are divided into fast and slow motions to study the effects independently. Moreover, the kind of motion the camera is undergoing also affects the choices. For instance, rotation tends to induce fast and rapid motion in the image and is independent of the scene depth. Therefore, we would like to know how pure-rotation affects the optimal choices. On the other hand the motion induced by forward and backward translation again is very much dependent on the scene depth when compared to sideways and lateral translation. The effect of different motions is what we study.



(a) Slow motion (b) Moderately-faster motion (c) Fast motion

Figure 6.1: We expect that for different dynamics optimal frame-rates should be high for very low computational power, making a transition towards somewhat lower values with a further increase in the budget and start increasing as more computation power is thrown at the algorithm. This is shown in the colour-coded curves where magenta colour denotes a hypothetical lower frame-rate that is expected to appear very early when the dynamics is slow and later when the dynamics is moderately faster and disappear in the curve when the motion dynamics is fast.

What results do we expect? Our intuition tells us that for slow motions even low frame-rates would provide very similar accuracy figures as we would get using high frame-rates. Therefore, it would make more sense to work on low frame-rate images for small computational budgets and switch to higher frame-rates to optimally use the budget to polish accuracy figures. As motion becomes faster, we would expect that low frame-rates would require more iterations to converge compared to high frame-rates. Therefore, low frame-rates would slide down where it is possible to operate only when computational budget is slightly increased. However, a further increase would again lead to high frame-rates as the natural choice for optimal budget usage. Lastly, tracking fast motion is possibly only for very high frame-rates and that low frame-rates disappear from the graph completely. This is due to the fact that tracking breaks the linearsation assumptions at low frame-rates.

This is very much explained graphically in Figure 6.1 where a hypothetical low frame-rate appears in the graph very early for slow motion and slides down as the motion becomes faster and disappears completely from the graph as motion becomes really fast.

6.4.2 Scene Lighting

The scene lighting directly affects the pixel intensities observed in the image. Our scene lighting characterisation clarifies the settings in which the tracking is operating. We denote a scalar variable α that quantifies the light in the scene.

Scene Lighting	**Characterisation**
$\alpha = 1$	Low Lighting.
$\alpha = 10 - 20$	Moderate Lighting.
$\alpha = 40$	High Lighting.
$\alpha = \infty$	Perfect Lighting.

What results do we expect? It is important to remember that scene lighting affects the SNR of the image which in turn affects the performance of the tracking algorithm. Particularly, at high frame-rates where exposure time is already very small, low scene lighting conditions mean that images obtained are darker and noisier. We, therefore, expect the tracking performance to suffer at high frame-rates more due to scene lighting conditions. On the other hand, low frame-rates due to longer shutter times manage to collect enough photons for good SNR image. For high lighting conditions, we expect high frame-rates to allow tracking to resume as normal while low frame-rates to suffer from saturation (if using very low frame-rates).

6.5 Tracking Analysis and Results

We used our framework to synthesize video at 10 different frame-rates in the range 20–200Hz using a five second rapid camera motion. To push frame-rate further we also synthesized 400Hz and 800Hz sequences. We present results from these sequences and concentrate on clarity of interpretation.

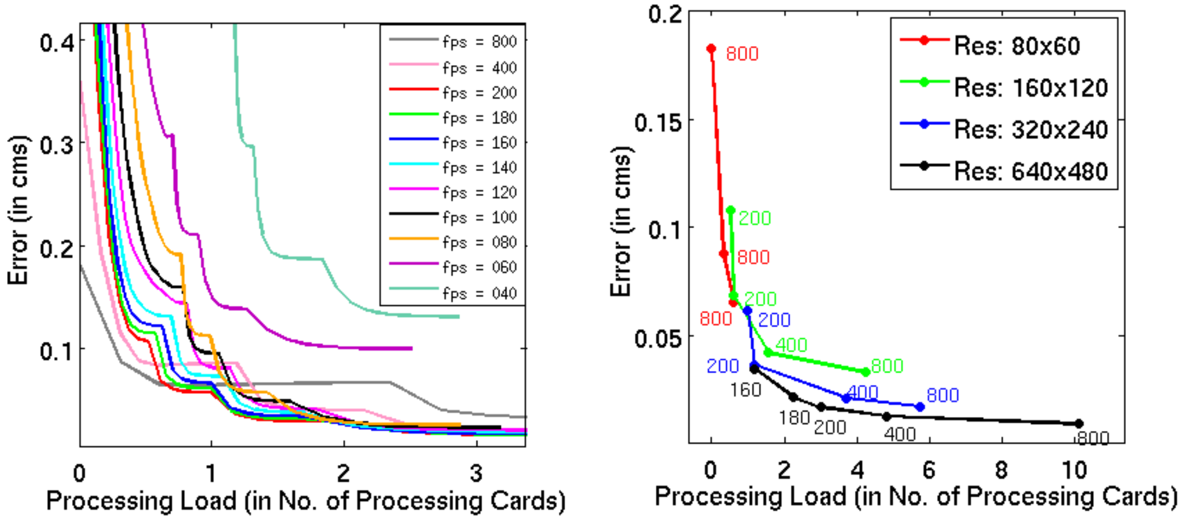


Figure 6.2: Left: error plots for different frame-rates as a function of available computational budget under perfect lighting conditions. Points of high curvature denote the switch from one pyramid level to the next. Right: Pareto front for minimum error/minimum processing load performance, highlighting with numbers the frame-rates that are optimal for each available budget.

6.5.1 Fast Handheld Motion

Experiments Assuming Perfect Lighting

Our first set of experiments assumes that whatever the frame-rate it is possible to assume blur- and noise-free images from the camera, and uses ‘pure’ ray-traced images. This assumption is most closely represented in reality in an *extremely* well-lit scene, with shutter time set to a very low constant value at all frame-rates while still capturing enough light for good SNR. Importantly, it lets us decouple the effects of scene lighting from baseline which are somewhat entangled given the choice of frame-rate we are using in the real world scenes.

Figure 6.2 shows the clean results we obtain in this case. In Figure 6.2(a) we plot how for each frame-rate setting, the average tracking error (our measure of accuracy) is reduced as more computation is thrown at the pyramidal LK optimisation per frame, permitting more iterations. Remember that the unit of computation we use is processor occupancy for real-time operation, obtained by multiplying computation time per frame by frame-rate. The sudden gradient changes in each curve are when the pyramidal optimisation switches from one level to the next — from low to high image resolution.

At the bottom of these overlapping curves is the interesting region of possible operating points where we can obtain the smallest average error as a function of processing load for our tracker, forming a Pareto Front. We display this more clearly in Figure 6.2(b), where we have rescaled the graph to focus on the interesting regions where crossovers occur, and used different lines coloured according to the maximum pyramid level resolution needed for that result (i.e. when we talk about resolution, we mean pyramidal optimisation using all resolutions of that value and below), and frame-rates are indicated as numbers. The behaviour of the Pareto Front is that generally very high frame-rates (200+) are optimal for very low computational budget. As the computational budget is increased, a switch to a higher resolution but a lower frame-rate is the optimal choice. This occurs at the crossovers and similar trend continues as computational budget is further increased *i.e.* a switch towards a lower frame-rate but higher resolution is the optimal choice. However, best accuracy is achieved under a very high computational budget working at high frame-rates of the order of 800Hz.

Experiments with Realistic Lighting Settings

We now extend our experiments to incorporate the shutter time-dependent noise and blur artifacts modelled in previous chapter that will affect most real world lighting conditions. We present a set of results for various global lighting levels.

We have used the same main frame-rate range of 20–200Hz and the same five second motion used with perfect images. We needed to make a choice about shutter time for each frame-rate, and while we have made some initial experiments (not presented here) on optimisation of shutter time, in these results we choose always half of the inverse of each frame-rate. To generate each synthesized video frame, we rendered multiple ray-traces for blur averaging from interpolated camera poses always 1.25ms part over the shutter time chosen; so to generate the 20Hz sequence with 25ms shutter time we needed to render $5 \times 20 \times 20 = 2000$ ray-traced frames. In fact this number is the same for every frame-rate (higher frame-rate countered by a reduced number of frames averaged for blurring), leading the the total of 20000 renders needed (with some duplicates).

An important detail is that dense matching is improved by pre-blurring the re-projected model template to match the level expected by the shutter time used so we implemented this; and the depth map is also pre-blurred by the same amount. We conducted some initial characterisation experiments to confirm aspects of the correct functioning of our photo-realistic synthetic framework including this template blurring (Figure 6.3).

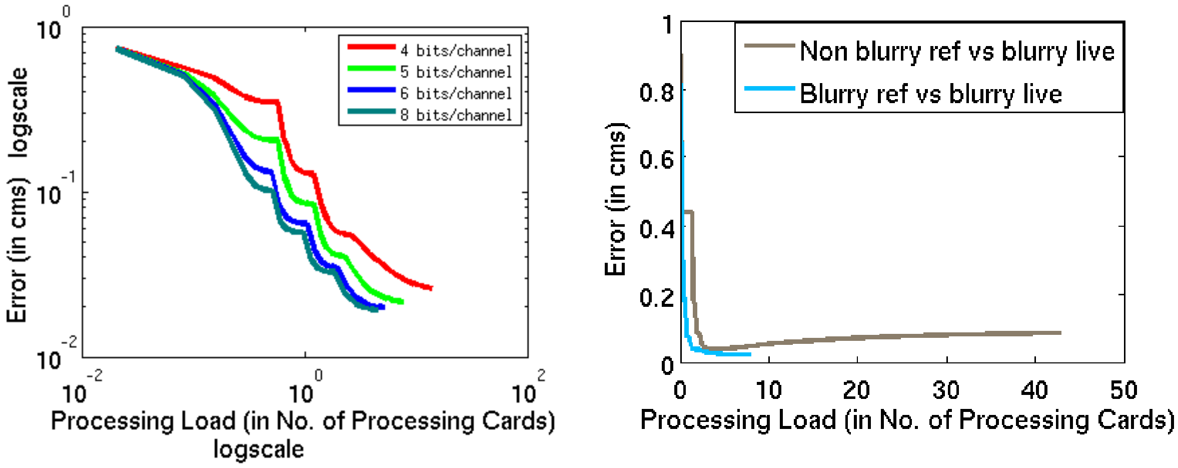
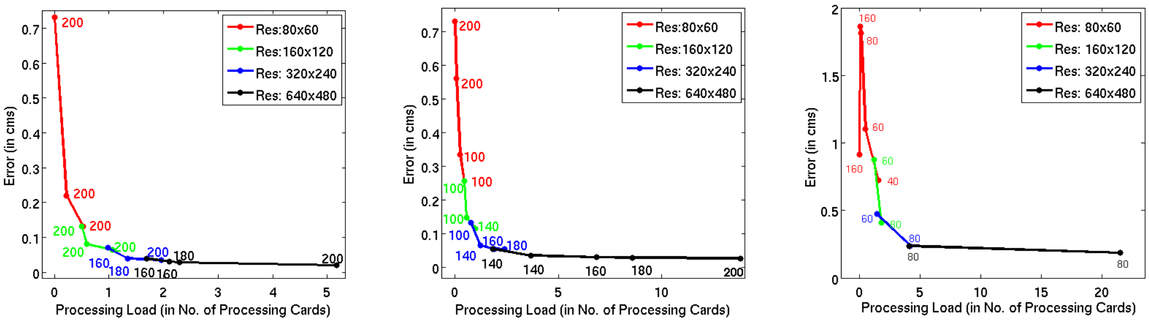


Figure 6.3: Characterisation experiments to confirm our photorealistic synthetic video. Left: For a single motion and 200Hz frame-rate, we varied only shutter time to confirm that reduced light indeed leads to lower SNR and worse tracking performance. Here SNR is directly quantified in bits per colour channel. Right: An experiment explicitly showing that quality of tracking improves if a deliberately blurred prediction is matched against the blurry live image.

High Lighting Figure 6.4 (a) shows the Pareto Front for α=40, where the scene light is high but not perfect. Images have good SNR, but high frame-rate images are darker; the 200Hz image is nearly 5 times darker than the corresponding image for perfect lighting and as frame-rate is pushed the images get darker and noisier. For clarity, we have omitted 400Hz and 800Hz in the graph. We observe that 200Hz at low resolution remains the best choice for low budgets, the image gradient information still strong enough to guide matching. And again a few iterations at 200Hz are enough because the baseline is short enough to aid accurate matching.

A further increase in budget reveals that 160Hz becomes the best choice for a higher resolution i.e. 320×240. This is where the error plots for frame-rates cross and better results are obtained by increasing resolution rather than frame-rate. As the processing load is further increased higher frame-rates are preferred, and the pattern repeats at 640×480. The plots however suggest a later transition to this highest resolution compared to the perfect



(a) α=40 (b) α=10 (c) α=1

Figure 6.4: Pareto Fronts for lighting levels (a) α = 40, (b) α = 10 and (c) α = 1. Numbers on the curves show the frame-rates that can be used to achieve the desired error with a given computational budget. The graph at α = 40 shows if the current processing card allows for 160Hz at 320×240, doubling the processing budget leads to suggesting the use of 160Hz but at 640×480 instead of increasing the frame-rate. We have observed that from α = 40 down to just α > 10, there is no change in the optimal frame-rates and as a result the Pareto Front remains the same suggesting that there is a range of lighting under which the choice of optimal frame-rates remains unchanged. As the scene lighting is decreased to α = 10, we observe that the optimal frame-rates are slightly lower than when α = 40. Further decrease in the lighting to α = 1 leads to even lower optimal frame-rates. The increase in the error at 160Hz for α = 1 with increasing budget is due to the distraction of tracker towards a different minima. Interesting observation that we make here is that an increase in resolution for any scene lighting conditions shows that the lowest optimal frame-rate for that resolution is lower than the highest optimal frame-rate at the resolution immediately lower than that before the cross-overs occur. For instance, α = 40 shows that 200Hz is the highest optimal frame-rate at 160×120 and the immediate next resolution shows that 140Hz is the lowest optimal frame-rate at 320×240 while α = 10 shows the highest optimal frame-rate at 320×240 is 160Hz while the lowest optimal frame-rate for 640×480 is 140Hz. This tells us that *similar levels of accuracy can be attained by working on a lower frame-rate but at higher resolution.*

sequence results in Figure 6.2 (b). The highest resolutions clearly have more benefit when SNR is high.

Moderate Lighting In Figure 6.4(b) we reduce α to 10, representative of a typical indoor lighting setting. 200Hz is still the best choice at very low processing load when predictions are very strong and only one or two alignment iterations are sufficient. A slight increase in processing load sees the best choice of frame-rate shifting towards 100Hz even without resolution change, in contrast to both perfect lighting and high lighting conditions where working with very low processing load demands a high frame-rate. In moderate lighting, it is better to do more iterations at a lower frame-rate and take advantage of improved SNR because increased noise at high frame-rate means that more iterations are required to converge. When we do shift to 160×120 resolution, we see that the best frame-rate has shifted to 100–140Hz compared to 200Hz in high lighting conditions. Higher resolutions, 320×240 and 640×480, follow similar trends.

Low Lighting Figure 6.4 (c) shows similar curves for scene illumination $\alpha = 1$, where images are now extremely dark. Our main observation here is that even at substantial processing load the Pareto Front does not feature frame-rates beyond 80Hz. These images have such low SNR that at high frame-rate essentially all tracking information has been destroyed. The overall quality of tracking results here is much lower (the error curve is much higher up the scale) as we would expect.

Conclusions obtained from tracking fast motion

- The error obtained at zero computational burden is the average inter-frame distance between two successive frames. This is obtained at the highest possible frame-rate used and is the error when no tracking is performed. This is therefore, the maximum error from where the tracking begins and it reduces as iterations progress.

- There are no early cross overs for high lighting conditions at lowest pyramid and that highest frame-rate has a natural advantage of tracking, both in terms of accuracy and computational cost because of short baseline. We start observing early cross-overs as scene lighting degrades, however, the optimal frame-rates are still in the range above the standard 30Hz that is most commonly used in real-time tracking. Therefore, under stringent budgeting conditions, a combination of low resolution and high frame-rate is the optimal choice.

- As the computational budget is increased, an emerging pattern is that, a switch to a higher resolution and lower frame-rate if possible is observed more often than just increasing frame-rate alone suggesting that increasing spatial resolution provides more gains than temporal resolution. However, for infinite processing budget, the best choice is a combination of the highest frame-rate and the highest resolution, a camera can afford.

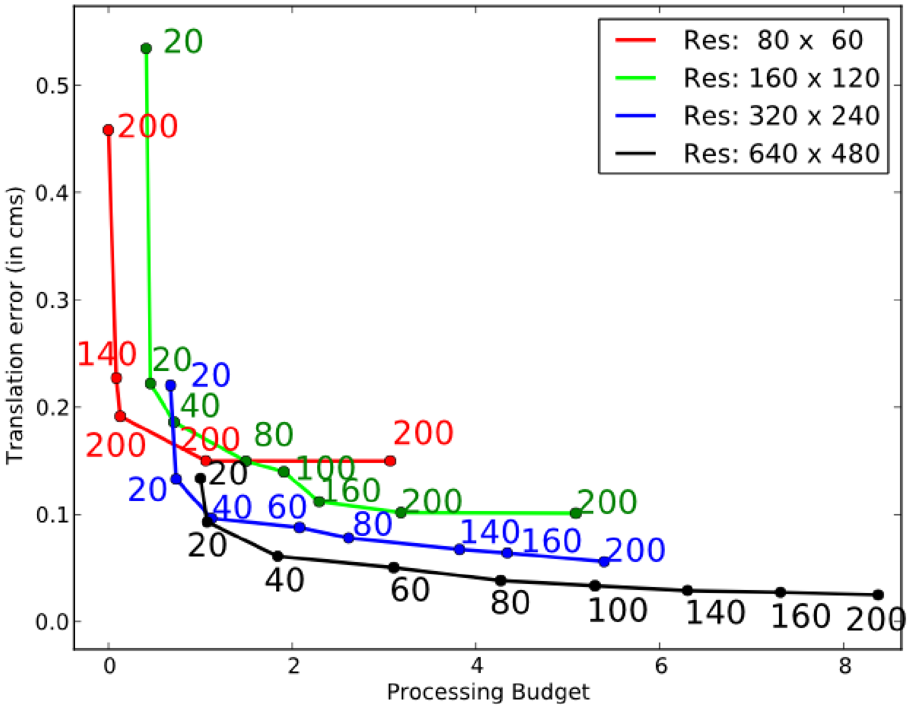
- As scene lighting conditions degrade, the Pareto Fronts show similar trends but the optimal frame-rates move to lower values to compensate for the high contrast required to achieve accurate results.

- The results highlight quick transitions towards higher resolutions for more accuracy, so a natural question would be: *should we keep on increasing the resolution?* The short answer is no, because after a certain resolution, we would see the gains in accuracy only saturating something we observe in our pure rotation case, suggesting that there is a limit to optimal resolution for a given camera motion.

6.5.2 Slow Handheld Motion

Slow motion tends to induce relatively less blur in the images provided the scene being imaged is relatively distant from the camera. Our second sequence involves a handheld motion of camera moving away and into a distant scene. Images pertaining to low frame-rates e.g. 20Hz are therefore free from image blur. This makes our analysis depend only on the image baseline and scene lighting.

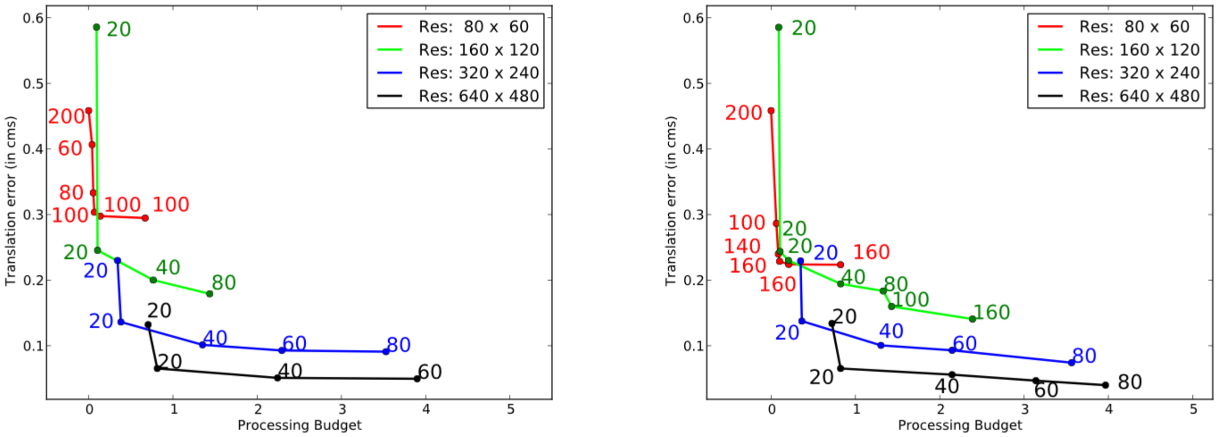
Perfect lighting conditions The results obtained with perfect lighting assumptions reveal a similar trend (Figure 6.5) *i.e.* use high frame-rates (140–200Hz) under a very stringent computational budget. They correspond to using first few iterations on the lowest resolution. The small baseline of the high frame-rates gives a natural advantage over low frame-rates that using first few iterations provides the best error reduction — even if the cost *per second* is smaller for low frame-rates, the error is still relatively larger. As the computational budget is increased, we see a progression towards switching to a higher resolution and lower frame-rate. The slow motion in the trajectory immediately brings the lowest operatable frame-rate down to 20Hz. The computational demands to work on a combination of high frame-rate and higher resolution swamp the achievable reduction in error, thereby bringing a lower frame-rate range in the operatable regime. As more computational power is thrown, it is no surprise to see a higher frame-rate and higher resolution together providing the best error reduction. Noteworthy is that there are early cross overs e.g. 140Hz at the lowest pyramid level. This is in contrast to conclusions obtained in the fast motion, where the level of accuracy at high frame-rate always leads the accuracy that can be obtained from lower frame-rates.



Pure Ray-traced

Figure 6.5: Pareto fronts for Pure Ray-traced images. The range of operating frame-rates increases to 140–200Hz when compared with $\alpha = 20$ and $\alpha = 10$ for resolutions of 80 × 60 when the computational budget is very low. However, the subsequent image resolutions see a similar operating range of 20–40Hz nearly to processind budget less than 2 processors. Importantly, frame-rates from 100–200Hz find their operatability at the highest resolution which could not be achieved under low lighting conditions for $\alpha = 10, 20$ due to the fact that highest resolutions were the noisiest among all resolutions. Something more that this graphs tells us is that the Pareto Front for resolution 160 × 120 is lies inside the Pareto envelope of other resolutions – this may be attributed to the slow motion the trajectory has that leads to very small gains on switching the resolution from 80 × 60 to 160 × 120 while further increase in the resolutions lead to 16 and 64 times increase in the number of pixels that are very informative of the camera motion.

High lighting conditions Our high lighting conditions here correspond to $\alpha = 40$. We see a change in the operatable frame-rates at lowest resolution, 140–200Hz range observed in perfect lighting conditions has moved down to 100–160Hz range suggesting the importance of high SNR image in camera tracking. Further resolution switches follow similar trend towards using a lower frame-rate but we also observe that frame-rates 100–200Hz do not appear in the Pareto Fronts and that 80Hz is the best operatable frame-rate for this scene



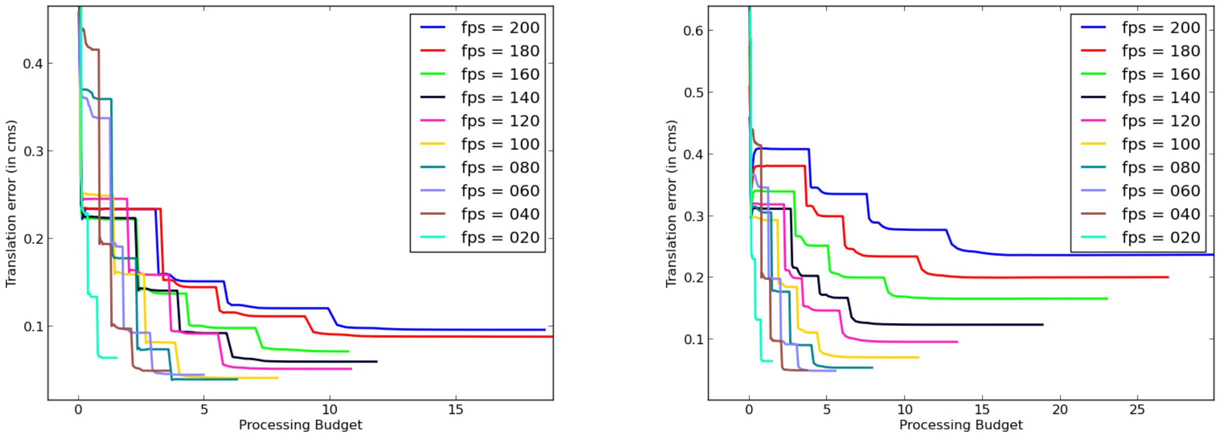
(a) $\alpha = 10$

(b) $\alpha = 40$

Figure 6.6: Parto Fronts for (a) $\alpha = 10$, and (b) $\alpha = 40$. Optimal frame-rates for different computational budgets are shown with image resolution. For extremely low computational demands, it is natural to see high frame-rates 100–160Hz appearing as optimal as they correspond to using single iteration of the optimisation outlining that lower frame-rates take more iterations to achieve similar or more error reduction. However, increasing the resolution even by a factor of four shows a remarkable change in the optimal frame-rate range – 20Hz immediaely becames optimal and this is again due to many factors, the brightness levels, reduction in image noise and slow trajectory motion. Interesting also to note is that 60Hz at 640×480 needs a processing budget of nearly 4 processors when $\alpha = 10$ to achieve similar error that can be achieved by working on the same frame-rate but requiring less processing budget when $\alpha = 40$. A higher frame-rate of 80Hz (and offering more robust tracking) can achieve smaller error for the same processing budget when lighting is high. Our current processor allows to work on highest resolution and therefore, a combination of 20Hz and 640×480 is possible. Again, we notice that the gains obtained by switching to a higher resolution are more than switching to a higher frame-rate clearly highlighting the importance of number of pixels.

and camera motion. This is explained by the Figure 6.7 (a) where we see the error after convergence at the highest resolution for frame-rates 100–200Hz progressively increases due to image noise and bit quantisation present in the images.

Moderate lighting conditions Moderate lighting conditions refer to $\alpha = 10$ here in Figure 6.6(a). The Pareto Fronts show very similar conclusions as high lighting conditions. The frame-rate range for lowest resolution is further moved to 60–100Hz. While the higher resolutions still maintain the 20Hz as the optimal operatable frame-rate we see that 80Hz which is still in shown in the Pareto Front corresponding to 320×240 resolution, disappears



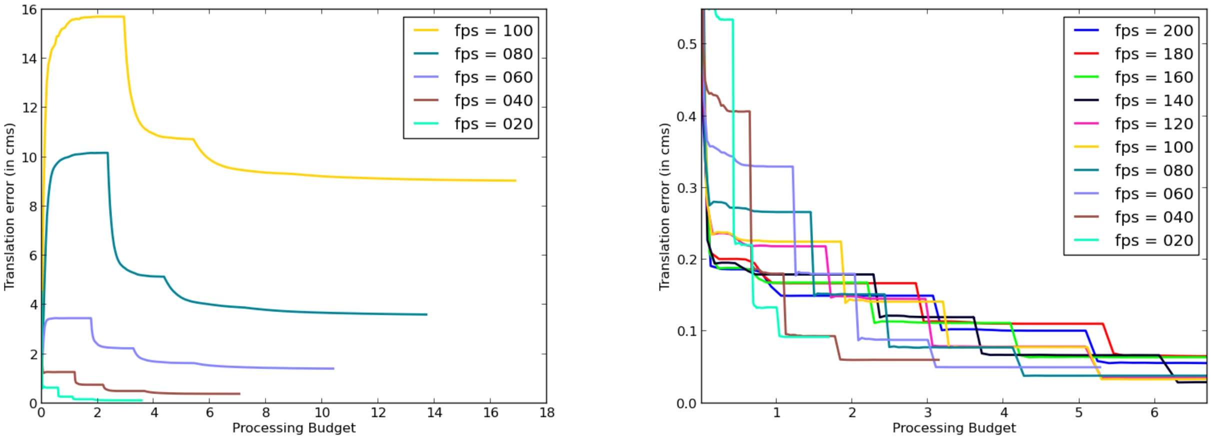
(a) α=40

(b) α=10

Figure 6.7: Error curves for (a) $\alpha = 40$, and (b) $\alpha = 10$. Plots show significant increase in the error as frame-rate is increased. This is largely due to the decrease in the brightness levels as well as the SNR of image. The slow motion in the trajectory also enables low frame-rates to obtain similar levels of accuracy as high frame-rates. Increase in brightness levels and noise reduction at low frame-rates therefore offer a natural advangate over working at high frame-rates. A closer look at the plots also reveals that decreasing scene lighting leads to increase in the error leading to similar conclusions that a brighter image gives more error reduction. It is also interesting to note that the frame-rate that gives the most error reduction, irrespective of computational budget is 80Hz when $\alpha = 40$, and reduces to 60Hz when $\alpha = 10$.

in the 640×480 resolution. This is because the noise present in the highest resolution is averaged out due to subsampling and that tracking can run without failures at a lower resolution. On the other hand, tracking at highest resolution suffers due to relatively more noise present in the image. Importantly, 80Hz, which corresponding to the Pareto Front of the highest resolution in high lighting conditions 6.6(b), for a computational budget of 4 is moved down to 60Hz for moderate lighting conditions. This also means that for lower lighting conditions tracker spends more number of iterations to converge to a similar error scale.

Low lighting conditions Low lighting conditions decimate the colour values so much that frame-rates 120–200Hz are just pitch-black images corrupted badly with noise. As a result these frame-rates do not appear in the error curves and consequently the Pareto Fronts. Due to noise and low SNR in the high frame-rate images, 20Hz appears to be the optimal frame-rate regardless of the computational budget as shown in the Figure 6.8(a). The peculiarity



(a) $\alpha = 1$

(b) Pure Ray-traced

Figure 6.8: Error curves for (a) $\alpha = 1$, and (b) Pure ray-traced images. Reducing the light further down ($\alpha = 1$) leads to increasingly pitch-black images at frame-rates from 120–200Hz. Tracking completely breaks at these frame-rates and therefore, this range of frame-rates do not appear in the graph. Lower frame-rates range, 20–100Hz, see an increase in the error achieved mainly again due to the darkening of images and addition of camera noise with increase in frame-rate. It is also interesting to note that the error increases with increase in the iterations working at lowest pyramids for frame-rates ranging from 60–100Hz. This can be explained by the fact that images at these frame-rates are still very dark and when sub-sampled to a resolution of 80 × 60 lead to many brightness levels mapped to a very small range of grey colour levels in between 0–5 pixel values. However, as the frame-rate is decreased further, the number of brightness levels in the high resolution image increase and as a result the lowest resolutions show significant difference in the colour levels when compared to high frame-rates and therefore appear brighter. Pure ray-traced images on the other hand maintain similar brightness levels across all frame-rates and due to the slow motion in the trajectory, there is as such no opposing factors like motion blur present in the image that hamper the tracking performance for lower-frame rates – only baseline is increasing. Consequently, high frame-rate images show best performance in reducing the error but, the computational requirements to meet the real-time constraints swamp the error reduction – similar error reduction can be achieved by working on lower frame-rates while still being in real-time regime. When new processors, offering a luxury of computational power, are slotted in, frame-rates as high as 200Hz find their operatability.

of increasing error with increasing budget at lowest resolution can be explained by the fact that unavailability of enough contrast and low SNR present in the image for high frame-rate images leads to diverging results.

Conclusions obtained from tracking slow motion

- There are early cross overs at lowest pyramid for all scene lighting settings for slow motion. This is not surprising because for slow motions slightly lower frame-rates are expected to achieve similar levels of accuracy as higher frame-rates. Therefore, when computational cost is also taken into account, one observes cross overs suggesting the early use of lower frame-rates for such motions. Nonetheless, these frame-rates are much higher than the standard real-time processing rate of 30Hz.

- As the computational budget is increased, again, a switch to a higher resolution and lower frame-rate if possible is a preferred choice than just increasing frame-rate alone.

- Importantly, slow motion allows for the operatability of low frame-rates and that they figure in the Pareto Fronts of the accuracy/computational cost curves.

- When lighting conditions degrade, as expected, the Pareto Fronts show similar trends but the optimal frame-rates move to lower values specially at lowest pyramid levels. The results quite well chime with the results obtained for faster motions.

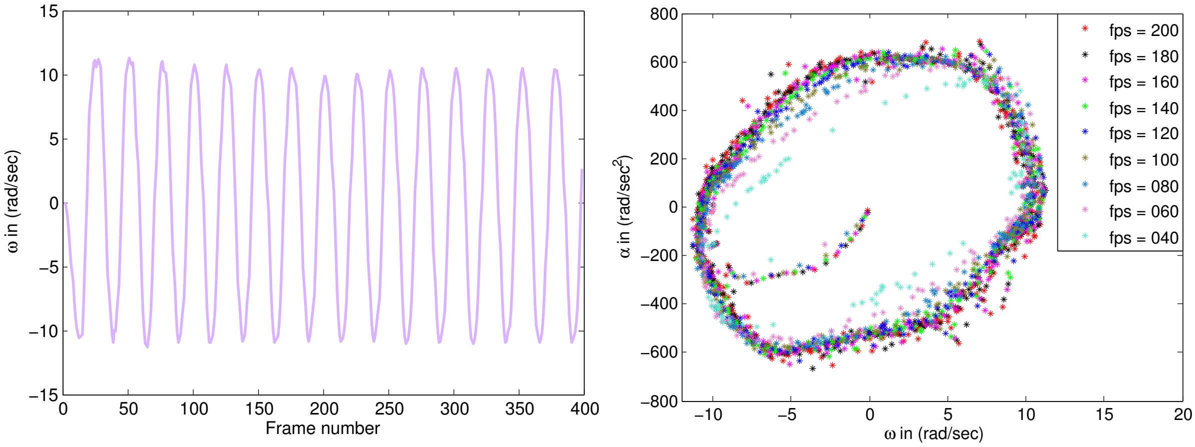
6.5.3 Pure Rotation

We have experimented with pure one-dimensional rotations to understand how different this camera motion is from the translation motion or motion involving both rotation and translation. The depth independence property of pure rotation makes it a special case to study in our analysis as this can be verified against a real experiment of camera mounted on a rotating platform. Again, the analysis is further broken into fast and slow movements of camera undergoing pure rotation. We expand the conclusions and results for them below.

Fast Motion

Our fast motion trajectory is collected from a servo motor rotating about its axis. We used this motion in our synthetic rendering frame-work to obtain images for different frame rates that are interpolated on this trajectory. Figure 6.9(a) shows the trajectory at 200Hz while (b) shows the trajectory in the form of acceleration vs velocity locus plot. The angular velocity curve is a near sine wave with the maximum velocity reaching 11.2 *rad/sec* and the minimum, -11.2 *rad/sec*.

Choice of sine wave Although any other motion could have been chosen, the reason for preferring a sine wave is purely driven by the symmetry observed in the motion and the circular/elliptic locus of data points in the trajectory. Ideally one would like to observe the performance of tracker at points where the acceleration is large and velocity near zero and acceleration is near zero while velocity is large. Such points are informative of the performance at different frame-rates and tell us the maximum/minimum velocity and acceleration at which the tracker breaks at a given frame-rate. Preferring smooth transitions between these two extreme points, i.e. a circular plot, leads us to think of sine wave as a choice of motion.



(a) Velocity at 200Hz

(b) Acceleration vs. Velocity

Figure 6.9: (a) The angular velocity at 200Hz used in the experiment for interpolation and Figure (b) the acceleration α, is plotted against angular velocity, ω for the trajectory.

Such fast motions induce typically a large amount of blur in the images at low frame-rates of 20Hz. Sample blurry images obtained at 20Hz are shown in Figure 6.10. An important thing to remember is that trajectory has varying degrees of acceleration and velocity which as a consequence brings varying degrees of blur typically in the low frame-rate images. There are more chances of tracker failing at these frame-rates not because of large motion blur between two consecutive images but mainly due to the uneven blur. For instance, when the trajectory begins, the camera is at rest and in the other instant the camera moves with a velocity but also has acceleration. This means the first image does not have blur due to stationarity of the camera while the other image has motion blur. Matching become significantly hard because as such there is no unique transformation that registers two images.

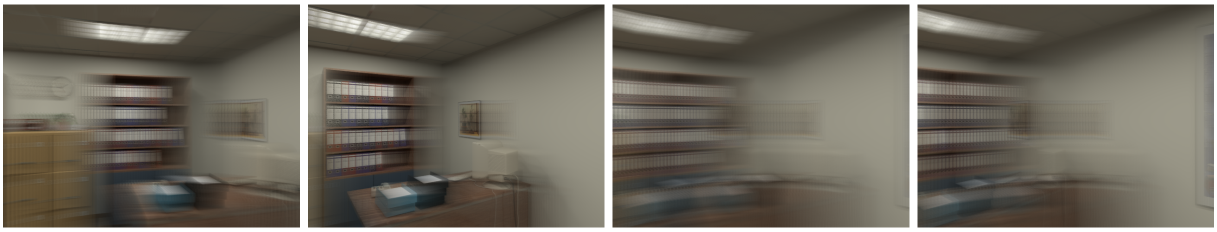
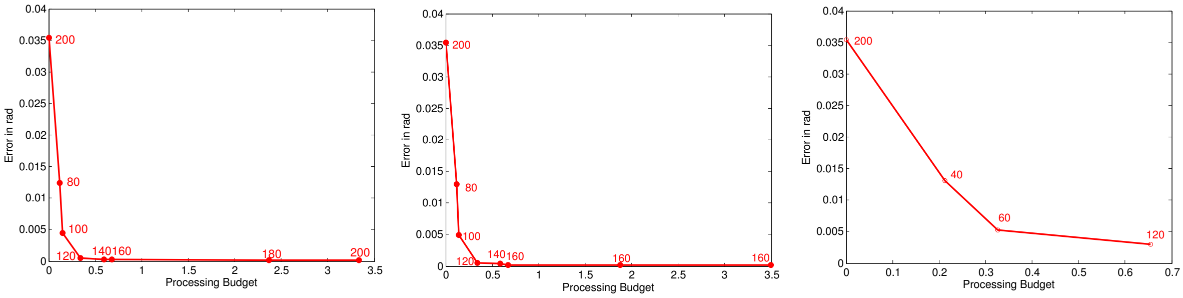


Figure 6.10: Synthetic photo-realistic images obtained for pure rotation around Y-axis. The images shown correspond to 20Hz frame-rate and show a significant amount of motion blur.

High Lighting Conditions Our high lighting results show (Figure 6.11(a)) that with increase in computational budget, the optimal frame-rate increases i.e. given more processing power, it is sensible to increase the frame-rate. It is also observed that for very low computational budget we still get frame-rates of the order of 80Hz which is higher than the standard 30Hz frame-rate range used in most tracking experiments.



(a) $\alpha = 40$ (b) $\alpha = 10$ (c) $\alpha = 1$

Figure 6.11: The Pareto Fronts obtained assuming strong ($\alpha = 40$) and moderate ($\alpha = 10$) lighting conditions. We have shown only the results obtained at the lowest pyramid since the rotation based tracker already provides a very good estimate of rotation at lowest pyramid. Therefore, the higher resolutions only provide very small gains. (c) The Pareto Fronts obtained for very low lighting conditions.

However, it is observed that there is as such no clear advantage of switching resolutions in pure-rotation case because the gains obtained are minuscule. This is expected, because the rotation is purely in one-dimension and the alignment obtained at the lowest resolution is already a very good estimate of the angle that the two frames are separated by — further increase in number of pixels, though for polishing the estimate, hardly gives any information in this case. Therefore, what this result says is that it is just *sufficient to work at the lowest resolution but high frame-rate to obtain quite accurate estimates of 1-D rotation* because lighting conditions permit to work on these high frame-rates.

Moderate Lighting Conditions We have observed (Figure 6.11(b))in this experiment that our results remain mostly the same except the preclusion of 200Hz frame-rate from the Pareto Front when compared against the corresponding results obtained assuming strong lighting conditions. The resolution does not help much here either. Overall, we see the degradation of image quality leads to omission of high frame-rates from the Pareto Front.

Low Lighting conditions The low lighting conditions see a similar trend — the optimal frame-rates are lower than the corresponding optimal frame-rates we obtain when the lighting conditions are better as shown in Figure 6.11(c). Moreover, as observed in moderate lighting conditions higher frame-rates tend to suffer as a result of low lighting conditions and we see that frame-rates above 120Hz completely disappear from the graph.

Slow Motion

We have obtained similar results for slow motion except that the optimal frame-rates shift to lower values as one would expect for slow motion.

6.6 Quantifying Performance Limits of Camera Tracking

In this section, we perform additional experiments to understand the limits of camera tracking by degrading the set of parameters the tracker depends on. This includes degrading depth-map, degrading and decimating grey values and spatial blurring of image pixels. We have performed all these experiments on joint translation and rotation motion.

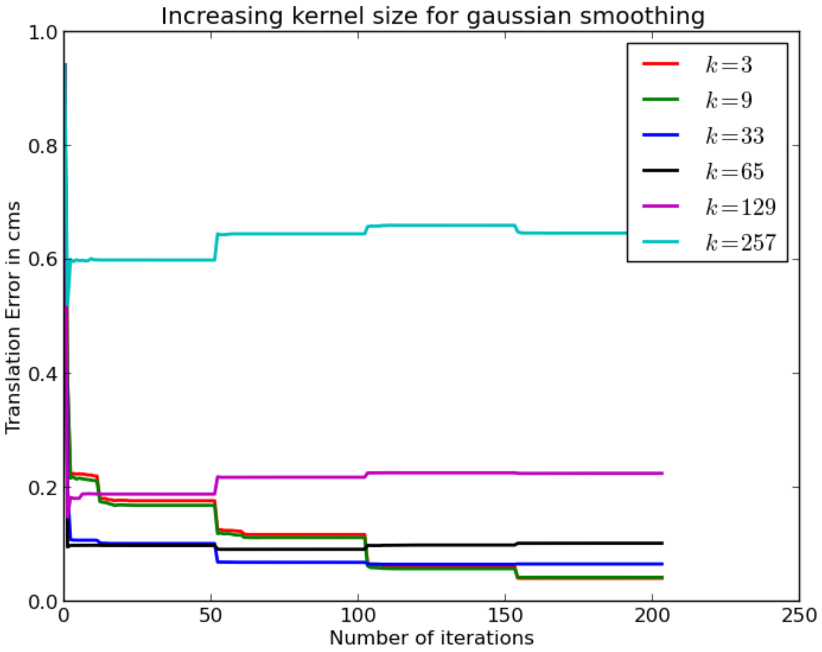
6.6.1 How Quickly does the Tracking Get Affected When Images Start to Get Blurry?

In this experiment, we study the limits of camera tracking as images are progressively blurred using a Gaussian kernel. Gaussian blurring an image here could be thought of as a case of camera defocus. The blur sigma and kernel size are chosen to be

$$\sigma = 0.3\left(\frac{ksize}{2} - 1\right) + 0.8 \tag{6.1}$$

6.6.2 Slow Motion

Figure 6.13 shows images obtained by progressively blurring the original image with different kernel sizes.



(a) Increasing kernel size

Figure 6.12: Tracking performance as the kernel size for image and depth blurring is increased. Tracking still runs at very low resolution images with kernel size set as high as 65. The standard deviation of Gaussian distribution is set as $0.3\left(\frac{ksize}{2}-1\right)+0.8$.

We have observed in our experiments that as the size of the blur kernel, *ksize*, used to smooth the images is increased, the tracking gets better till a certain point because of the ease in localising the minima when the images are smooth but the accuracy begins to degrade once the kernel size becomes very large. In our experiments on slow motion trajectory we have observed that tracking performs better still at lowest resolution when the kernel size is 65 × 65 but further increase in kernel size leads to only degradation of results. On the other hand, high resolutions begin to show remarkable degree of degradation in results relatively quite early. Figure 6.12 shows the tracking performance at different resolutions for increasing kernel size.

6.6.3 Fast Motion

We used the same slow motion trajectory but skipped frames to create a lower frame-rate equivalent and ran the same set of experiments progressively blurring the images. We have observed similar results where tracking performs better at the lowest resolution when the kernel size is 65× 65 (and $\sigma = 10.25$) and further increase in the kernel size (and consequently the Gaussian standard deviation) sees the accuracy tailing off while the highest resolution performs poorly with the increase in the blur kernel.

6.6.4 How is Tracking Affected by Quantisation of Pixel Values?

In this experiment, we progressively darken the images by various decimating factors. We use 16-bit images obtained from POVRay and divide the intensity values by the factors and convert them to 8-bit images. Performing a comprehensive analysis of the image pixel values and hence the image gradients lets us understand when and under what lighting conditions, the tracker breaks. In fact, if the camera was a perfect analog sensor and reported the pixel values in floats, the tracking would have remained perfectly immune to the lighting conditions. Recall that the update in the SE3 is obtained by

$$\delta \mathrm{u} = \left(\sum_{\mathrm{x} \in \mathrm{I}} \mathrm{J_x}^T \mathrm{J_x} \right)^{-1} \sum_{\mathrm{x} \in \mathrm{I}} \left(\mathrm{J_x}^T \mathrm{e_x} \right) \tag{6.2}$$

If the image intensities are floats and they all change by the same amount, the expression is invariant to the scale by which they change. In other words, the content of the image is not destroyed. However, it is due to the discretisations in the digital camera that similar pixel values get packed in the same bin leading to same grey colour. This destroys and changes the original image gradient, the effects of which are clearly seen in the tracking performance. Lowest resolutions are affected the most as the decimation of the pixel values is increased. Due to fewer pixels available at the lowest resolution and the loss of gradient information, the ability of the tracker to obtain a meaningful update is greatly challenged.

Figures 6.14–6.17 show how histograms of the images change as images are decimated by different factors ranging from 40 to 1. Figure 6.18 shows images obtained at different decimating factors for two different resolutions.

The results of image decimation are shown in Figure 6.19(a) where we see that tracking performance gets severely affected at the lowest resolution where downsampling and bit

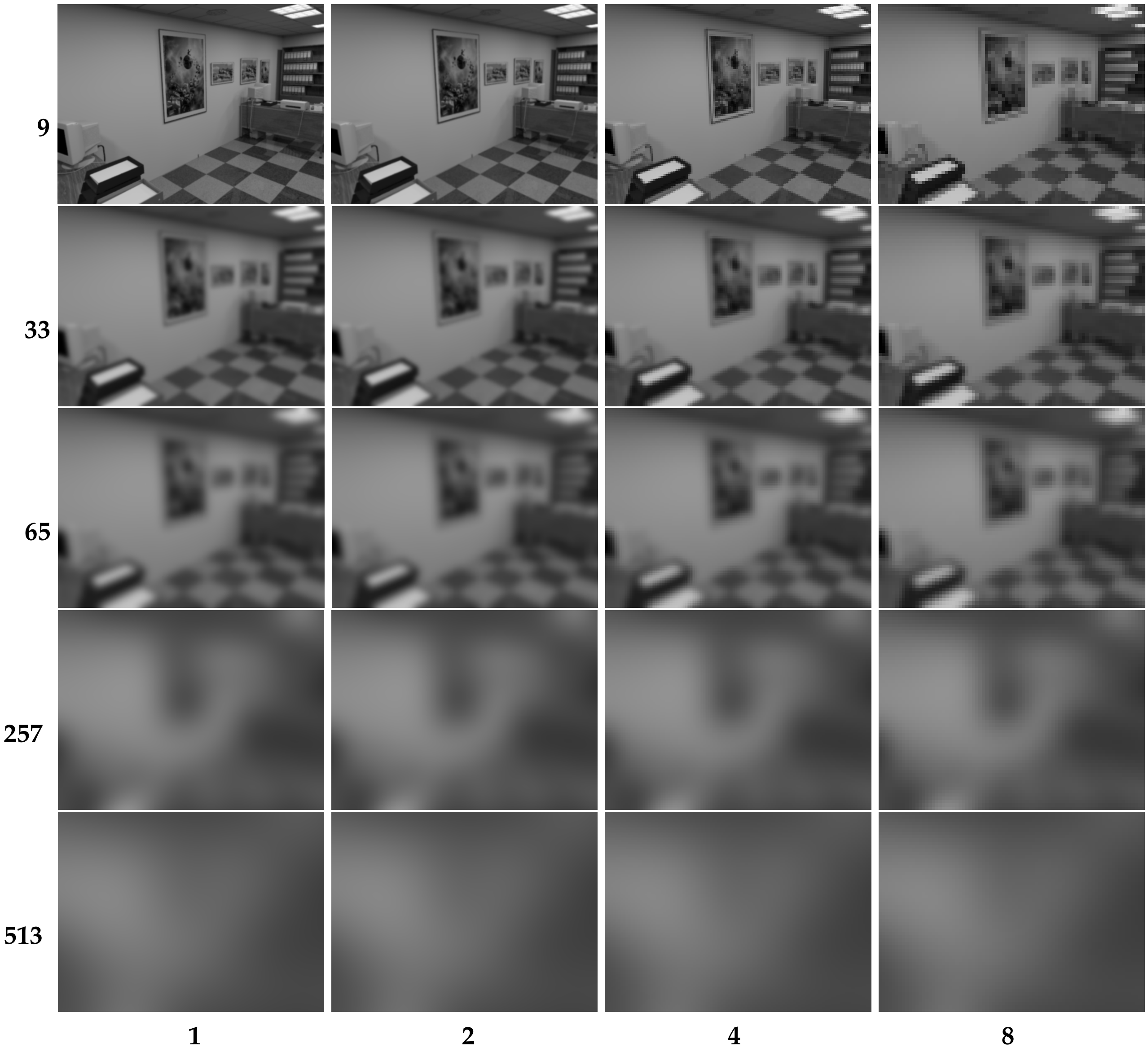


Figure 6.13: Sample image shown with progressive increase in the amount of blur with their pyramidal levels. The vertical axis reads the kernel size and horizontal axis, the subsampling factor of image. Such blurring could appear in the case of camera defocus. In all the experiments the depth-map was blurred by the same amount. Tracking performs without breaking even at 257 × 257 kernel size.

quantisation leads to a huge loss in the information contained in the image. On the other hand, at the highest image resolution we still see that error curves are tightly clustered.

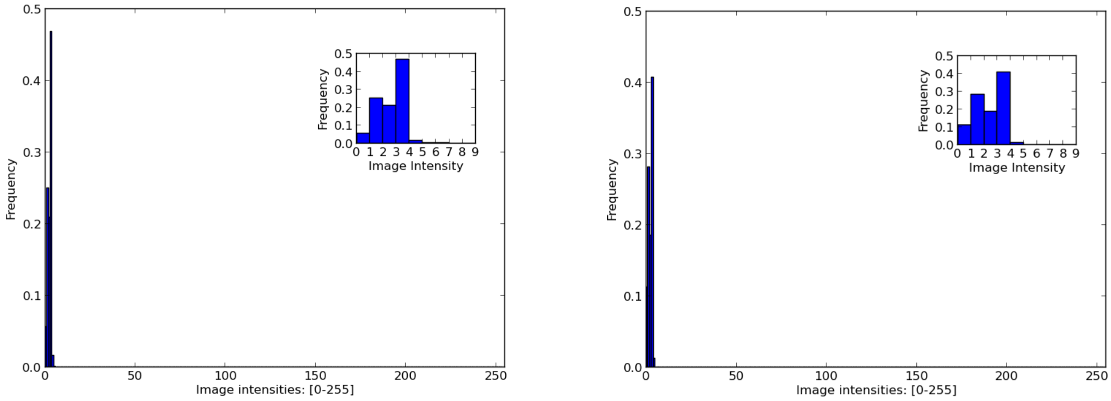


Figure 6.14: Left: Histogram of highest image resolution with intensities decimated by 40 and Right: Histogram of the same image but at the lowest resolution 80×60.

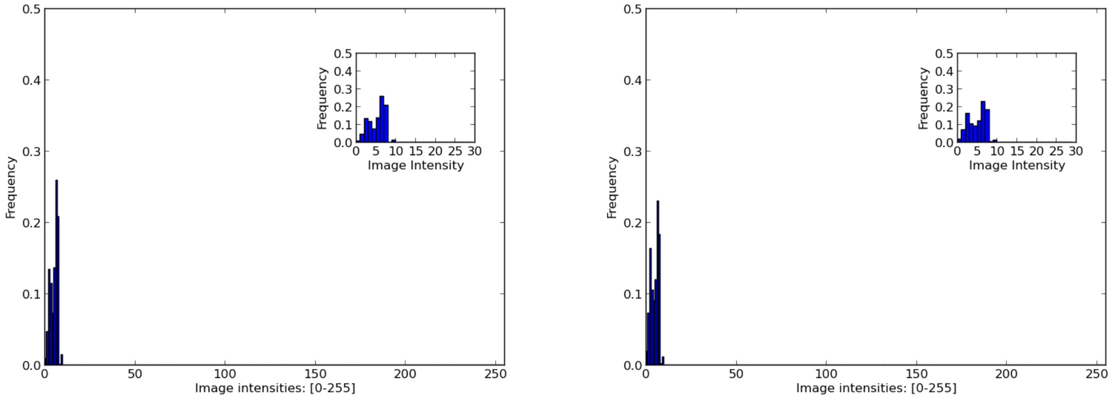


Figure 6.15: Left: Histogram of highest image resolution with intensities decimated by 20 and Right: Histogram of the same image but at the lowest resolution 80×60.

6.6.5 How does the Tracking Degrade as We Add Noise to the Depth-Map?

In this experiment, we analyse how the tracking accuracy changes as the depth-map is made noisier understanding the importance of accurate depth-map in the optimisation. We add Gaussian noise in the depth map with increasing standard deviations — the units of standard deviations are same as the depth. Figure 6.19(b) shows our results for adding noise to the depth-map. We understand that at lowest resolutions, the accuracy of a depth-map is hardly essential leading to the error curves tightly clustered. This is explained by the fact that lowest resolution motion tends to be either planar or rotational both of which do not rely on depth-map. Also remember that the depth-map at lowest resolution is obtained by coarse

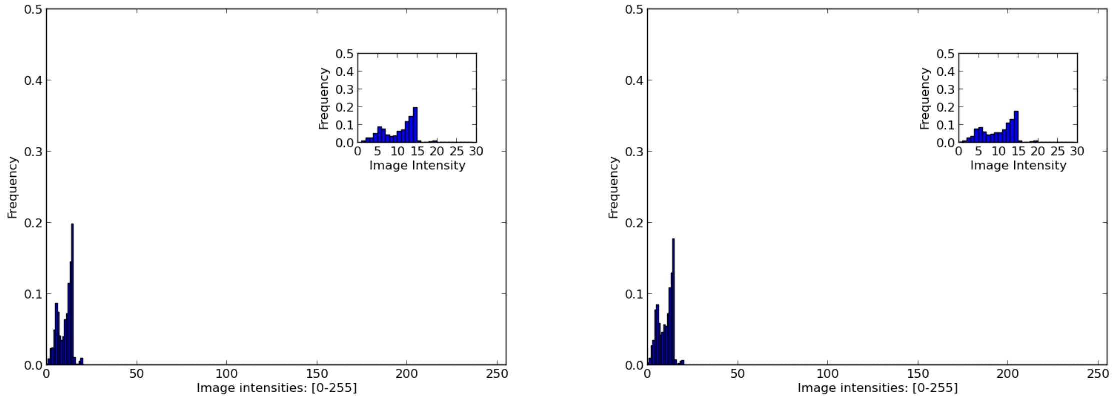


Figure 6.16: Left: Histogram of highest image resolution with intensities decimated by 10 and Right: Histogram of the same image but at the lowest resolution 80×60.

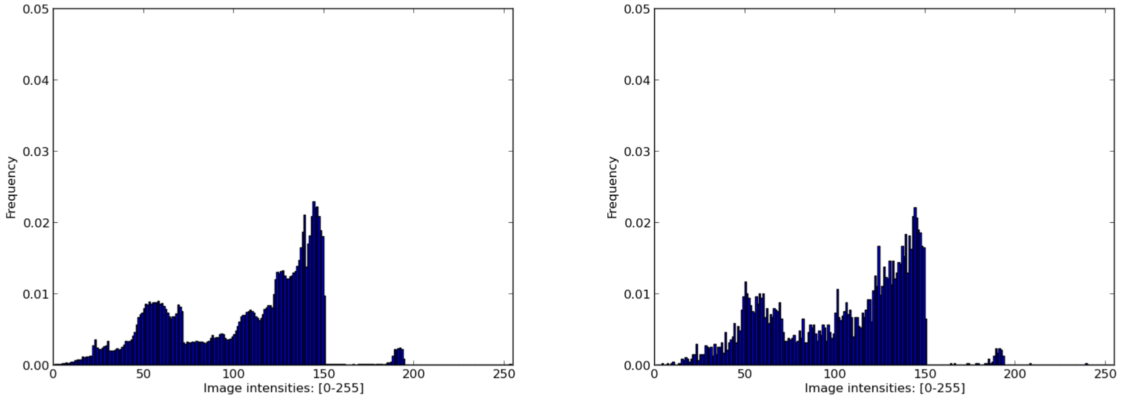


Figure 6.17: Left: Histogram of highest image resolution with intensities decimated by 1 and Right: Histogram of the same image but at the lowest resolution 80×60.

subsampling which results in also averaging out the noise. Increasing the resolution shows a gradual increase in the error with increase in Gaussian noise while the highest resolutions show an aggressive degradation even for relatively smaller noise standard deviation.

6.7 Conclusions

Our experiments give an acute insight into the trade-offs involved in high frame-rate tracking. With perfect lighting and essentially infinite SNR, the highest accuracy is achieved using a combination of high framerate and high resolution, with limits only set by the

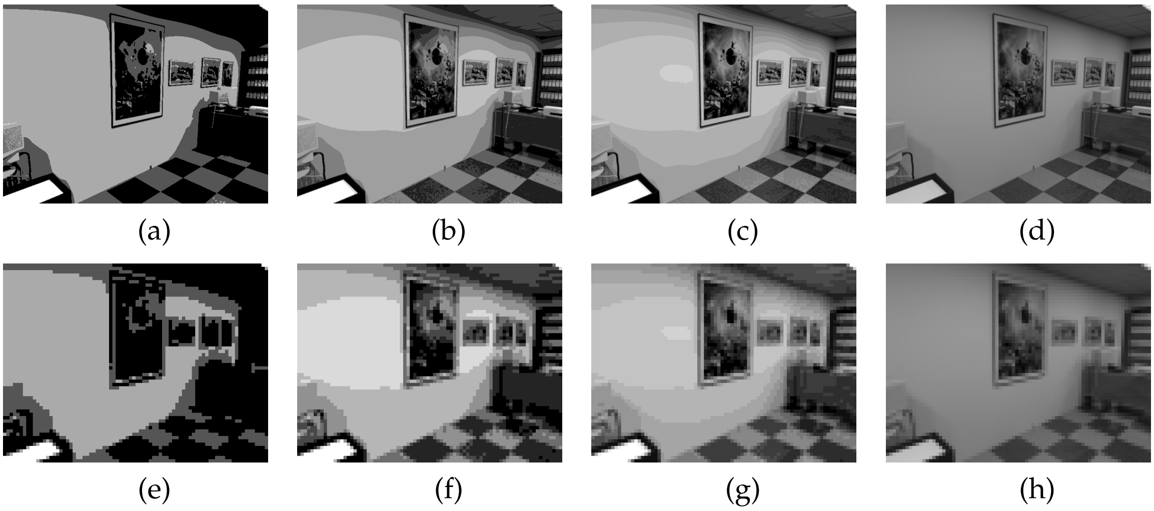
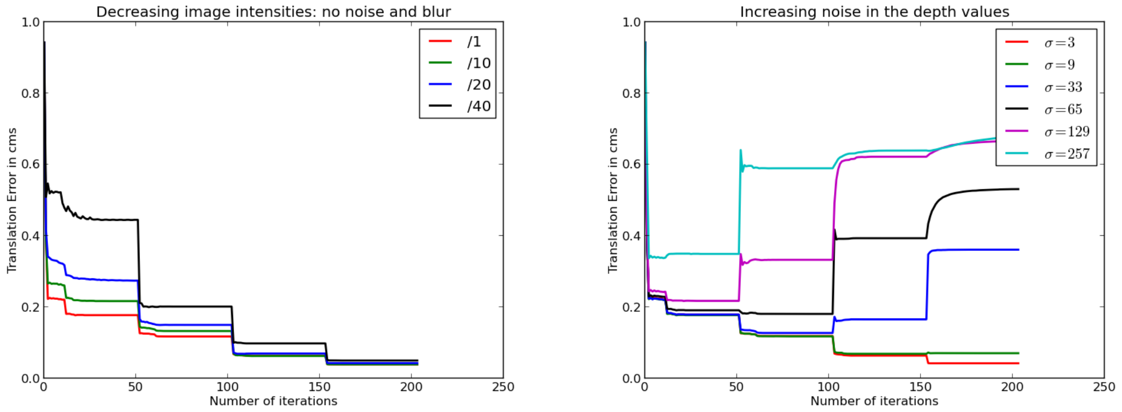


Figure 6.18: Sample image in one of the sequences. First row: Images (640×480) corresponding to different decimation factors in decreasing order. Second row: Similar image decimation but with resolution decreased by a factor of 8 (i.e. 80×60) at the same time. The actual images for high decimation factors are enhanced using gthumb enhancement function while the upsampling (80 × 60) for display is done using nearest neighbour interpolation.

available computational budget. But using a realistic camera model, there is an optimal famerate for given lighting levels due to the tradeoff between SNR and motion blur. Therefore, frame-rate cannot be arbitrarily pushed up even if the budget allows because of image degradation. Decreasing lighting in the scene shifts the optimal frame-rates to slightly lower values that have higher SNR and somewhat more motion blur for all resolutions, but overall increasing resolution results in quicker improvement in accuracy than increasing frame-rate. Hasinoff etal [Hasinoff et al., 2009] [Hasinoff et al., 2010] also worked on time-bound analysis but only for SNR image quality assessment with either static cameras or simple planar motions.

Our dataset, rendering scripts used to generate it and other materials are available from http://www.doc.ic.ac.uk/~ahanda/HighFrameRateTracking/. We hope this will further motivate both applied and theoretical investigations of high frame-rate tracking relevant to the design of practical vision systems. Our dataset may also be useful for analysis of many other 3D vision problems.



(a) Colour Decimation

(b) Depth Noise

Figure 6.19: Left: Tracking performance as a function of progressive decimation of pixel values by different factors. The general trend is that as pixel values are subsampled, the tracking gets affected the most at the lowest resolution but the error graphs begin to cluster closely as image resolution is increased. Right: The performance of tracking on the same sequence but with increasing Gaussian noise added to the depth values while still maintaining original image intensity values. The graph shows that noise in the depth values does not significantly affect the results at the lowest pyramid; this is in contrast to decimating image intensities, while the higher resolutions show a remarkable performance degradation even for very low noise in the depth values.

Chapter 7

Tracking Analysis: Real Experiments

Contents

In this chapter, we focus on new experiments performed in a real world setting to validate the experiments carried out with synthetic framework. Our experiments are only for the pure rotation case. A validation of any general 6DoF camera motion is still difficult due to the complexities involved in obtaining ground-truth at the extreme dynamics where investigation of high frame-rate is interesting. On the other hand, pure rotation motion can be easily controlled in a real-environment and the rotations can be measured with ground truth precision. Therefore, this seems the relevant motion we can try in our experiments.

Our pure rotation set-up is constructed with a dynamixel servo motor for generating pure rotation motion clasped by a wooden platform from either side of it. The camera sits on the servo and a high bandwidth gyro is strapped on the camera for obtaining ground truth. Figure 7.1 shows the wooden platform we constructed and various different components that we use in our set-up. Next, we state the assumptions used in the experiments and discuss the results in detail later.

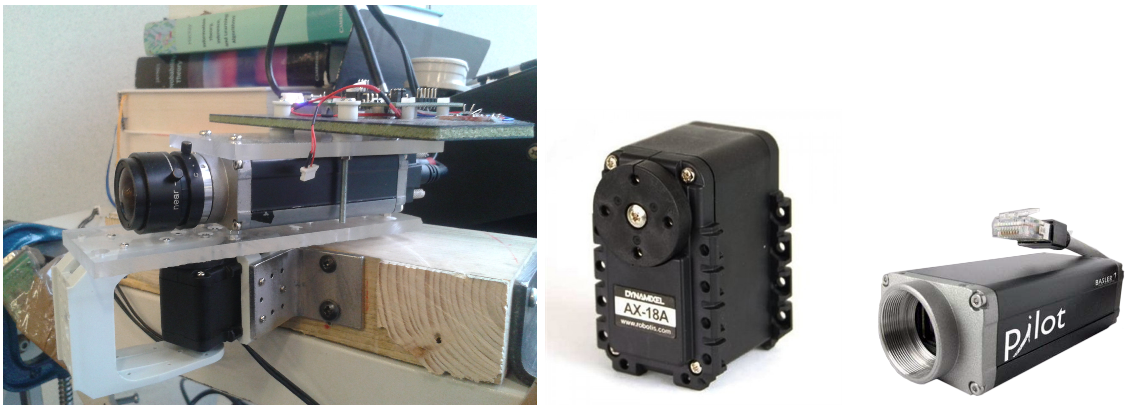


Figure 7.1: The wooden platform we constructed and the set-up we use to enable pure-rotation about the vertical axis. A high bandwidth gyro mounted on the camera is able to sample as fast as 4kHz to obtain the ground truth angular velocity. The digital dynamixel servo rotates the camera at different dynamics with magnitude of accelerations touching as high as 600 rad/sec^2 and velocities 15 rad/sec.

7.1 Assumptions

1. Our primary assumption in carrying out real experiments is that our real scene may not have to look similar at all when compared to the scene we used in our synthetic experiments. A natural question then is *why are we doing real experiments?* We have argued in Chapter 3 about replicating a real scene is quite a labourious task and out of the scope of this thesis. Therefore, the short answer is that we do not aim to show that we get exactly similar optimal frame-rates as our synthetic experiments show but we would like to show that the pattern of optimal frame-rates remains more or less the same. For instance, with a decrease in scene lighting we should see the optimal frame-rates shifting towards lower values.

2. We are only verifying results for pure 1-D rotation motion of a camera which we have already experimented with in our synthetic framework. The choice of pure 1-D rotation motion is also driven by the complexities involved in recovering a near perfect ground-truth and camera poses for any other sort of motion that involves translation.

3. We would also like to experiment with various dynamics of the motion to study robustness of tracker. For instance, we would like to know the dynamics of the motion that becomes untrackable for a given frame-rate. This is something we have not studied in our synthetic experiments — the enormity and the huge rendering time associated with collecting synthetic data needed for obtaining meaningful statistics for robust-

ness largely prohibits our understanding of this parameter in synthetic experiments. However, real data collection makes the whole task much easier.

The robustness is defined as the frequency of success/failure of a tracker over a given sequence. The success/failure mode is determined by a user defined fixed threshold. We later show how robustness increases both as a function of error threshold and camera frame-rate. Further, we show how the optimal frame-rate changes with the dynamics of the motion.

4. The variety and the magnitude of dynamics of the motion in which we are interested in the experiments requires us to choose a sensor that is able to provide estimates at a rate faster than the fastest sampling rate of image capture. Therefore, we have used a very high bandwidth and high frequency gyro to capture ground truth data.

5. Most of our other assumptions are the same as in synthetic experiments. *e.g.* the data is already collected and stored in the hard disk.

7.2 Angular Velocity Computation

It is important to remember that the dense tracker we use returns the positional estimates of the motion *i.e.*, it returns the rotation matrix instead of angular velocity. On the other hand, a gyro provides estimates of angular velocity. Therefore, appropriate conversion of rotation to angular velocity or vice-versa is required. We choose to obtain the angular velocity from the rotation matrix. This is because (a) the rotations we obtain from the algorithm are relative only between two frames (b) integrating the gyro for rotation often leads to drift due to the noise present in the readings. Given the alignment of two frames, the angular velocity can be computed as follows

$$\log(\mathsf{R}_{rl}) \times \mathsf{fps}\ . \tag{7.1}$$

7.2.1 Derivation

If the pose of the reference frame is taken to be Identity, I, the dense alignment algorithm then returns a relative pose between the reference frame and the live frame[1] *i.e.* R_{lr}. The standard way of computing 3×3 skew-symmetric angular velocity matrix of camera given a

[1] The matrix returned by the algorithm is R_{lr} but we compute the velocity from the inverse of this matrix.

rotation matrix follows a simple manipulation[2]:

$$\hat{\omega}(t) = \frac{d\mathsf{R}_{rl}}{dt}\mathsf{R}_{rl}^{T}(t) \; . \tag{7.2}$$

This can be easily derived from simple principles of first order derivative [3]

$$\hat{\omega}(t) = \frac{\mathsf{R}_{lr}(t+dt) - \mathsf{R}_{lr}(t)}{dt} \tag{7.3}$$

$$\begin{aligned} \mathsf{R}_{lr}(t+dt) - \mathsf{R}_{lr}(t) &= \mathsf{R}_{rl}(dt)\mathsf{R}_{lr}(t) - \mathsf{R}_{lr}(t) && (7.4)\\ \Rightarrow \hat{\omega}(t) &= \left(\frac{\mathsf{R}_{rl}(dt) - \mathsf{I}}{dt}\right)\mathsf{R}_{lr}(t) && (7.5)\\ \therefore \hat{\omega}(t) &= \frac{d\mathsf{R}_{rl}}{dt}\mathsf{R}_{rl}^{T}(t) \; . && (7.6) \end{aligned}$$

Using the orthonormality constraint of rotation matrices and rewriting the expression for angular velocity matrix, we get

$$\begin{aligned} \hat{\omega}(t)\mathsf{R}_{rl}(t) &= \frac{d\mathsf{R}_{rl}}{dt} \quad \because \mathsf{R}_{rl}^{T}\mathsf{R}_{rl} = \mathsf{I} && (7.7)\\ \Rightarrow \mathsf{R}_{rl}(t) &= \exp^{\int \hat{\omega}(t)dt}\mathsf{R}_{rl}(0) && (7.8)\\ \mathsf{R}_{rl}(t) &= \exp^{\int \hat{\omega}(t)dt} \quad \because \mathsf{R}_{rl}(0) = \mathsf{I} \; . && (7.9) \end{aligned}$$

Taking the matrix logarithm of both sides and assuming that the angular velocity remains constant within the small time change $dt = \frac{1}{\mathsf{fps}}$, one arrives at an expression of angular velocity, $\omega(t)$ as $\log(\mathsf{R}_{rl}) \times \mathsf{fps}$.

7.3 Gyro Characteristics

We use a 3-axis MEMS[4] ITG-3200 gyro[5] that has enhanced biased and sensitivity temperature stability that reduces the need for user calibration. The bandwidth of the gyro can be tuned using a selectable user specified low-pass filter. The output is obtained from 16-bit ADCs via I^{2}C bus. The gyro characteristics are summarised in Table 7.1. The gyro bandwidth is kept to be 256Hz that is high enough to pick up high frequencies in the input but at the same time not high enough to allow picking up noise in the input.

[2] The expression is obtained from the urls: http://en.wikipedia.org/wiki/Rotation_formalisms_in_three_dimensions#Rotation_matrix_.E2.86.94_angular_velocities and http://www.physics.sc.edu/~yar/phys701_2011/lectures/notes_rotations_ver2011.pdf. $\hat{\omega}(t)$ is a 3×3 matrix while $\omega(t)$ is a 3×1 vector.

[3] A beautiful short mathematical tour of various properties of rotation matrices is given in http://www.scribd.com/doc/69129232/5/Vector-Cross-Products-and-Skew-Symmetric-Matrix-Algebra

[4] Micro Electrical Mechanical Systems

[5] http://www.invensense.com/mems/gyro/documents/PS-ITG-3200A.pdf and http://www.invensense.com/mems/gyro/documents/RM-ITG-3200A.pdf

Rate Gyro	**Characteristics**
Rate Noise Spectral Density	5.2e-4 $rad/sec/\sqrt{Hz}$.
Nonlinearity	0.2 %.
Cross axis sensitivity	2 %
Full scale range	$\pm$ 34.90 rad/sec
Sensitivity scale factor	823.62 LSB/(rad/sec)

Table 7.1: Different characterisitics of the ITG-3200 gyro we used in our experiments. The numbers are obtained from the datasheet. http://www.invensense.com/mems/gyro/documents/RM-ITG-3200A.pdf

7.4 Aligning Gyro and Camera Angular Velocity Estimate

Our experimental set-up allows for a camera mounted on a servo to capture images at varying frame-rates while a gyro strapped on to the camera measures the angular velocity of the motion. The gyro is clocked at 4kHz with a bandwidth of 256Hz and thereby provides highly upsampled velocity estimates. This is quite essential for our analysis where we want to measure extreme fast dynamics of motion. A high band-width gyro therefore captures easily the high frequency content present in the velocity which a low-bandwidth gyro filters out.

Before comparing the velocity estimates given by the tracker with the gyro readings, there are some key observations we would like to state:

1. The gyro velocity capture and camera image capture run on two different buses — USB and Gigabit Ethernet respectively — and are not synchronised. They must either be synchronised via a hardware trigger or via an optimisation to obtain the offset in time.

2. We also noticed that there is an extra scale factor associated with estimates of velocity obtained via pure vision and raw gyro estimates; this is a calibration parameter. This factor stems from the inaccuracies in the camera calibration, camera velocity estimation process and noise in the gyro readings. However, in our experiments we have observed that most of the time this calibration parameter is near to unity.

3. The gyro bias, the offset in the gyro values, needs to be cancelled before the gyro readings can be used. When the gyro is at rest, it is not always reporting zero angular velocity. This leads to a biased reading: an extra non-zero value is added on top of the real gyro readings. However, the bias is constant and can be obtained by a long term

average of gyro readings when it is at rest. It can be subtracted from the obtained gyro readings to get a bias free estimate of angular motion.

4. The camera optic center does not sit exactly on the axis of rotation of the servo and therefore it may not be undergoing pure rotation. To counter the effects of this, we image a far away scene so that the parallax that may arise in the image due to the camera not undergoing pure rotation is negligible.

5. The orthogonal axes of the gyro may not be aligned properly with the orthogonal axes of the camera. Therefore, we need an extrinsic calibration of gyro and camera angular velocity estimates to obtain the misalignment in their orthogonal axes. Although we have found the alignment to be very close to Identity and have instead only focussed on aligning angular velocities around the y-axis.

6. Lastly, the image timestamps reported by the camera are not exactly integer multiples of the set shutter time. This could be due to lags in camera triggering and inaccuracies of the clock inside the camera that measures timestamps.

We have used a non-linear optimisation to align the gyro readings with the camera readings and then obtain the error in rotation. This is because even if the gyro and camera velocities were synchronised, we would still need a calibration to obtain the scale factor, the bias in the gyro readings [6] and the misalignment in the axes. The optimisation parameters, s, the scale factor and b, the gyro bias; x, time shift in the two signals and α, the misalignment in the timestamps, are embedded in a least squares optimisation that is formally written as:

$$[x^*, \alpha^*, s^*, b^*] = \arg \min_{x,\alpha,s,b} \sum_{t=t_{start}}^{t_{end}} (\omega_{cam}^{y}(t) - s(\omega_{gyro}^{y}(\alpha t + x) + b))^2 \tag{7.10}$$

The gyro bias and noise model is inspired by [Kapaldo, 2005, p. 7] and [Hwangbo et al., , p. 5].

We use the MATLAB function fminsearch to solve this non-linear least squares optimisation. fminsearch does not compute the gradients of the signal and instead uses a simplex algorithm to align the two signals. The optimisation is initialised with [0 fps 1 0]. The convergence is obtained when the error in the alignment reaches a fixed threshold.

[6] This bias varies with every new experiment and moreover, the gyro has another start-up bias that is different every time a gyro is initialised.

7.5 Experiments

We classify our real-experiments similarly based on the dynamics of motion with which the camera is moving and lighting in the scene. To generate different dynamics of motion, the servo is commanded to oscillate back and forth between two angular positions, θ_{min} and θ_{max}, within a given time period. The servo compliance margin[7], that creates torque profile which servo motor follows as it is nearing the goal position, is adjusted so that near sine waves are observed in the velocity profile. To vary the scene lighting, we simply turn on and off different buttons that control the lighting in the room.

We run our dense pure **SO(3)** rotation tracker on consecutive images of the camera. Each such image pair provides an estimate of the relative orientation between them. The estimated orientation is then compared against the corresponding interpolated ground truth values as described in Section 7.4 to obtain the error.

It is important to remember that the ground truth values obtained are susceptible to vagaries of the interpolation technique used. Therefore, the actual ground truth may be different to the interpolated ground truth but within a given region of uncertainty. We therefore, decided to compare only the results until where the camera estimation is definitely inferior to the interpolated ground truth *e.g.* this could be comparing the results at the lowest resolution.

We register camera and gyro velocity independently for every frame-rate for a given experiment. The scale factors obtained from different frame-rates are further averaged. Scale factors with differences less than 1% of the maximum scale factor are considered to obtain the corresponding averaged scale factor and bias. We are motivated by the fact that in our experiments we observed decreasing scale factor values on decreasing the frame-rates which is an indication that lower frame-rates tend to underestimate the camera motion owing to increasing baseline and motion blur. Therefore, only those scale factors are averaged that are close to the maximum scale factor. These averaged values are then fixed and the optimisations for all frame-rates for that given experiment are re-run to obtain final interpolated ground truth values. This ensures that all registrations are performed with constant scale factors and biases. The averaging is done independently for each different experiment carried out at any different time due to the fact that biases are time dependent and vary with time.

[7] The dyamixel servo provides various control parameters that can be adjusted to obtain a particular kind of motion, http://robosavvy.com/store/product_info.php/products_id/638.

7.5.1 Experiments With Scene Lighting

Our real experiment with scene lighting demonstrates how it affects the optimal frame-rate. We collect image data at different frame-rates of office room first with light settings set to the highest level and later turning off the lights.

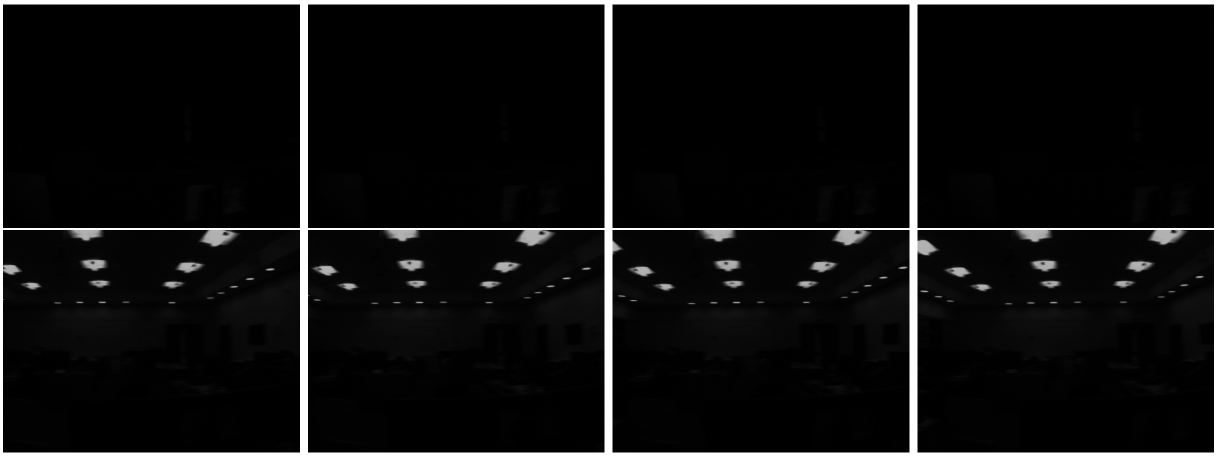
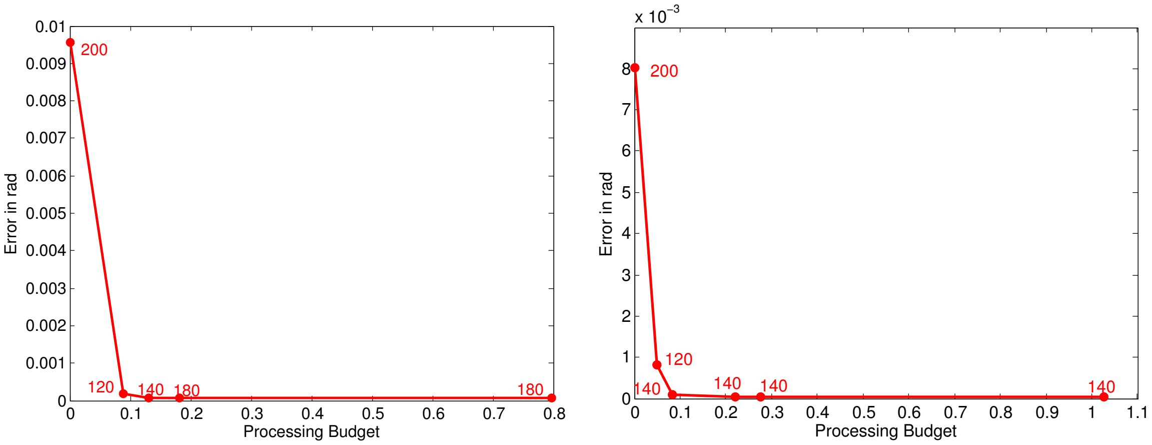


Figure 7.2: Images obtained at 200Hz for two different lighting conditions. Top row shows images obtained when lights switched off while bottom rows shows the images captured with light on. The difference due to light settings is clear.



(a) High Lighting

(b) Low Lighting

Figure 7.3: The results of the camera tracking with varying lighting settings.

We observe that as scene lighting decreases, the optimal frame-rates tend to shift towards a lower value confirming our conclusions we obtain in our synthetic experiments. We feel only a single experiment is sufficient to validate our conclusions.

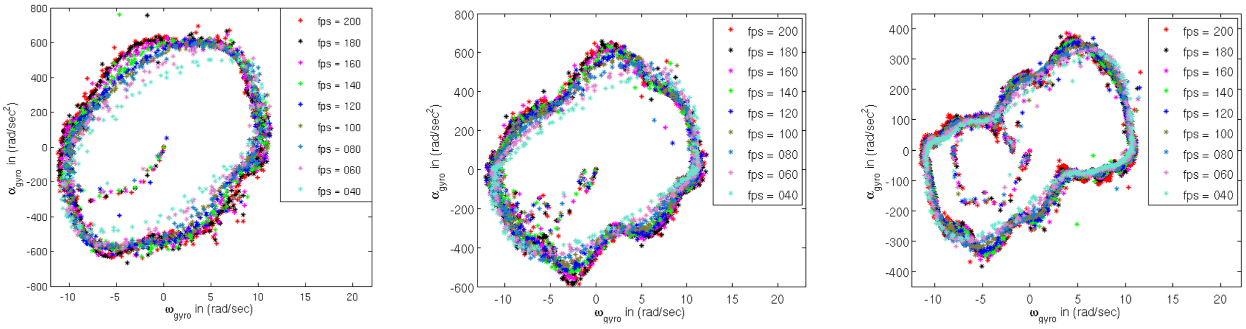
7.5.2 Experiments With Camera Motion

We obtain sequences for varying degrees of camera motion to analyse how the camera motion affects the optimal frame-rate. Table 7.2 summarises the dynamics we analyse in our experiments. We also show various parameters for the servo motion. The variables θ_{min} and θ_{max} denote the minimum and maximum angle the servo is commanded to move within the time shown in milliseconds though it may not necessarily sweep this angle sector fully within the time limit.

Time (in ms)	θ_{min}	θ_{max}	ω *(rad/sec)* **(min/max)**	α *(rad/sec^2)* **(min/max)**	Freq
58	62	217	-11.1/11.2	-667.2/983.2	8.62Hz
108	62	217	-11.7/11.8	-606.9/658.2	4.62Hz
201	86	208	-11.3/11.3	-736.3/386.7	2.48Hz

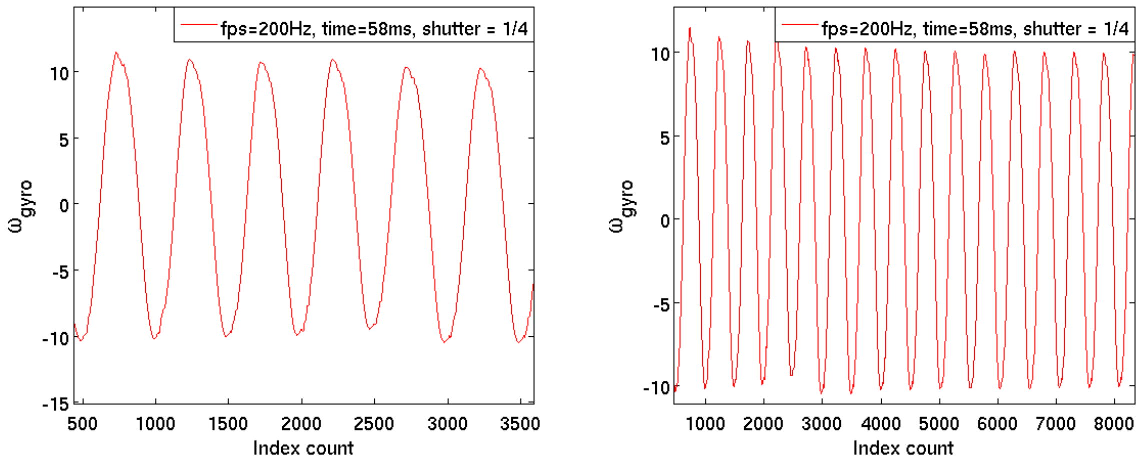
Table 7.2: Different range of dynamics analysed in our experiments.

Importantly, we also display acceleration vs velocity profile in Figure 7.4 plotting various data points as the camera moves through various different parts of the trajectory. This type of plot is more informative when quantifying the limits of camera tracking. For instance, tracking may break beyond a certain limit of acceleration and velocity or a combination of both and we would like to know the limits for different frame-rates. We begin our results section with the fastest motion.



(a) 8.62Hz (b) 4.62Hz (c) 2.48Hz

Figure 7.4: Acceleration vs velocity profiles observed in the servo motion for variety of fast dynamics used in the experiment as described in Table 7.2



(a) Zoom-in

(b) Zoom-out

Figure 7.5: Raw angular velocity reported by gyro for frequency of motion set to 8.6Hz. The curve is not exactly sine wave but a near sine wave with high frequencies squiggles observed when it is near to achieving its maximum or minimum velocity. The curve also shows that it has some low frequency carrier wave modulated on top of it.

Camera Motion Frequency 8.6Hz

This is the fastest motion we analyse in this experiment. The servo makes 40 sweeps back and forth at a frequency of 8.6Hz with the aim that the velocity profile of the motion resembles a sine wave. The choice of sine wave motion is driven by the fact that (a) it is easy to generate compared to a rectangular wave which is ideal for the experiments but nearly impossible to generate (b) it leads to a circular/elliptical locus of the smooth motion that allows us to split it up into regions of constant acceleration and velocity which we later analyse for better understanding of the tracking (specially failures) in real experiments. It is easy to see that for pure sine wave the locus of points lying on the acceleration-velocity curve is expected to be standard ellipse of the form

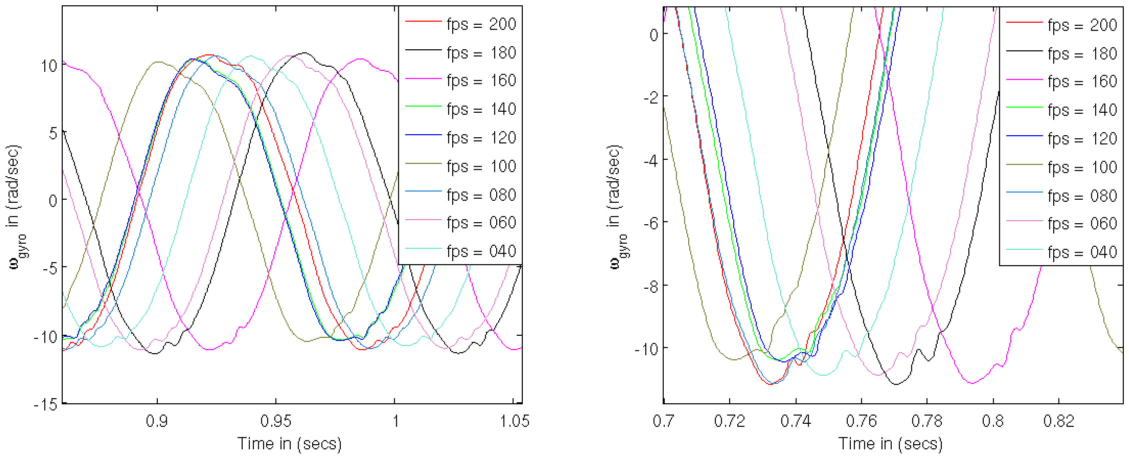
$$\omega(t) = a \sin(f_1 t) \tag{7.11}$$

$$\alpha(t) = a\omega_1 \cos(f_1 t) \tag{7.12}$$

$$\frac{(\alpha(t))^2}{\omega_1^2} + (\omega(t))^2 = a^2 \, . \tag{7.13}$$

a stands for the amplitude of the motion, f_1 stands for the frequency of motion which in this case is 8.6Hz. Variables α and ω denote the angular acceleration and velocity respectively.

A sample raw gyro velocity for this motion is shown in Figure 7.5 obtained when col-



(a) Gyro ground truth

(b) Zoom-in version

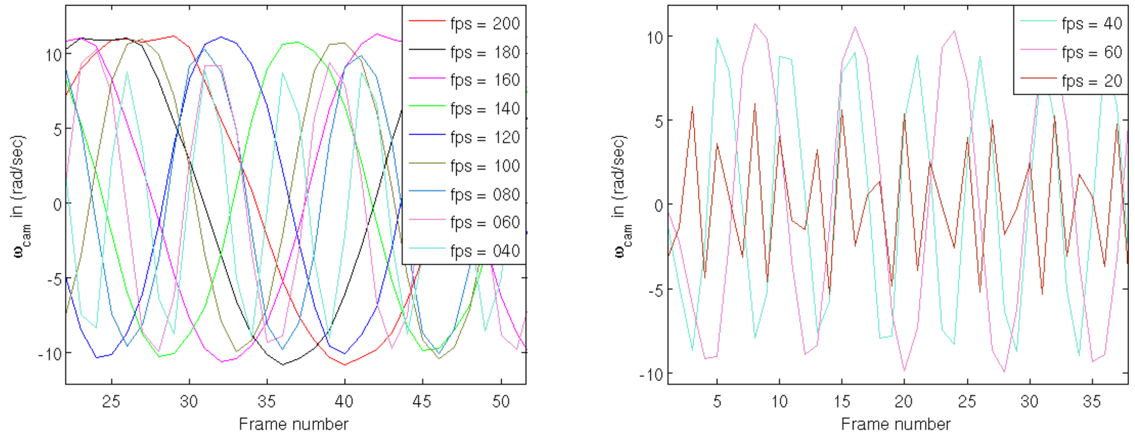
Figure 7.6: Raw gyro plots shown for different frame-rates. The servo makes 40 sweeps back and forth in total. Right plot also highlights the high frequencies captured by this high bandwidth gyro.

lecting data at 200Hz with shutter time set to half the maximum allowed. Ground truth velocities for all frame-rates are plotted in Figure 7.6 together with the zoomed-in versions to also show the high frequencies captured by the gyro. The corresponding acceleration and velocity plots are shown in Figure 7.9(a). The servo oscillation generates a near sine wave motion. However, the locus we obtain as shown in Figure 7.9(a) is a rather titled ellipse. The tilt can be explained by the fact that the sine wave is not a perfect sine wave and that it is modulated by a very low frequency carrier wave which can also be represented by a sine wave *i.e.*

$$v(t) = a\cos(kf_2t)\sin(f_1t) \ . \tag{7.14}$$

The estimates of angular velocity from images are shown in Figure 7.7 for different frame-rates. In particular, Figure 7.7(b) shows how tracking degrades as frame-rate is decreased. The performance suffers the most for low frame-rates where fast motion means motion blur and less image overlap.

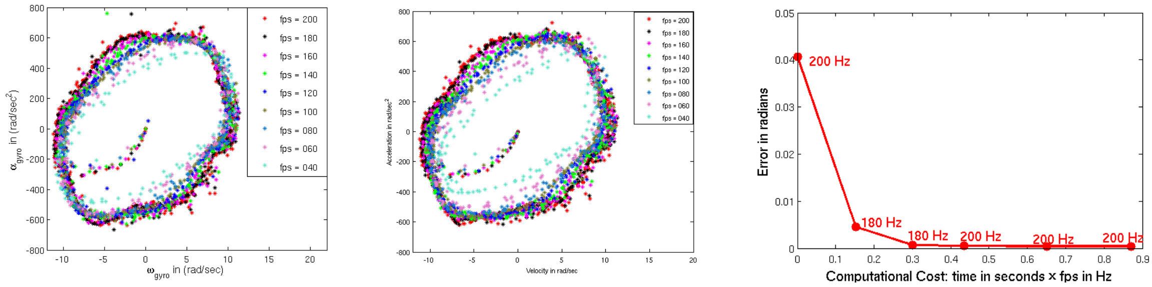
Finally, we plot the Pareto Front of the optimal frame-rates for this motion and observe that it is only high frame-rates 180–200Hz that figure in the plot and therefore suitable for such fast motion tracking. This is not very surprising as one would expect only high frame-rates to appear in the graph for fast motion.



(a) Angular velocity from images.

(b) Velocity at low frame-rates.

Figure 7.7: Velocity estimates returned by pure vision system plotted for different frame-rates. Left plot shows all the different frame-rates used while the Right plot highlights the tracking results for low frame-rates. Subsequent lower frame-rates tend to underestimate the camera velocity due to large baseline and motion blur observed in images and at even lower frame-rates e.g. 20Hz, tracking completely *fails*.



(a) Gyro

(b) Camera

(c) Pareto Front

Figure 7.8: Acceleration vs Velocity graph for camera oscillating back and forth on a servo motor with a frequency of 8.6Hz. Gyro velocity (a) put side by side against the dynamics returned by algorithm (b). The corresponding Pareto Fronts are shown in (c)

Camera Moving Frequency 4.62Hz

Our next set of experiments focus on nearly half the frequency of fastest motion. Like previous experiments, we again collect data by repeated trails of the same motion profile and running optimisations to align the camera and gyro velocities. The acceleration and velocity profiles of gyro and camera are put side by side in Figure 7.10. Again, we observe that tracking suffers more for low frame-rates. However, we do see a minor improvement in the robustness of slightly higher frame-rates when compared to fastest motion. For instance,

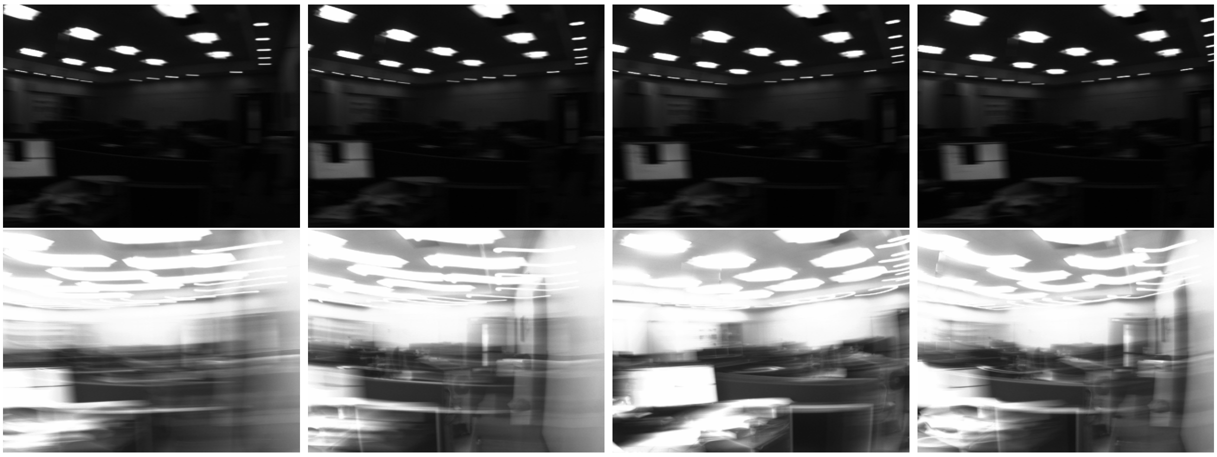
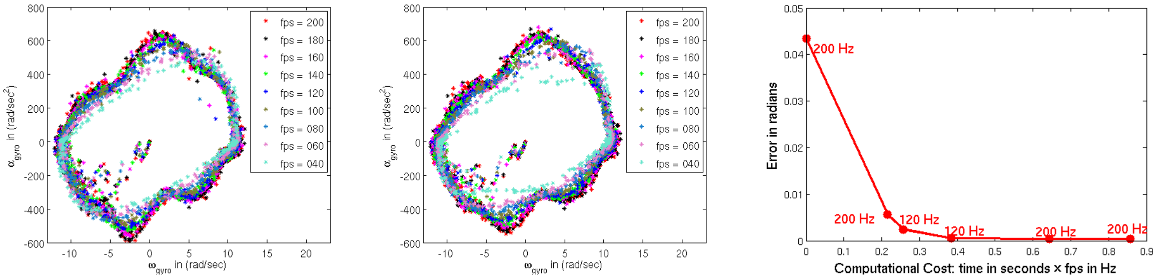


Figure 7.9: Consecutive images obtained at 200Hz (top row) and 20Hz (bottom row) for this motion. Such fast motion introduces motion blur even at 200Hz which is more evident on the monitor screen. Far more importantly, low frame-rates *e.g.* 20Hz not only have significant motion blur but also large displacement which makes it hard to register images and often leads to tracking failures at these frame-rates.

40Hz in this motion seems to perform better simply because of the rapidity of the motion that is reduced when the frequency is halved.



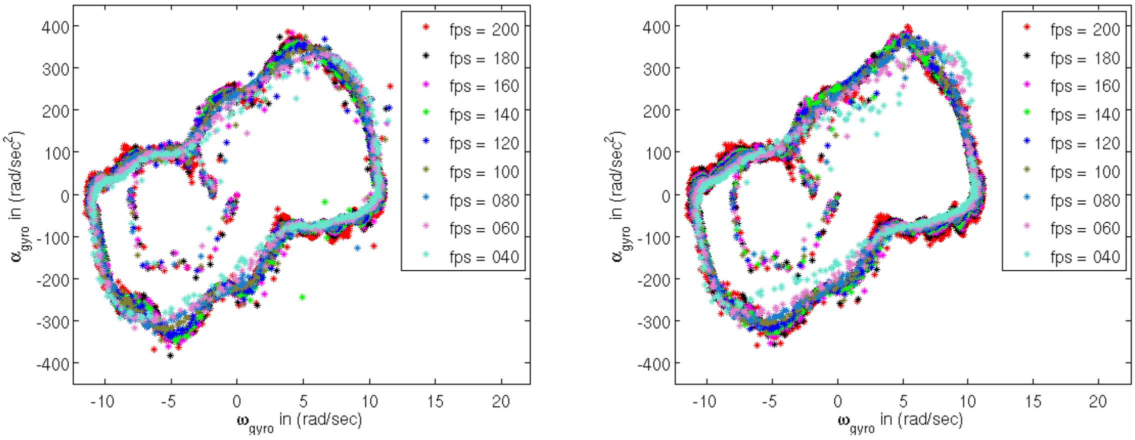
(a) Gyro (b) Camera (c) Pareto Front

Figure 7.10: Acceleration vs velocity graph for gyro and camera plotted next to each other. The Pareto Front on the other hand shows a slightly reduced optimal frame-rates showing how working on extreme high frame-rates does not necessarily lead to optimal use of computational budget.

The Pareto Fronts for this motion show a predictable pattern that with slow motion the optimal frame-rates should switch to slightly lower values. Therefore, these results chime well with the results from synthetic framework. Interestingly, the optimal frame-rates are again higher than the range of standard frame-rates used most often in tracking.

Camera Motion Frequency 2.48Hz

The last set of experiments halved the frequency even further leading to a even slower motion of frequency 2.48Hz. The consequence of working at such motions is that the optimal frame-rates tend to fall down to even lower values. For instance, Figure 7.12 shows how the optimal frame-rates transition from 200Hz down to 140Hz.



(a) Gyro

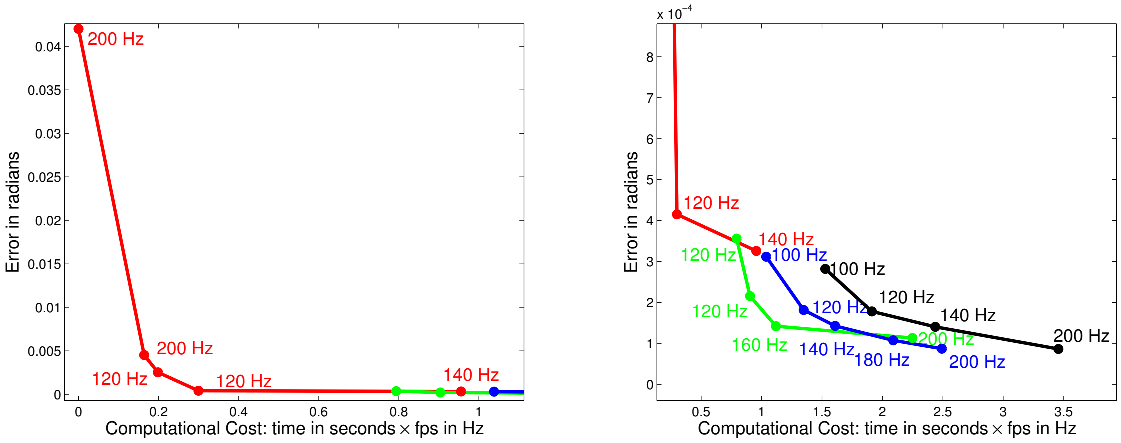
(b) Camera

Figure 7.11: Acceleration vs velocity graph for gyro and camera plotted next to each other for shutter time set to maximum.

It is also interesting to observe in Figure 7.11 how robustness increases even at lower frame-rates as the motion becomes slower.

7.6 Robustness

We turn our attention to the third metric, robustness, which the synthetic framework does not allow immediately to evaluate due to the huge timing demands involved in rendering the images. A real data collection on the other hand greatly facilities the task. For our real-experiments we collect data for more than one minute and this means that for high frame-rates the number of images obtained become very large. This would have taken huge rendering effort to generate this in synthetic framework which is why real data collection simplifies the evaluation of robustness . We characterise the motion below and give insights into the results next.



(a) Lowest Resolution

(b) All Resolutions

Figure 7.12: Pareto Front of the error and computational cost for (a) Lowest resolution and (b) All resolutions. Although we have mentioned previously that we have only compared the results at the lowest resolution, a plot for all resolutions in only to ascertain the fact that increasing image resolution for pure rotation cases only leads to diminishing returns.

7.6.1 Characterisation of Motion

To fully understand the limits of tracking and dynamics of motion where a tracker can easily perform without failures, it is imperative that dynamics contains a variety of motions. The operating region of a tracker can be then found where it is always expected to work.

We collect data for the trajectory that has enough variations in the motion. The servo is commanded to oscillate back and forth with varying frequency and velocities and the same motion profile is repeated for collecting data at frame-rates. Again, we split the experiments to understand the effects of slow as well as fast motion. Our fast motion has the velocities reaching 11 rad/sec and accelerations, 600 rad/sec^2. On the other hand, our slow motion contains a momentary shake in between the otherwise low accelerations of 200 rad/sec^2 and similar velocities as in fast motion, in the trajectory.

The camera velocity profile obtained by running the tracking on the corresponding images is registered with the gyro velocity (ref. Section 7.4) via non-linear least squares optimisation. Registered velocity profiles are compared against the corresponding ground truth velocity obtained from the gyro to obtain error. We use a threshold to decide whether the tracking between the frames was successful or not. Robustness is then simply the frequency of successful tracking instances that occurred in the sequence. The acceleration vs velocity graphs are overlayed with the colour-coded value denoting the error against the corresponding ground truth. It is worth mentioning that we have run the tracker with bright lighting conditions. The results for varying lighting conditions follow the trends we observe in other experiments *i.e.* with a decrease in scene lighting the performance of high frame-rates tends to degrade.

7.6.2 What do We Interpret from the Graphs?

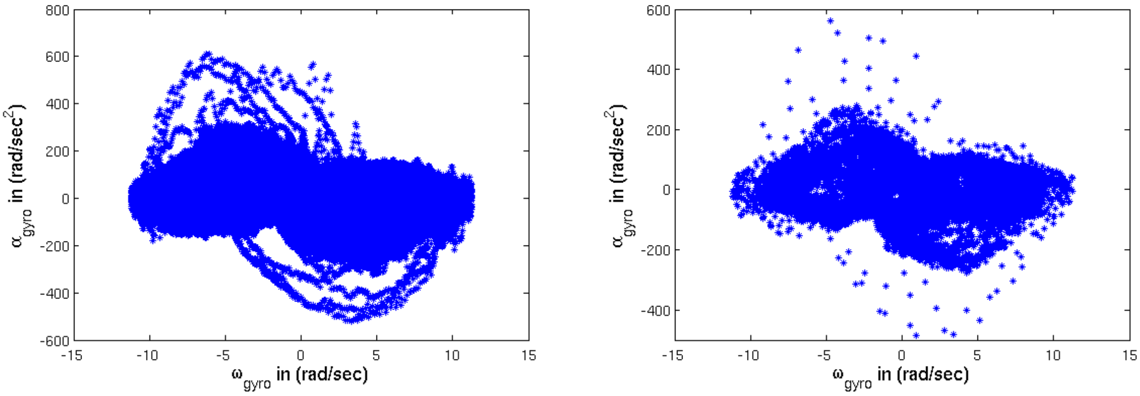
We throw insights into the robustness experiments and examine how high frame-rates often lead to more robust tracking due to the small motion assumption and linearisations that hold more valid as frame-rate is pushed higher. Before expanding on the results, we would like to state that the robustness is evaluated only at the highest resolution used. We begin with results from slow motion first.

Slow motion

Our slow motion trajectory has camera moving back and forth for pure rotation with considerably small velocities. The trajectory begins with small velocity sine wave of low frequency. Gradually this frequency of the oscillation is increased and a momentary jerky motion is thrown in to understand how tracking behaves in this part of otherwise slowly varying trajectory. The motion then resumes to the low frequency sine wave again to end a nearly one minute long trajectory. This motion profile is repeated for all different frame-rates we have previously experimented with, to collect data. Figure 7.13 show the acceleration and velocity plots for the motion and Figure 7.15 show sample images taken at 200Hz and 20Hz. This results in total of 14,531 images for 200Hz and 1,354 for 20Hz. Looking at the number of images, it is clear that this data collection would not have been possible with synthetic framework.

Figure 7.17—7.21 show robustness plots with colour coded error values. The jerky motion in the trajectory shows as arcs away from the densely clustered acceleration and velocity data points. Increased robustness at high frame-rates 100–200Hz is evident from the graphs where the tracker works without any signs of failures. Tracking starts to suffer as frame-rate is decreased further down but it still seems to perform without any gross failures. It is from 40–20Hz we see that the tracking tends to get affected more due to fast motion and motion blur that arise in low frame-rate sequences.

It is interesting to observe that the parts of the trajectory where the motion is accelerating,



(a) ~ 4KHz

(b) 200Hz

Figure 7.13: **Slow Motion:** Acceleration vs Velocity profile of the motion. The highly sampled motion at a frequency of nearly 4KHz put aside with the motion resampled at 200Hz. The high frequency signal is first filtered using a low pass filter and then downsampled to avoid any aliasing effects.

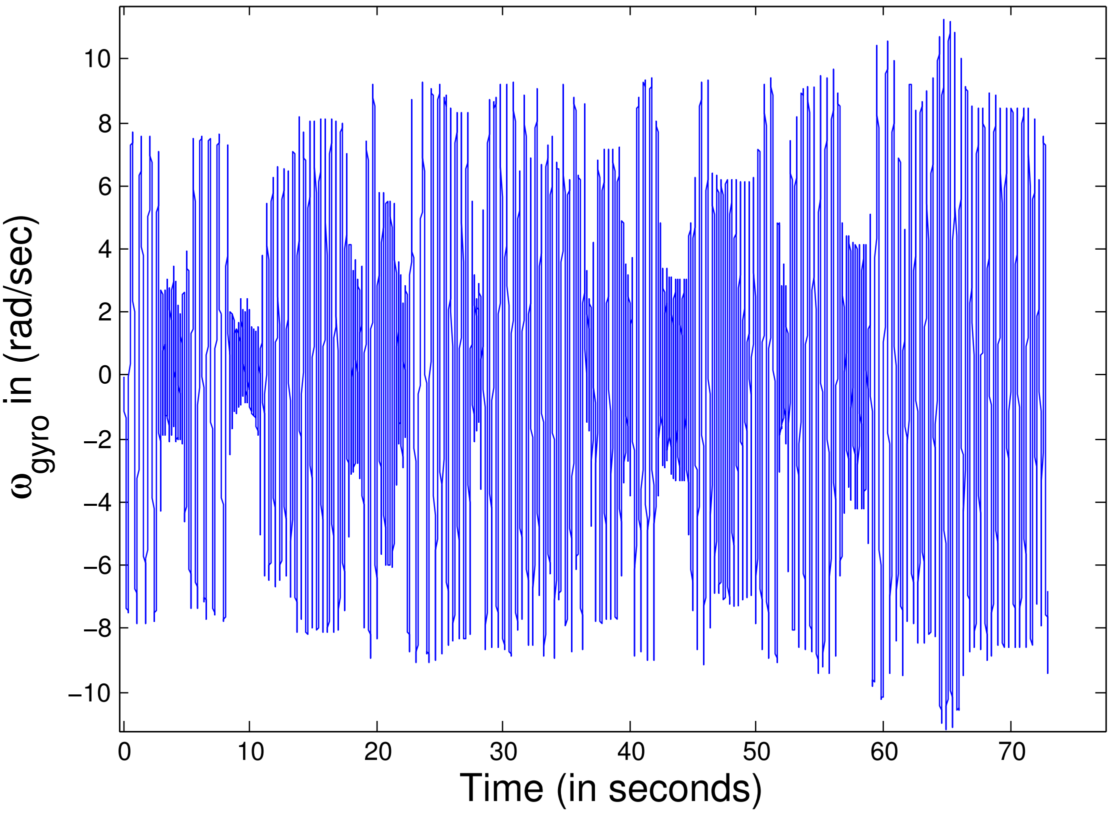


Figure 7.14: The trajectory of the slow motion which we have used in our experiments.

the tracking tends to suffers more. It is due to the fact that acceleration brings inconsistent blur between two consecutive frames and therefore there is no unique transformation that

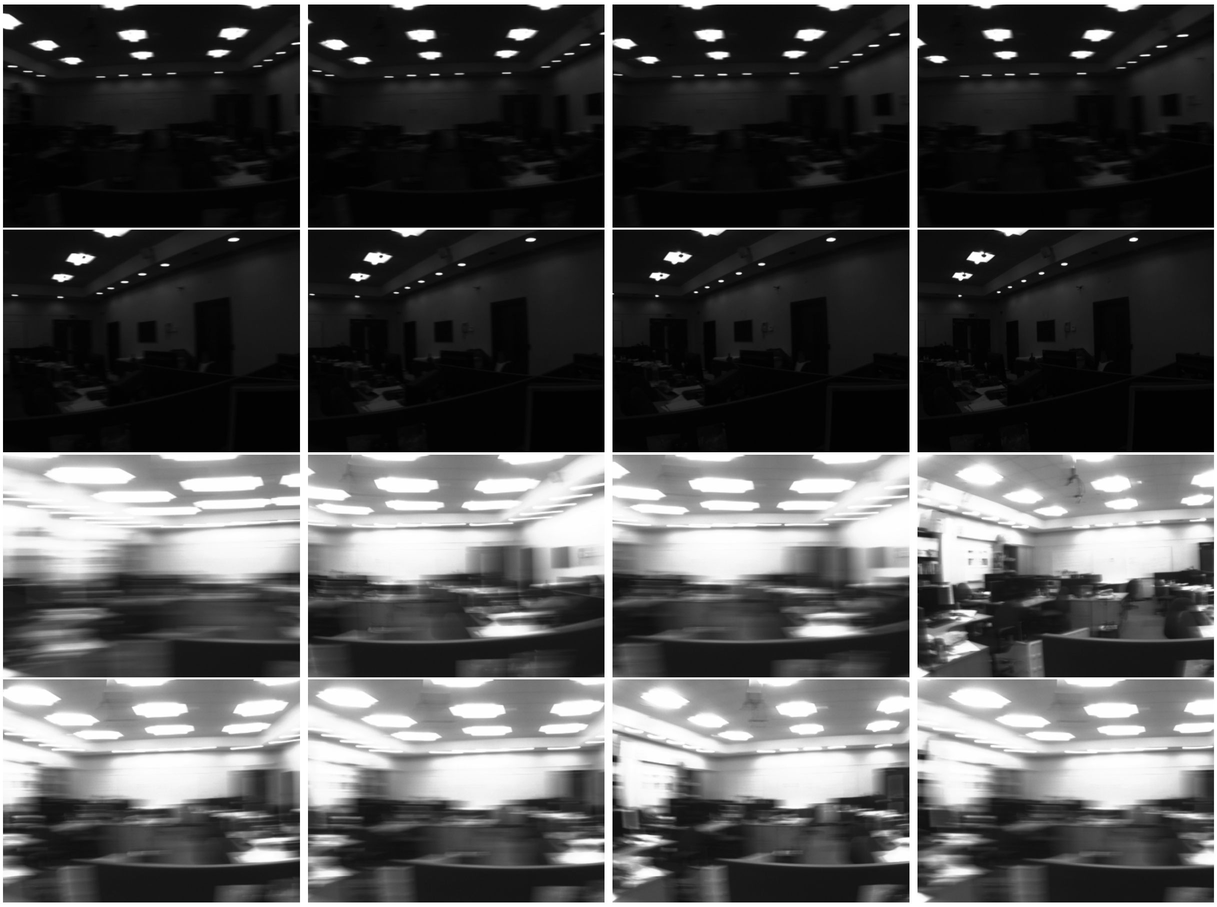
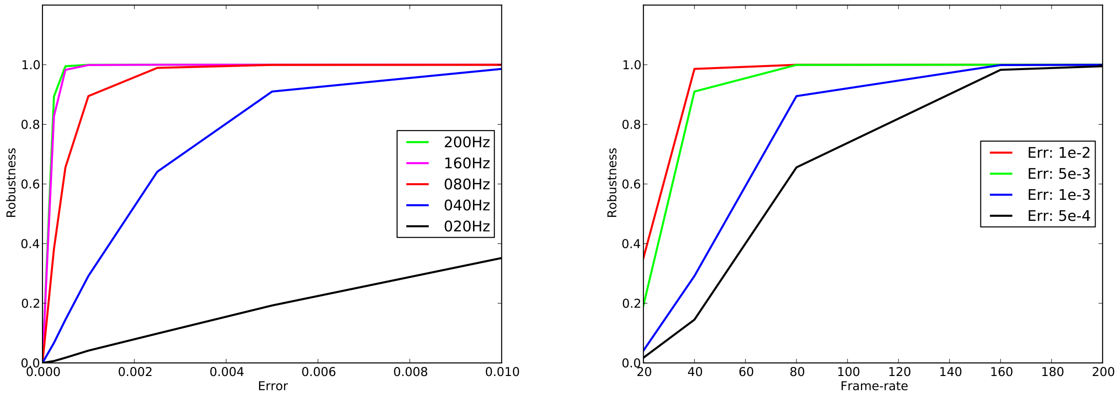


Figure 7.15: Sequence of images taken at 200Hz (top two rows) and 20Hz (bottom two rows) for slow motion robustness experiment. All images are captured with shutter time set to half the maximum allowed. If 200Hz images are dark, motion blur affects tracking if run at 20Hz.

registers two frames. We expand more on this in subsection 7.6.3.

Fast motion

As mentioned earlier, the accelerations in this motion reach as high as 600 rad/sec^2 compared to small motion. The notable difference is the way tracker behaves when the motion becomes too fast. The servo is commanded to again move back and forth however with a faster angular velocity and frequency than before. We collect data again performing repeated trails of the same motion profile for different frame-rates. To highlight the main bits of the results, we have only experimented with frame rates, fps $\in$ {20,40,80,160,200} for fast



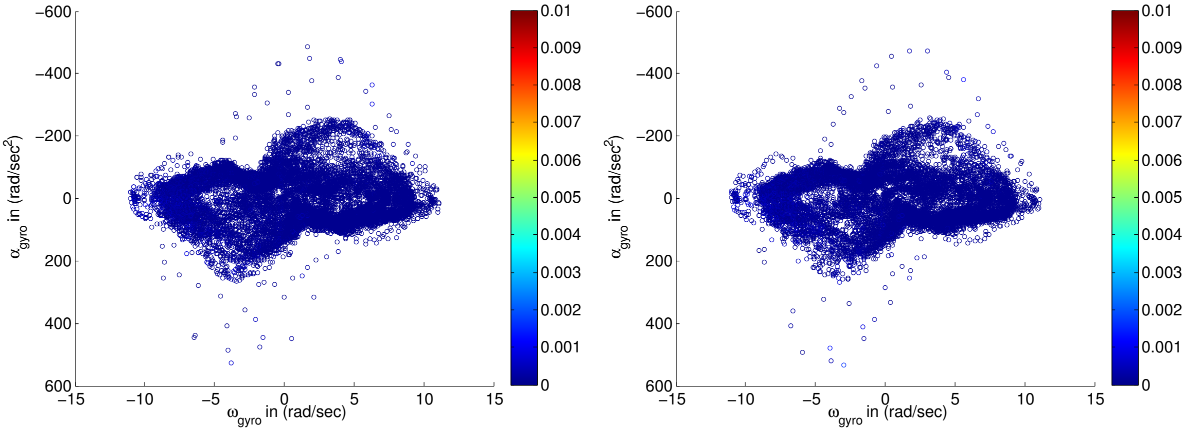
(a) Robustness vs Error

(b) Robustness vs Frame-rate

Figure 7.16: Robustness vs Error and Robustness vs Frame-rate graphs. As error threshold is increased, the tolerence in accepting the tracker to give more erroneous results is also increased leading to increase in the accepted robustness. However, tight thresholds see a decrease in robustness making tracker classified to be failing more often. (b) shows a different way of representing the same information, this time with frame-rate as the slider on x-axis.

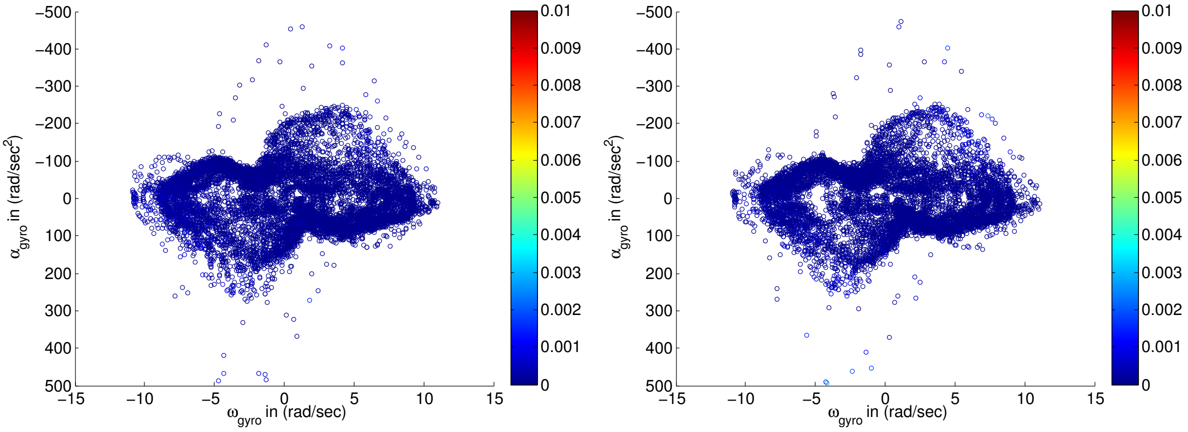
motion.

We observe that at low frame-rates, it is only possible to run the tracker successfully for when the camera is moving with a relatively small velocity. Tracking tends to increasingly suffer as visual conditions deteriorate with increasing camera velocity that induces motion blur and occlusions in the images at these frame-rates. In our experiments, tracking, as shown in Figure 7.24, at 20Hz works only in a very narrow operating region near to zero velocity and very small acceleration. It breaks more often in regions of high velocity and small acceleration (mostly the ends of the longitude in the graph) as well as in regions of high acceleration and low velocity (ends of the latitude). Increasing the frame-rate to 40Hz shows an improvement in the robustness in the regions of high velocity and very small acceleration while regions corresponding to high acceleration and low velocity still remain the failure modes of the tracker. Further increasing the frame-rate to 80Hz (Figure 7.25 (c)) shows a remarkable improvement in the tracker performance where operating region expands to an relatively larger area. Lastly, frame-rates 160Hz (Figure 7.25(d)) and 200Hz (Figure 7.26(b)) clearly show that such high frame-rates bring the tracker to a regime where the operating region covers the whole area of the curve and that the robustness is nearly 100 percent. Pushing the frame-rate to even higher values (until a certain point) will only affirm the fact that tracker works all the time, provided lighting conditions allow the tracker



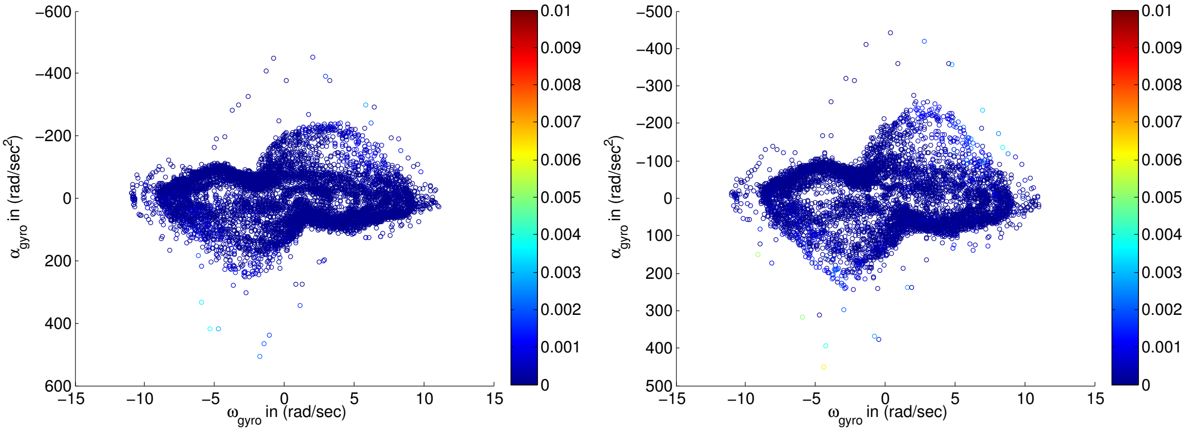
(a) 200Hz (b) 180Hz

Figure 7.17: Robustness graphs for higher frame-rates 180–200Hz. Tracking seems to pretty much work all the time as the graphs suggest.



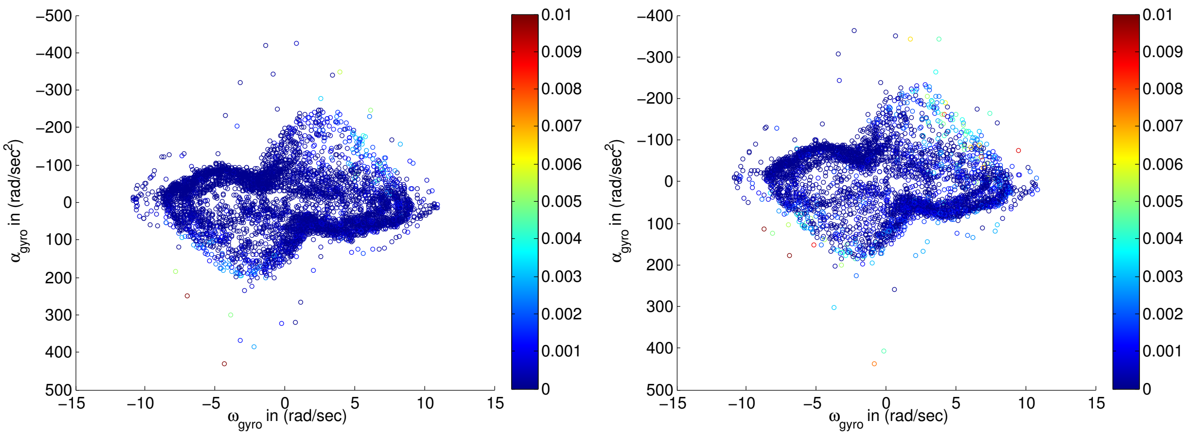
(c) 160Hz (d) 140Hz

Figure 7.18: Robustness graphs for frame-rates from 140–160Hz. Tracking still works without any signs of failures.



(e) 120Hz (f) 100Hz

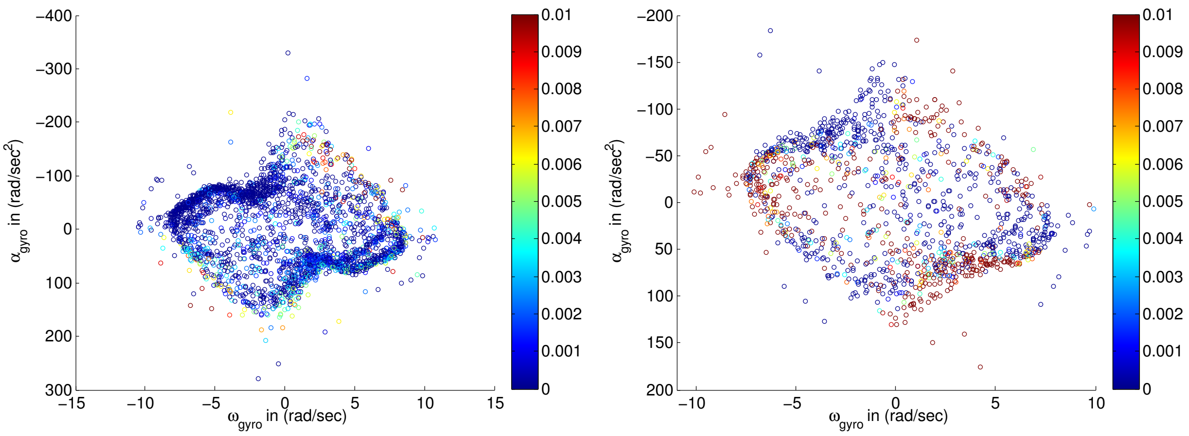
Figure 7.19: Robustness graphs show similar trends appear for frame-rates from 100–120Hz for slow motion and that tracking still seems robust enough even at these frame-rates.



(g) 80Hz

(h) 60Hz

Figure 7.20: Tracking at frame-rates 80–60Hz now tends to suffer at the jerky motion profile that is purposely added to the otherwise slowly varying small motion.



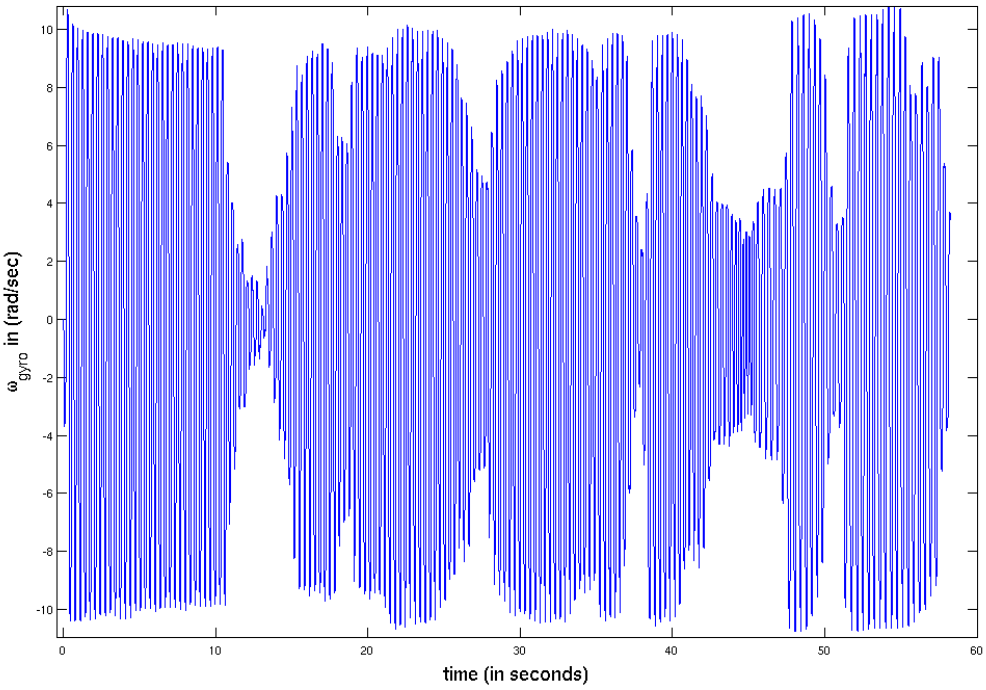
(i) 40Hz

(j) 20Hz

Figure 7.21: The slow motion allows for the camera tracker to work most of the time for all the high frame-rates. It is only at 40Hz and 20Hz we observe tracker beginning to break.

to work without failures.

We also show plots in Figure 7.27 on how robustness varies with the error threshold applied to decide the success or failure of tracking. Remarkable is the improvement in the robustness both as a function of frame-rate and increase in the error threshold. The increased threshold allows for more room for deviation of estimated velocity from the ground truth. Particularly, Figure 7.27 (a) summarises how robustness rapidly improves as the frame-rate is increased from 20Hz to 40Hz. A further increase in the frame-rate to 80Hz, shows even



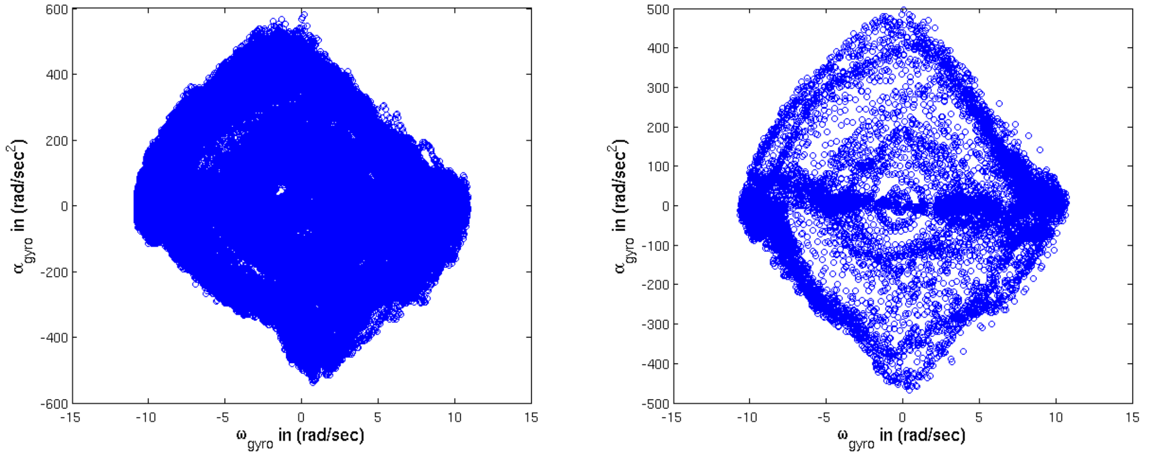
Gyro angular velocity profile for fast motion.

Figure 7.22: Angular velocity profile of gyro as it moves back and forth with varying degrees and frequencies of motion. The gyro starts from rest and oscillates with a peak velocity of nearly 10 rad/sec. It is then brought back to rest slowly and then made to move in varying oscillatory motion profiles with varying peak velocities ranging from 11 rad/sec down to nearly rest. The whole process is repeated for roughly a minute. The velocity profile shown above is for when camera was capturing images at 200Hz.

more improvement and high frame-rates of the order of 160Hz and 200Hz bring the tracker in a regime where it works all the time for the given dynamics we experimented. It is important to mention here that the underlying assumption in this experiment is that light settings still allow the tracker to work at as high frame-rates as 200Hz.

7.6.3 Why does the Tracker Fail?

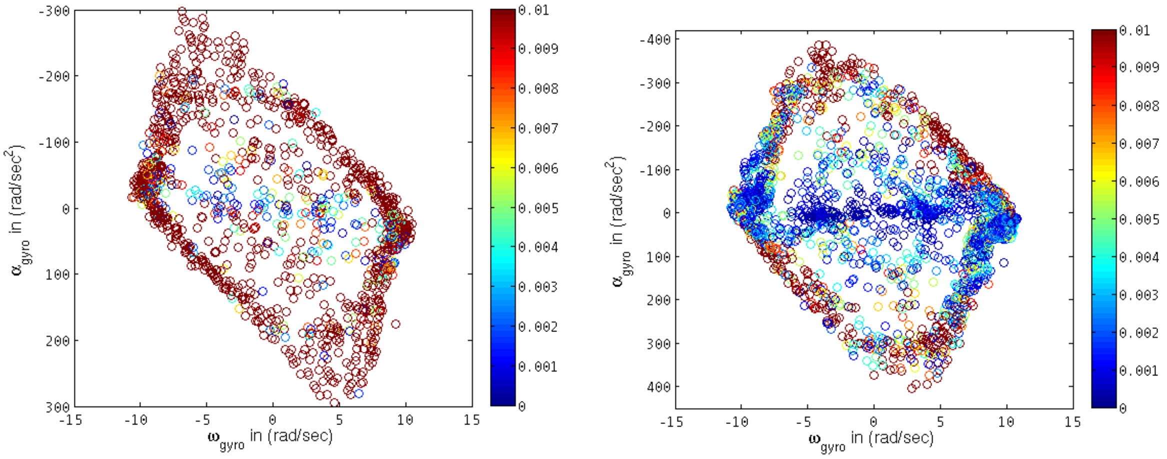
A tracker is more susceptible to failures if the camera is moving with velocity higher than the sampling rate of the camera image acquisition. This happens more often at low frame-rates. Such fast motion introduces motion blur which is detrimental to the direct tracking algorithm that relies on per-pixel image gradients to align the images. In our experiments,



(a) ~ 4KHz

(b) 200Hz

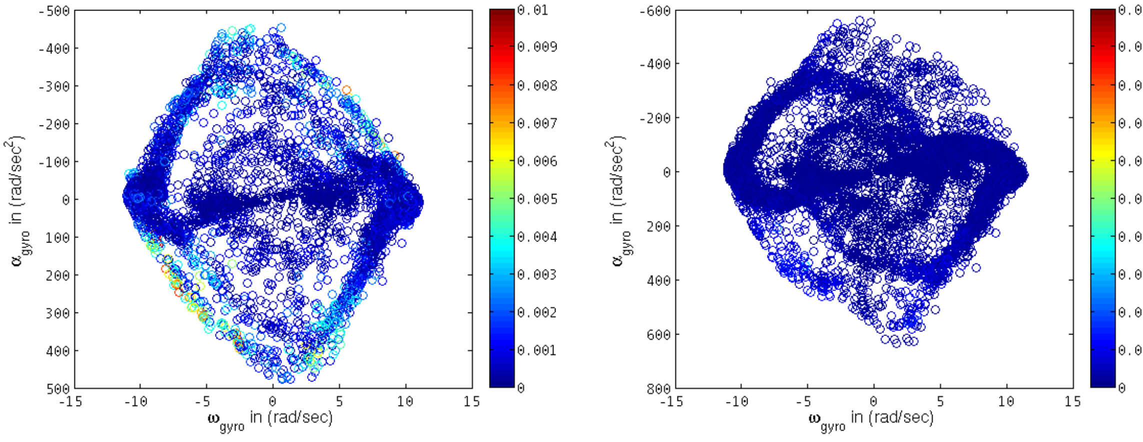
Figure 7.23: Acceleration vs Velocity profile of the motion. The highly sampled motion at a frequency of nearly 4KHz put aside with the motion resampled at 200Hz.



(a) 20Hz

(b) 40Hz

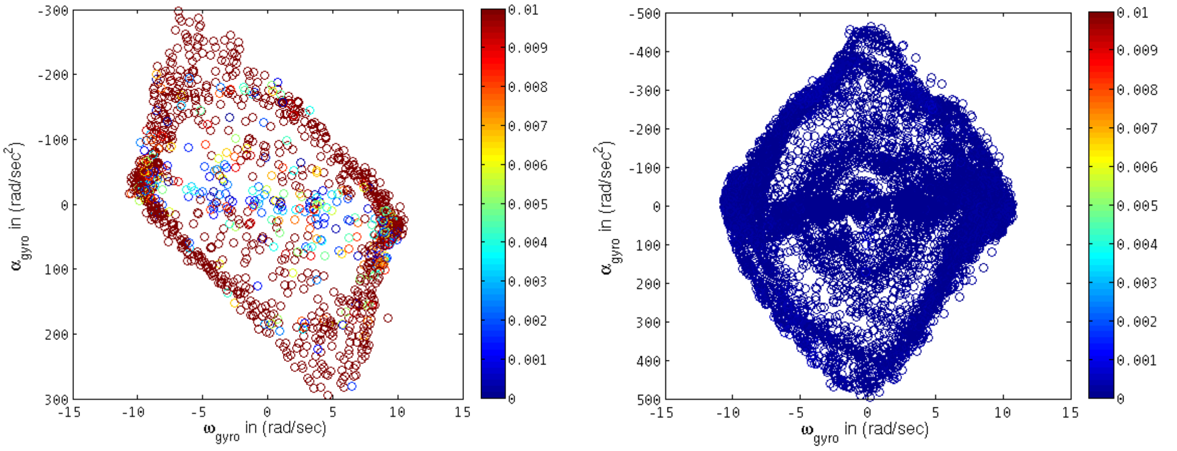
Figure 7.24: Low frame-rates show how robustness tends to suffer when the motion is fast. At 20Hz, the tracker fails nearly all the time except a small operating region very close to zero velocity and zero acceleration. On the other hand, 40Hz show a improvement in robustness however, has the tracker breaking when the acceleration as well as velocity is high are very high.



(c) 80Hz

(d) 160Hz

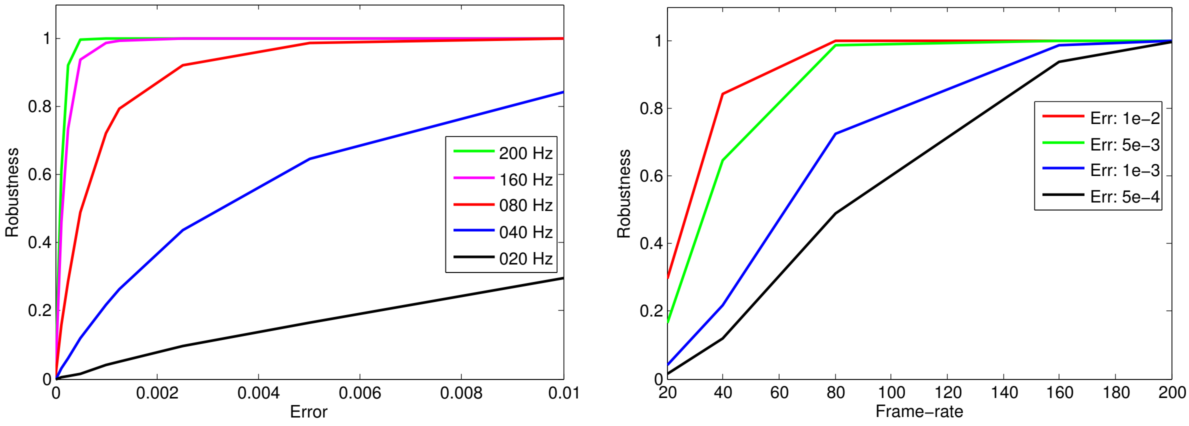
Figure 7.25: Frame-rates 80Hz and 160Hz large improvement in tracking that 160Hz it can be assumed that the tracker works without any signs of failures at all.



(a) 20Hz

(b) 200Hz

Figure 7.26: Direct comparison of tracking performance at 20Hz against 200Hz. It is remarkable how tracking robustness improves as frame-rate is increased to as high as 200Hz.



(a) Robustness vs Error

(b) Robustness vs Frame-rate

Figure 7.27: Robustness vs Error and Robustness vs Frame-rate graphs. As error threshold is increased, the tolerence in accepting the tracker to give more erroneous results is also increased leading to increase in the accepted robustness. However, tight thresholds see a decrease in robustness making tracker classified to be failing more often. (b) shows a different way of representing the same information, this time with frame-rate as the slider on x-axis.

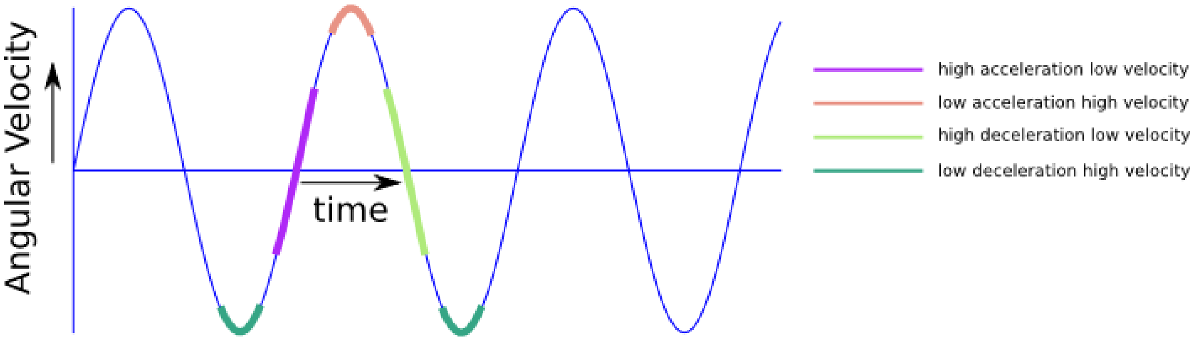


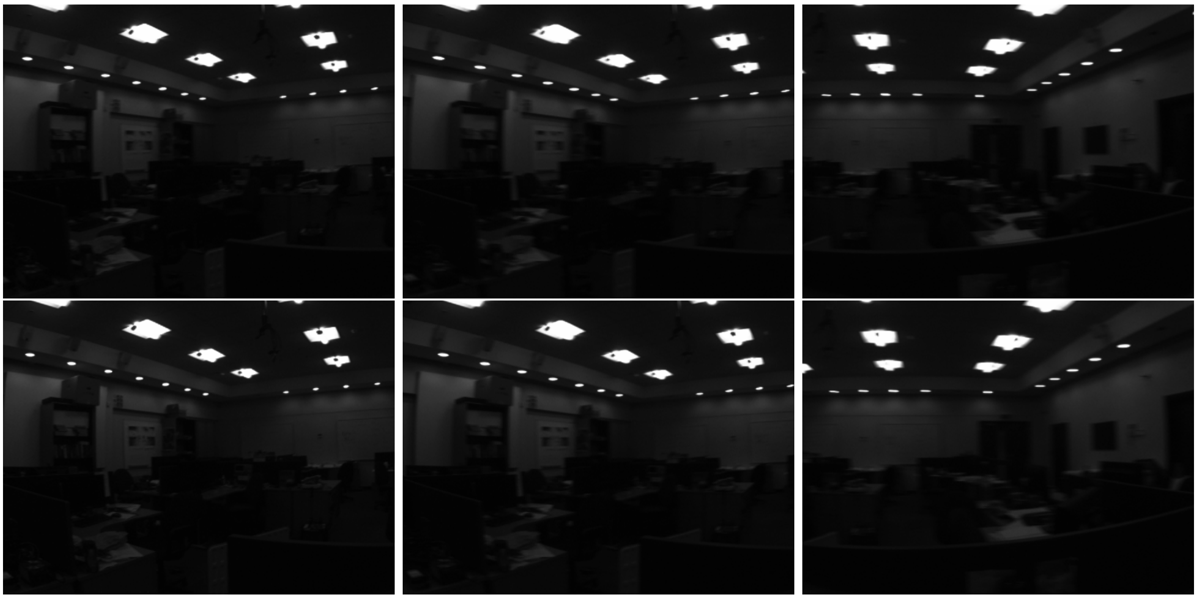
Figure 7.28: Velocity profile broken into four different regions. As the camera goes through different sections of the trajectory it passes regions of zero acceleration and zero velocity. These regions are particularly interesting. The region of zero acceleration is when the camera achieves the highest angular velocity. This is where two consecutive frames have similar motion blur while regions of zero angular velocity means that acceletaion is highest. This is where the consecutive images have different motion blur.

the oscillatory motion of the camera means that it travels through various different regions of varying acceleration and velocity. Figure 7.28 shows the velocity profile broken into four different regions. As a result, we observe that consecutive frames have two different kinds of blur, namely:

1. **Consistent Motion Blur:** This is when the camera is moving with high velocity and there is very little or zero acceleration/decelaration.
2. **Varying Motion Blur:** This arises when the camera is accelerating/decelerating.

Further, each experiment is sub-divided into regions of contant velocity (where servo achieves it's maximum velocity and acceleration is zero) and constant acceleration/deceleration (where servo begins to move or comes to rest). In regions of constant velocity we observe similar blur between consecutive pair of images while in regions of constant acceleration there is uneven blur between two consecutive images. Images obtained at high sampling (high frame-rate) remarkably improve the tracking robustness in regions of constant acceleration. While low sampling leads to matching an unblurry image with a blurry image which does not give a unique registration between two images and hence leads to tracking failures at those points.

Figure 7.30 shows a part of the trajectory with the pyramid number that yields the smallest warped error overlaid. An interesting pattern that emerges out from this is that at points of high acceleration and low velocity, the algorithm provides best results at smaller resolutions.



(a) 3500–3501 (b) 9170–9171 (c) 4498–4499

Figure 7.29: We show three samples of consecutive frames from the trajectory that achieve best alignment only at 160×120 resolution due to inconsistent image blur. The top row shows different reference images that are aligned against the live images shown in the bottom row. The numbers at the bottom are the frame numbers of the images that are aligned.

Remember that this leads to different motion blur in two consecutive images. This difference in the image blur reduces as images are downsampled which is why we see tracking often produces smaller errors at lower resolutions for such images. On the other hand points of high velocity and small acceleration yield images that have very similar motion blur which is why the algorithm gives smaller warped errors at higher resolutions. Figure 7.29 shows three samples of images where the resolution 160×120 gives the lowest warped error.

7.7 Summary

In this chapter, we validated our conclusions from synthetic experiments with carefully controlled real world experiment involving pure rotation. The results very well chime with the synthetic experiments. Additionally, we provide quantitative results for robustness of a tracker with two experiments with contrasting motion profiles aimed to study how robustness varies with the degree of motion. A tracker is more prone to failures if consecutive images have inconsistent motion blur which happens quite often when the camera is ac-

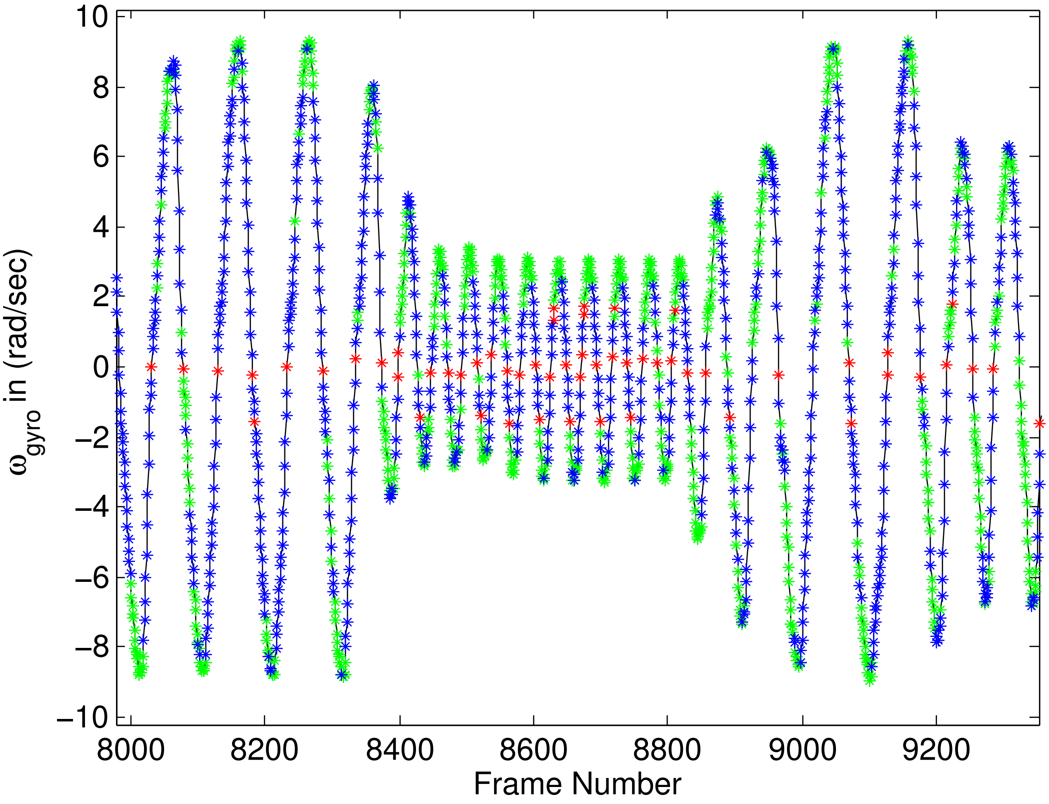


Figure 7.30: Angular velocity overlaid with different pyramid levels the algorithm returns the least warped error. Red colour is the lowest resolution in the pyramid (80×60), blue is 160×120 and green is 320×240. An interesting pattern emerges out from ths plot. That the lowest possible error is achieved at the peaks of the graph working at higher resolutions (remember that this is when the acceleration is near to zero) while red and blue denote the choice of lower resolution when the camera is accelerating. Particularly, when the camera is passing through region of zero angular velocity, it has the highest acceleration and and therefore, the motion blur in one of the images is more than the other at the highest resolution. As a result, these images cannot be registered uniquely at this resolution and that a lower resolution where the difference in the motion blur as well as the image overlap decreases, lowest error can be obtained. The angular velocity is obtained at 200Hz for slow motion experiment.

celerating in contrast to the constant velocity motion which induces same amount of blur. This forms the basis of our explanation for how the robustness decreases as the frame-rate is lowered down.

Chapter 8

Conclusions and Future Directions

Contents

8.1 What do We Learn from this Thesis?

This thesis has explored the possibilities of camera tracking at frame-rates higher than the range usually assumed to be standard for any real-time operation. Three different metrics namely accuracy, robustness and computational budget are evaluated to rate the performance of different frame-rates. An interesting revelation that has also come out is the fundamental connections between frame-rate and image resolution.

A thorough evaluation was performed with datasets generated with our synthetic framework which replicates carefully the real camera image acquisition process as well as validating these with a pure one dimensional rotation motion in real world. Our synthetic dataset achieves a level of photorealism beyond that in most well-known datasets for image motion analysis [Baker et al., 2011], [Geiger et al., 2012], [Butler et al., 2012] and [Peris et al., 2012] where no special care was taken to model the real camera acquisition process that adds many real-world artefacts in the images *e.g.* image noise depending upon scene lighting.

The main conclusions of our evaluation advocate the use of a combination of high frame-rate and low image resolution for pure camera tracking operating under low computational budgets and normal office lighting conditions. A few iterations at high frame-rate and low resolution seems the optimal choice that yields the most accurate results under stringent computational budgets. An increase in the computational budget leads to a transition towards a combination of lower frame-rate but higher resolution as the optimal choice *i.e.* better accuracy can be achieved with a rather lower frame-rate but higher resolution under that computational budget. A similar pattern follows for increasing resolutions. A further increase in the budget for a given resolution only advocates the use of high frame-rate to optimally use that budget and yield more accurate results.

Another interesting insight our evaluation provides is the effect of scene lighting on the performance of camera tracking. We find that degradation of scene lighting conditions leads to a transition towards lower optimal frame-rates when compared to normal lighting conditions where a combination of frame-rate and resolution is unchanged.

We also see via a series of real experiments with gyro mounted on a camera, how high frame-rate leads to more robust tracking. The evaluation of this metric in synthetic experiments was not feasible due to the cumbersome renderings of the order of thousands of images that we needed to obtain meaningful statistics.

As the motion of the camera slows down, we see that choices of low frame-rate emerge as optimal at all resolutions. This is expected, as one would guess that there is no need to go to higher frame-rates when similar levels of accuracy can be achieved with low frame-rates.

The dependence of our results on varying camera motion is also manifested in our experiments on camera tracking on pure rotation and joint translation and rotation motion *i.e.* for pure 1D rotation resolution does not necessarily help while joint translation and rotation motion higher level of accuracy is achieved only when the resolution is increased.

8.2 Future Directions

8.2.1 A Full Theoretical Understanding

The conclusions obtained from this work have largely been through an empirical and experimental evaluation. In the future, we aim to understand this work from a theoretical perspective to provide analytically the best combination of frame-rate and image resolution given the camera image acquisition model, scene lighting and statistics of depth and texture of the scene. For instance, using a simple planar surface model together with a point light source and lambertian properties of the texture on the scene, we write the intensity at given pixel location as:

$$
\begin{aligned}
\mathrm{I}_{\mathrm{p}} &= f_{CRF}\left(\int_{t_o}^{t_c}\left(\sum_{m\in\mathcal{L}} k_d(\mathrm{P}(t))\cos\theta^{m}_{\mathrm{P}(t)} i_{m,d}) + n_s\right)dt\right) && (8.1)\\
\mathrm{P}(t) &= \mathrm{t}(t) + u\mathrm{v}(t) && (8.2)\\
\mathrm{v} &= (\mathrm{K}[\mathrm{R}(t)|\mathrm{t}(t)])^{-1}\mathrm{p},\ \ [\mathrm{R}(t)|\mathrm{t}(t)] \backsim \mathrm{T}(t) && (8.3)\\
u &= \frac{-\mathrm{t}(t)\cdot n + d}{\mathrm{v}\cdot n} && (8.4)\\
\mathrm{T}(t) &= \exp^{\int_{t_o}^{t}\xi(t)dt}\mathrm{T}(t_o)\ . && (8.5)
\end{aligned}
$$

Here variable k_d[1] denotes the diffusion constant which varies with the location on the surface and $\theta^m(\mathrm{P}(t))$ represents the incident angle a ray coming from the light source makes with the normal at point $\mathrm{P}(t)$. The point $\mathrm{P}(t)$ can be obtained as the intersection of the ray from camera at a pixel location p with the normal n of the surface. As the camera moves on a trajectory, the continuous pose at any time instant t, $\mathrm{T}(t)$, after the shutter is opened at time t_o is via the exponential map given the instantaneous rotational and translation velocities encoded in $\xi(t)$. f_{CRF} denotes the camera response function and d is the parameter related to the plane. This analytical image model governs how a pixel intensity is observed as a function of motion of the camera as well as the surface properties of the scene.

An abstract model for the computational cost of the gradient descent scheme used in image alignment can also be designed. The maximum pixel displacement between the images for the gradient descent to converge in one step can be empirically found for both translation and rotation. A displacement more than this maximum displacement then requires coarse-to-fine pyramidal hierarchy that progressively aligns the images but at the expense of more iterations. Therefore, given the knowledge of camera motion and hence the displacement between the images, it is possible to then calculate the number of iterations it would take to align images.

After the matching between the images has been performed, we can describe the reduction

[1] http://users.eecs.northwestern.edu/~yingwu/teaching/EECS432/Notes/lighting.pdf

in the uncertainty on the camera pose via standard Bayes rule:

$$\frac{1}{\sigma^2_{\mathrm{X|Z}}} = \frac{n_{eff}}{\sigma^2_{\mathrm{Z|X}}(\mathrm{r})} + \frac{1}{\sigma^2_{\mathrm{X}_0}\Delta t} \tag{8.6}$$

$$\frac{1}{\sigma^2_{\mathrm{X|Z}}} = \frac{n_{eff}}{\sigma^2_{\mathrm{Z|X}}(\mathrm{r})} + \frac{f}{\sigma^2_{\mathrm{X}_0}} \tag{8.7}$$

$$\sigma^2_{\mathrm{Z|X}}(\mathrm{r}) = 2^{\mathrm{r}}\sigma^2_{\mathrm{Z|X}}(0)\ . \tag{8.8}$$

$\sigma^2_{\mathrm{X|Z}}$ is the posterior uncertainty, $\sigma^2_{\mathrm{Z|X}}$ is the measurement uncertainty and $\sigma^2_{\mathrm{X}_0}\Delta t$ is the predicted uncertainty that grows with time as modelled by random walk distribution. The inverse of Δt is the frame-rate f. If the alignment is performed at a smaller resolution, the measurement uncertainty grows accordingly as $2^{\mathrm{r}}\sigma^2_{\mathrm{Z|X}}(0)$ where $\sigma^2_{\mathrm{Z|X}}(0)$ is the uncertainty at the highest resolution and r is the resolution level. n_{eff} is the number of pixels that have the residual with the 3σ threshold of the M-estimator used for alignment.

The future directions for this work would be take a deeper look at the 6DoF camera pose using a similar mathematical approach breaking free from these simple assumptions and thinking more about any general surface with not necessarily simple BRDF properties.

8.2.2 Using Sparse Measurements for High Frame-Rate Tracking

Remember that most of the information for image alignment comes from the high intensity gradients present in the image. Therefore, it is worth asking if a camera that provided only changes in the image gradients instead of pixel colour values as done in most of the cameras today, whether we would be able to significantly improve the efficiency of matching? Indeed, ATIS cameras [Posch et al., 2011] and [Posch, 2011] are the next generation cameras that are available today that report the changes in the images as events that arrive asynchronously, are a potential route towards ultra high frame-rate matching.

8.2.3 Joint Analysis of Tracking and Mapping

Another potential future direction would be to couple the tracking with the 3D mapping in a full SLAM framework. Accuracy of tracking and mapping can be jointly analysed as a function of frame-rate and image resolution. Multi-frame stereo analysis has already been done before in [Rumpler et al., 2011] and [Gallup et al., 2008] for baseline (frame-rate) and resolution. This is very desirable in quantifying the accuracy of any SLAM system.

Bibliography

[Adiv, 1985] Adiv, G. (1985). Determining Three-Dimensional Motion and Structure from Optical Flow Generated by Several Moving Objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 4:348–401. 93

[Alexe et al., 2010] Alexe, B., Deselaers, T., and Ferrari, V. (2010). What is an object? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 73–80. IEEE. 19

[Anandan, 1987] Anandan, P. (1987). A Computational Framework and an Algorithm for the Measurement of Visual. Technical report, University of Massachusetts, Amherst, MA, USA. 109

[Anandan, 1989] Anandan, P. (1989). A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision (IJCV)*, 2:283–310. 109

[Armstrong and Zisserman, 1995] Armstrong, M. and Zisserman, A. (1995). Robust Object Tracking. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*. 20

[Arun et al., 1987] Arun, K. S., Huang, T. S., and Blostein, S. D. (1987). Least-squares fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, (5):698–700. 27

[Ayache and Faugeras, 1986] Ayache, N. and Faugeras, O. (1986). Hyper: A new approach for the recognition and positioning of two-dimensional objects. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (1):44–54. 92

[Bailey, 2002] Bailey, T. (2002). *Mobile robot localisation and mapping in extensive outdoor environments*. PhD thesis. 33

[Baker et al., 2003a] Baker, S., Gross, R., and Matthews, I. (2003a). Lucas-Kanade 20 Years On: A Unifying Framework: Part 3. Technical report, Carnegie Mellon University. 36

[Baker et al., 2004a] Baker, S., Gross, R., and Matthews, I. (2004a). Lucas-Kanade 20 Years On: A Unifying Framework: Part 4. Technical report, Carnegie Mellon University. 36

[Baker et al., 2003b] Baker, S., Gross, R., Matthews, I., and Ishikawa, T. (2003b). Lucas-Kanade 20 Years On: A Unifying Framework: Part 2. Technical report, Carnegie Mellon University. 36

[Baker and Matthews, 2004] Baker, S. and Matthews, I. (2004). Lucas-Kanade 20 years on: A unifying framework: Part 1. *International Journal of Computer Vision (IJCV)*, 56(3):221–255. 108

[Baker et al., 2004b] Baker, S., Patel, R., Cheung, G., and Matthews, I. (2004b). Lucas-Kanade 20 Years On: A Unifying Framework: Part 5. Technical report, Carnegie Mellon University. 23, 36, 37, 92, 108

[Baker et al., 2011] Baker, S., Scharstein, D., Lewis, J. P., Roth, S., Black, M. J., and Szeliski, R. (2011). A Database and Evaluation Methodology for Optical Flow. *International Journal of Computer Vision (IJCV)*. 140, 199

[Bayro-Corrochano and Ortegon-Aguilar, 2004] Bayro-Corrochano, E. and Ortegon-Aguilar, J. (2004). Template tracking with lie algebras. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 5183–5188. IEEE. 98

[Benhimane and Malis, 2004] Benhimane, S. and Malis, E. (2004). Real-Time Image-Based Tracking of planes using Efficient Second-order Minimization. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*. 108

[Bergen et al., 1992] Bergen, J. R., Anandan, P., Hanna, K. J., and Hingorani, R. (1992). Hierarchical model-based motion estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 91, 95, 109

[Besl and McKay, 1992] Besl, P. and McKay, N. (1992). A method for Registration of 3D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 14(2):239–256. 28, 33

[Bibby and Reid, 2008] Bibby, C. and Reid, I. (2008). Robust Real-Time Visual Tracking using Pixel-Wise Posteriors. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 22

[Bishop, 2006] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc. 72

[Black, 1992] Black, M. (1992). *Robust incremental optical flow*. PhD thesis, PhD thesis, Yale university. 103

[Black and Anandan, 1991] Black, M. and Anandan, P. (1991). Robust dynamic motion estimation over time. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on*, pages 296–302. IEEE. 103

[Black and Rangarajan, 1996] Black, M. and Rangarajan, A. (1996). On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. *International Journal of Computer Vision*, 19(1):57–91. 103

[Black and Anandan, 1993] Black, M. J. and Anandan, P. (1993). A framework for the robust estimation of optical flow. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 103

[Black et al., 1998] Black, M. J., Sapiro, G., Marimont, D. H., and Heeger, D. (1998). Robust anisotropic diffusion. *IEEE Trans. Image Processing*, 7:421–432. 105

[Botterill et al., 2009] Botterill, T., Mills, S., and Green, R. (2009). New conditional sampling strategies for speeded-up ransac. In *Proceedings of the British Machine Vision Conference (BMVC)*. 35

[Bregler and Malik, 1998] Bregler, C. and Malik, J. (1998). Tracking people with twists and exponential maps. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 8–15, Washington, DC, USA. IEEE Computer Society. 92, 98

[Brooks, 1981] Brooks, R. (1981). Symbolic reasoning among 3-d models and 2-d images. *Artificial intelligence*, 17(1):285–348. 92

[Bruss and Horn, 1983] Bruss, A. R. and Horn, B. K. P. (1983). Passive navigation. *Computer Vision and Image Understanding (CVIU)*, 21(1):3–20. 93

[Butler et al., 2012] Butler, D. J., Wulff, J., Stanley, G. B., and Black, M. J. (2012). A naturalistic open source movie for optical flow evaluation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 611–625. 141, 199

[Calonder et al., 2010] Calonder, M., Lepetit, V., and Fua, P. (2010). Pareto-optimal Dictionaries for Signatures. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 119

[Canelhas, 2012] Canelhas, D. (2012). Scene representation, registration and objectdetection in a truncated signed distance functionrepresentation of 3d space. Master's thesis, Örebro University. 38

[Chli and Davison, 2008] Chli, M. and Davison, A. J. (2008). Active Matching. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 35, 36, 65, 66, 67, 68

[Chli and Davison, 2009a] Chli, M. and Davison, A. J. (2009a). Active Matching for visual tracking. *Robotics and Autonomous Systems*, 57(12):1173 – 1187. Special Issue 'Inside Data Association'. 66, 82, 84

[Chli and Davison, 2009b] Chli, M. and Davison, A. J. (2009b). Automatically and Efficiently Inferring the Hierarchical Structure of Visual Maps. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 71, 72

[Chow and Liu, 1968] Chow, C. K. and Liu, C. N. (1968). Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467. 72

[Chum and Matas, 2002] Chum, O. and Matas, J. (2002). Randomized ransac with td, d test. In *Proceedings of the British Machine Vision Conference (BMVC)*, volume 2, pages 448–457. 35

[Chum and Matas, 2005] Chum, O. and Matas, J. (2005). Matching with prosac-progressive sample consensus. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 220–226. 35, 78

[Chum and Matas, 2008] Chum, O. and Matas, J. (2008). Optimal randomized ransac. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 30(8):1472–1482. 35

[Civera et al., 2009] Civera, J., Grasa, O. G., Davison, A. J., and Montiel, J. M. M. (2009). 1-Point RANSAC for EKF-Based Structure from Motion. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*. 35, 64

[Coffin et al., 2010a] Coffin, C., Kim, S., and Hollerer, T. (2010a). Evaluation of tracking robustness in real time panorama acquisition. In *Virtual Reality Conference (VR), 2010 IEEE*, pages 259–260. IEEE. 117

[Coffin et al., 2010b] Coffin, C., Kim, S., and Hollerer, T. (2010b). A metric for tracking robustness in real-time panorama acquisition. In *International Conference on Artifical Reality and Teleexistence, ICAT*, pages 96–103. IEEE. 117

[Comport et al., 2006] Comport, A., Marchand, E., and Chaumette, F. (2006). Statistically robust 2D visual servoing. *IEEE Transactions on Robotics*, 22(2):415–421. 106

[Comport et al., 2007] Comport, A. I., Malis, E., and Rives, P. (2007). Accurate Quadri-focal Tracking for Robust 3D Visual Odometry. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 26, 54

[Comport et al., 2011] Comport, A. I., Meilland, M., and Rives, P. (2011). An asymmetric real-time dense visual localisation and mapping system. In *Workshop on Live Dense Reconstruction from Moving Cameras at ICCV*. 37

[Cootes et al., 2001] Cootes, T. F., Edwards, G. J., and Taylor, C. J. (2001). Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 23(6):681–685. 92

[Cossairt, 2011] Cossairt, O. (2011). *Tradeoffs and limits in computational imaging*. PhD thesis, COLUMBIA UNIVERSITY. 132

[Curless and Levoy, 1996] Curless, B. and Levoy, M. (1996). A volumetric method for building complex models from range images. In *Proceedings of SIGGRAPH*. 37

[Curless, 1997] Curless, B. L. (1997). *New Methods for Surface Reconstruction from Range Images*. PhD thesis, Stanford University. 37

[Dame and Marchand, 2010] Dame, A. and Marchand, E. (2010). Accurate real-time tracking using mutual information. In *Mixed and Augmented Reality (ISMAR), 2010 9th IEEE International Symposium on*, pages 47–56. IEEE. 96

[Danielsson, 1980] Danielsson, P.-E. (1980). Euclidean distance mapping. *Computer Graphics and image processing*, 14(3):227–248. 37

[Davison, 2003] Davison, A. J. (2003). Real-Time Simultaneous Localisation and Mapping with a Single Camera. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 31, 36, 89, 90

[Davison, 2005] Davison, A. J. (2005). Active Search for Real-Time Vision. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 36, 64

[Davison et al., 2007] Davison, A. J., Molton, N. D., Reid, I., and Stasse, O. (2007). MonoSLAM: Real-Time Single Camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 29(6):1052–1067. 66, 81, 82

[Debevec and Malik, 1997] Debevec, P. and Malik, J. (1997). Recovering high dynamic range radiance maps from photographs. In *Proceedings of SIGGRAPH*. 56, 58, 133

[Dellaert and Collins, 1999] Dellaert, F. and Collins, R. (1999). Fast image-based tracking by selective pixel integration. In *Proceedings of the ICCV Workshop on Frame-Rate Vision*. 96

[Dellaert et al., 1998] Dellaert, F., Thorpe, C., and Thrun, S. (1998). Super-resolved texture tracking of planar surface patches. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, volume 1, pages 197–203. IEEE. 96

[Devernay and Faugeras, 2001] Devernay, F. and Faugeras, O. (2001). Straight lines have to be straight. *Machine Vision and Applications*, 13:14–24. 55

[Drummond and Cipolla, 1999a] Drummond, T. and Cipolla, R. (1999a). Real-time tracking of complex structures with on-line camera calibration. In *Proceedings of the British Machine Vision Conference (BMVC)*, volume 2, pages 574–583. Citeseer. 20

[Drummond and Cipolla, 1999b] Drummond, T. and Cipolla, R. (1999b). Visual tracking and control using Lie algebras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 98

[Drummond et al., 2002] Drummond, T., Society, I. C., and Cipolla, R. (2002). Real-time visual tracking of complex structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 24:932–946. 20, 97

[Dunn et al., 2004] Dunn, E., Olague, G., Lutton, E., and Schoenauer, M. (2004). Pareto optimal sensing strategies for an active vision system. In *Evolutionary Computation, 2004. CEC2004. Congress on*, volume 1, pages 457–463. IEEE. 119

[Dutter and Huber, 1981] Dutter, R. and Huber, P. (1981). Numerical methods for the nonlinear robust regression problem. *Journal of Statistical Computation and Simulation*, 13(2):79–113. 106

[Eade, 2009] Eade, E. (2009). Gauss-Newton / Levenberg-Marquardt Optimization. Technical report. 107, 108

[Eliazar and Parr, 2004] Eliazar, A. I. and Parr, R. (2004). Dp-slam 2.0. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1314–1320. IEEE. 34

[Eustice et al., 2005] Eustice, R. M., Singh, H., and Leonard, J. J. (2005). Exactly Sparse Delayed State Filters. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 36

[Everingham et al., 2006] Everingham, M., Muller, H., and Thomas, B. (2006). Evaluating image segmentation algorithms using the pareto front. *Computer Vision—ECCV 2002*, pages 255–259. 119

[Fischler and Bolles, 1981] Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395. 35

[Fitzgibbon, 2001] Fitzgibbon, A. W. (2001). Robust Registration of 2D and 3D Point Sets. In *Proceedings of the British Machine Vision Conference (BMVC)*. 38, 103

[Förstner, 1987] Förstner, W. (1987). Reliability analysis of parameter estimation in linear models with applications to mensuration problems in computer vision. *Computer Vision, Graphics, and Image Processing*, 40(3):273–310. 103

[Funke and Pietzsch, 2009] Funke, J. and Pietzsch, T. (2009). A Framework for Evaluating Visual SLAM. In *Proceedings of the British Machine Vision Conference (BMVC)*. 126

[Fusiello et al., 1999] Fusiello, A., Trucco, E., Tommasini, T., and Roberto, V. (1999). Improving feature tracking with robust statistics. *Pattern Analysis and Applications*, 2(4):312–320. 106

[Gallup et al., 2008] Gallup, D., Frahm, J.-M., Mordohai, P., and Pollefeys, M. (2008). Variable baseline/resolution stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 202

[Geiger et al., 2012] Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the KITTI vision benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361. IEEE. 141, 199

[Gennery, 1992] Gennery, D. B. (1992). Visual tracking of known three-dimensional objects. *International Journal of Computer Vision (IJCV)*, 7(3):243–270. 20

[Grimson and Lozano-Perez, 1987] Grimson, W. E. L. and Lozano-Perez, T. (1987). Localizing overlapping parts by searching the interpretation tree. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, (4):469–482. 32

[Grossberg and Nayar, 2003] Grossberg, M. D. and Nayar, S. K. (2003). What is the Space of Camera Response Functions? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 130, 135

[Gruen, 1985] Gruen, A. W. (1985). Adaptive Least Squares Correlation: A Powerful Image Matching Technique. *South African Journal of Photogrammetry, Remote Sensing, and Cartography*, 14. 103

[Gutmann and Konolige, 1999] Gutmann, J.-S. and Konolige, K. (1999). Incremental Mapping of Large Cyclic Environments. In *International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*. 33

[Hager and Belhumeur, 1998] Hager, G. D. and Belhumeur, P. N. (1998). Efficient region tracking with parametric models of geometry and illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:1025–1039. 92, 106

[Hahnel et al., 2003] Hahnel, D., Burgard, W., Fox, D., and Thrun, S. (2003). An efficient fastslam algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, volume 1, pages 206–211. IEEE. 34

[Hampel et al., 1986] Hampel, F. R., Ronchetti, E. M., Rousseeuw, P. J., and Stahel, W. A. (1986). *Robust Statistics: The Approach Based on Influence Functions (Wiley Series in Probability and Statistics)*. Wiley-Interscience, New York, first edition. 102, 106

[Handa et al., 2010] Handa, A., Chli, M., Strasdat, H., and Davison, A. J. (2010). Scalable Active Matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 36

[Hanna, 1991] Hanna, K. (1991). Direct multi-resolution estimation of ego-motion and structure from motion. In *Visual Motion, 1991., Proceedings of the IEEE Workshop on*, pages 156–162. IEEE. 95

[Harris and Stennet, 1990] Harris, C. and Stennet, C. (1990). RAPiD – A video-rate object tracker. In *British Machine Vision Conference*, pages 73–77. 19, 92

[Harris, 1992] Harris, C. G. (1992). Tracking with Rigid Models. In Blake, A. and Yuille, A., editors, *Active Vision*. MIT Press, Cambridge, MA. 19, 92, 97

[Hasinoff, 2012] Hasinoff, S. (2012). Photon, poisson noise. `http://people.csail.mit.edu/hasinoff/pubs/hasinoff-photon-2012-preprint.pdf`. 129

[Hasinoff et al., 2010] Hasinoff, S. W., Durand, F., and Freeman, W. T. (2010). Noise-Optimal Capture for High Dynamic Range Photography. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 169

[Hasinoff et al., 2009] Hasinoff, S. W., Kutulakos, K. N., Durand, F., and Freeman, W. T. (2009). Time-Constrained Photography. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 169

[Heeger and Jepson, 1992] Heeger, D. and Jepson, A. (1992). Subspace methods for recovering rigid motion i: Algorithm and implementation. *International Journal of Computer Vision (IJCV)*, 7(2):95–117. 93

[Heel, 1990] Heel, J. (1990). Direct estimation of structure and motion from multiple frames. Technical report, DTIC Document. 96

[Higgins and Prazdny, 1980] Higgins, L. H. C. and Prazdny, K. (1980). The interpretation of a moving retinal image. *Proceedings of the Royal Society of London. Series B, Biological Sciences (1934-1990)*, 208(1173):385–397. 93

[Hinterstoisser et al., 2007] Hinterstoisser, S., Benhimane, S., and Navab, N. (2007). N3M: Natural 3D Markers for Real-Time Object Detection and Pose Estimation. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 77

[Horn et al., 2007] Horn, B., Fang, Y., and Masaki, I. (2007). Time to contact relative to a planar surface. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, pages 68–74. 96

[Horn and Schunck, 1981] Horn, B. and Schunck, B. (1981). Determining optical flow. *Artificial Intelligence*, 17:185–203. 91

[Horn and Schunck, 1993] Horn, B. K. P. and Schunck, B. G. (1993). Determining optical flow: A retrospective. *Artificial Intelligence*, 59:81–87. 93

[Horn and Weldon, 1988] Horn, B. K. P. and Weldon, E. J. (1988). Direct methods for recovering motion. *International Journal of Computer Vision*, 2(1):51–76. 94

[Huang and Mumford, 1999] Huang, J. and Mumford, D. (1999). Statistics of natural images and models. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 1. IEEE. 103

[Huber, 1981] Huber, P. J. (1981). *Robust Statistics*. Wiley Series in Probability and Statistics. Wiley-Interscience. 106

[Hwangbo et al.,] Hwangbo, M., Kim, J.-S., and Kanade, T. Gyro-aided feature tracking for a moving camera: fusion, auto-calibration and gpu implementation. 176

[Irani and Anandan, 1998] Irani, M. and Anandan, P. (1998). A unified approach to moving object detection in 2d and 3d scenes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(6):577–589. 96

[Irani and Anandan, 1999] Irani, M. and Anandan, P. (1999). All About Direct Methods. In *Proceedings of the International Workshop on Vision Algorithms, in association with ICCV*. 88

[Irani et al., 1996] Irani, M., Anandan, P., Bergen, J., Kumar, R., and Hsu, S. (1996). Efficient Representations of Video Sequences and Their Applications. In *Proceedings of Signal Processing: Image Communication (SPIC)*, pages 327–351. 95

[Irani et al., 1999] Irani, M., Anandan, P., and Cohen, M. (1999). Direct recovery of planar-parallax from multiple frames. In *Proceedings of the International Workshop on Vision Algorithms, in association with ICCV*, pages 1528–1534. 95

[Irani et al., 2000] Irani, M., Anandan, P., and Cohen, M. (2000). Direct recovery of planar-parallax from multiple frames. *Vision Algorithms: Theory and Practice*, pages 9–75. 95

[Irani et al., 1998] Irani, M., Anandan, P., and Weinshall, D. (1998). From reference frames to reference planes: Multi-view parallax geometry and applications. *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 829–845. 95

[Irani et al., 1994] Irani, M., Rousso, B., and Peleg, S. (1994). Recovery of ego-motion using image stabilization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 454–460. IEEE. 95

[Isard and Blake, 1996] Isard, M. and Blake, A. (1996). Contour Tracking by Stochastic Propagation of Conditional Density. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 22

[Ishii et al., 1996] Ishii, I., Nakabo, Y., and Ishikawa, M. (1996). Target tracking algorithm for 1 ms visual feedback system using massively parallel processing. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 3, pages 2309–2314. IEEE. 41

[Ishii et al., 2009] Ishii, I., Taniguchi, T., Sukenobe, R., and Yamamoto, K. (2009). Development of high-speed and real-time vision platform, h¡ sup¿ 3¡/sup¿ vision. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 3671–3678. IEEE. 42

[Ishii et al., 2010] Ishii, I., Taniguchi, T., Yamamoto, K., and Takaki, T. (2010). 1000-fps real-time optical flow detection system. In *IS&T/SPIE Electronic Imaging*. International Society for Optics and Photonics. 42

[Ishikawa et al., 1992] Ishikawa, M., Morita, A., and Takayanagi, N. (1992). High speed vision system using massively parallel processing. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, volume 1, pages 373–377. 40, 42

[Jin et al., 2003] Jin, H., Favaro, P., and Soatto, S. (2003). A semi-direct approach to structure from motion. *The Visual Computer*, 19(6):377–394. 96

[Jurie and Dhome, 2001] Jurie, F. and Dhome, M. (2001). A simple and efficient template matching algorithm. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 200–1. 92

[Kagami, 2010] Kagami, S. (2010). High-speed vision systems and projectors for real-time perception of the world. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, pages 100–107. IEEE. 39, 42

[Kapaldo, 2005] Kapaldo, A. J. (2005). *Gyroscope Calibration and Dead Reckoning for an Autonomous Underwater Vehicle*. PhD thesis, Virginia Polytechnic Institute and State University. 176

[Kemp and Drummond, 2004] Kemp, C. and Drummond, T. (2004). Multi-modal tracking using texture changes. In *Proceedings of the British Machine Vision Conference (BMVC)*. Citeseer. 20

[Kemp and Drummond, 2005] Kemp, C. and Drummond, T. (2005). Dynamic measurement clustering to aid real time tracking. In *Proceedings of the International Conference on Computer Vision (ICCV)*, volume 2, pages 1500–1507. IEEE. 20

[Klein and Drummond, 2003] Klein, G. and Drummond, T. (2003). Robust visual tracking for non-instrumental augmented reality. In *Mixed and Augmented Reality, 2003. Proceedings. The Second IEEE and ACM International Symposium on*, pages 113–122. IEEE. 98

[Klein and Murray, 2007] Klein, G. and Murray, D. W. (2007). Parallel Tracking and Mapping for Small AR Workspaces. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*. 55, 56, 81, 82, 89, 90, 98

[Klein and Murray, 2010] Klein, G. and Murray, D. W. (2010). Simulating Low-Cost Cameras for Augmented Reality Compositing. *IEEE Transactions on Visualization and Computer Graphics (VGC)*, 16(3):369–380. 133

[Konolige, 2004] Konolige, K. (2004). Large-scale map-making. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, pages 457–463. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999. 33

[Kruppa, 1913] Kruppa, E. (1913). *Zur Ermittlung eines Objektes aus zwei Perspektiven mit innerer Orientierung*. Hölder. 26

[Kumar et al., 1994] Kumar, R., Anandan, P., and Hanna, K. (1994). Direct recovery of shape from multiple views: A parallax based approach. In *Pattern Recognition, 1994. Vol. 1-Conference A: Computer Vision & Image Processing., Proceedings of the 12th IAPR International Conference on*, volume 1, pages 685–688. IEEE. 95

[Kumar et al., 1995] Kumar, R., Anandan, P., Irani, M., Bergen, J., and Hanna, K. (1995). Representation of scenes from collections of images. In *Representation of Visual Scenes, 1995.(In Conjuction with ICCV'95), Proceedings IEEE Workshop on*, pages 10–17. 95

[Kumar and Hanson, 1989] Kumar, R. and Hanson, A. (1989). Robust estimation of camera location and orientation from noisy data. Technical report, Amherst, MA, USA. 94

[Lepetit and Fua, 2005] Lepetit, V. and Fua, P. (2005). *Monocular-Based 3D Tracking of Rigid Objects*. Now Pub. 19

[Lin and Chang, 2006] Lin, H. and Chang, C. (2006). Photo-consistent motion blur modeling for realistic image synthesis. *Advances in Image and Video Technology*, pages 1273–1282. 133

[Liu et al., 2006] Liu, C., Freeman, W. T., Szeliski, R., and Kang, S. B. (2006). Noise Estimation from a Single Image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 130, 131, 133

[Longuet-Higgins, 1981] Longuet-Higgins, H. (1981). A Computer Algorithm for Reconstructing a Scene from Two Projections. *Nature*, 293:133–135. 26

[Lovegrove and Davison, 2010] Lovegrove, S. J. and Davison, A. J. (2010). Real-Time Spherical Mosaicing using Whole Image Alignment. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 90, 98

[Lovegrove et al., 2011] Lovegrove, S. J., Davison, A. J., and Ibanez-Guzmán, J. (2011). Accurate Visual Odometry from a Rear Parking Camera. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*. 26

[Lowe, 1987] Lowe, D. (1987). Three-dimensional object recognition from single two-dimensional images. *Artificial intelligence*, 31(3):355–395. 92

[Lowe, 1992] Lowe, D. (1992). Robust model-based motion tracking through the integration of search and estimation. *International Journal of Computer Vision (IJCV)*, 8(2):113–122. 20

[Lowe, 1991] Lowe, D. G. (1991). Fitting parameterized three-dimensional models to images. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 13:441–450. 92, 97

[Lu and Milios, 1997a] Lu, F. and Milios, E. (1997a). Globally Consistent Range Scan Alignment for Environment Mapping. *Autonomous Robots*, 4(4):333–349. 33

[Lu and Milios, 1997b] Lu, F. and Milios, E. (1997b). Robot pose estimation in unknown environments by matching 2d range scans. *Journal of Intelligent and Robotic Systems*, 18(32):249–275. 33

[Lucas and Kanade, 1981] Lucas, B. D. and Kanade, T. (1981). An Iterative Image Registration Technique with an Application to Stereo Vision. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 36, 91, 92

[Lucas and Kanade, 1985] Lucas, B. D. and Kanade, T. (1985). Optical Navigation by the Method of Differences. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 981–984. 94

[Lyu and Farid, 2005] Lyu, S. and Farid, H. (2005). How realistic is photorealistic? *Signal Processing, IEEE Transactions on*, 53(2):845–850. 140

[Maimone et al., 2007] Maimone, M., Cheng, Y., and Matthies, L. (2007). Two years of visual odometry on the mars exploration rovers. *Journal of Field Robotics*, 24(3):169–186. 28

[Malis, 2004] Malis, E. (2004). Improving vision-based control using efficient second-order minimization techniques. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 26

[Martull et al., 2012] Martull, S., Peris, M., and Fukui, K. (2012). Realistic cg stereo image dataset with ground truth disparity maps. *ICPR2012 workshop TrakMark2012*. 141

[Matthies et al., 1989] Matthies, L., Kanade, T., and Szeliski, R. (1989). Kalman filter-based algorithms for estimating depth from image sequences. *International Journal of Computer Vision (IJCV)*, 3(3):209–238. 96

[Mayol et al., 2002] Mayol, W., Tordoff, B., and Murray, D. (2002). Designing a miniature wearable visual robot. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 119

[Meilland and Comport, 2012] Meilland, M. and Comport, A. I. (2012). Simultaneous super-resolution, tracking and mapping. Research Report RR-2012-05, CNRS-I3S/UNS, Sophia-Antipolis, France. 91, 98

[Meilland et al., 2011] Meilland, M., Comport, A. I., and Rives, P. (2011). Dense visual mapping of large scale environments for real-time localisation. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*. 37

[Meyer et al., 1986] Meyer, G., Rushmeier, H., Cohen, M., Greenberg, D., and Torrance, K. (1986). An experimental evaluation of computer graphics imagery. *ACM Transactions on Graphics (TOG)*, 5(1):30–50. 137

[Monacos et al., 2001] Monacos, S. P., Portillo, A. A., Liu, W., Alexander, J. W., and Ortiz, G. G. (2001). A high frame rate ccd camera with region-of-interest capability. In *Aerospace Conference, 2001, IEEE Proceedings.*, volume 3, pages 3–1513. IEEE. 42

[Moravec, 1980a] Moravec, H. P. (1980a). Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover. Technical Report CMU-RI-TR-3, Carnegie Mellon University, Robotics Institute. 94

[Moravec, 1980b] Moravec, H. P. (1980b). *Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover*. PhD thesis, Stanford University. 94

[Moschini and Fusiello, 2009] Moschini, D. and Fusiello, A. (2009). Tracking human motion with multiple cameras using an articulated model. In *Proceedings of the 4th International Conference on Computer Vision/Computer Graphics Collaboration Techniques and Applications (MIRAGE)*, volume 5496 of *Lecture Notes in Computer Science*, pages 1–12, Rocquencourt. 106

[Murray et al., 1994] Murray, R., Li, Z., and Sastry, S. (1994). *A mathematical introduction to robotic manipulation*. CRC. 49

[Nakabo et al., 1996] Nakabo, Y., Ishii, I., and Ishikawa, M. (1996). High speed target tracking for 1 ms visual feedback system. In *Video Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 42

[Nakabo and Ishikawa, 1998] Nakabo, Y. and Ishikawa, M. (1998). Visual impedance using 1 ms visual feedback system. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 3, pages 2333–2338 vol.3. 42

[Namiki et al., 1999] Namiki, A., Nakabo, Y., Ishii, I., and Ishikawa, M. (1999). High speed grasping using visual and force feedback. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 4, pages 3195–3200. IEEE. 42

[Nebot et al., 2003] Nebot, E., Masson, F., Guivant, J., and Durrant-Whyte, H. (2003). Robust simultaneous localization and mapping for very large outdoor environments. In *Experimental Robotics VIII*, pages 200–209. Springer. 34

[Negahdaripour and Horn, 1985a] Negahdaripour, S. and Horn, B. (1985a). Direct passive navigation: Analytical solution for planes. In *In Proceedings of IEEE Conference on Robotics and Automation*, pages 1157–1163. IEEE Computer Society Press. 94

[Negahdaripour and Horn, 1985b] Negahdaripour, S. and Horn, B. K. P. (1985b). Determining 3-d motion of planar objects from image brightness patterns. In *Proceedings of the 9th international joint conference on Artificial intelligence - Volume 2*, IJCAI'85, pages 898–901, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. 94

[Negahdaripour and Horn, 1989] Negahdaripour, S. and Horn, B. K. P. (1989). A direct method for locating the focus of expansion. *Computer Vision, Graphics, and Image Processing*, 46(3):303–326. 94

[Negahdaripour and Lee, 1992] Negahdaripour, S. and Lee, S. (1992). Motion recovery from image sequences using only first order optical flow information. *International Journal of Computer Vision*, 9(3):163–184. 93

[Neira and Tardós, 2001] Neira, J. and Tardós, J. D. (2001). Data Association in Stochastic Mapping using the Joint Compatibility Test. *IEEE Transactions on Robotics and Automation*, 17(6):890–897. 32, 66

[Newcombe et al., 2011a] Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohli, P., Shotton, J., Hodges, S., and Fitzgibbon, A. (2011a). KinectFusion: Real-Time Dense Surface Mapping and Tracking. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*. 28, 38

[Newcombe et al., 2011b] Newcombe, R. A., Lovegrove, S., and Davison, A. J. (2011b). DTAM: Dense Tracking and Mapping in Real-Time. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 37, 90, 92, 98, 136, 137

[Ng and Chang, 2004] Ng, T. and Chang, S. (2004). Classifying photographic and photorealistic computer graphic images using natural image statistics. Technical report, Technical report, Columbia University (October 2004). 140

[Ni et al., 2009] Ni, K., Jin, H., and Dellaert, F. (2009). GroupSAC: Efficient Consensus in the Presence of Groupings. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 78

[Nieto et al., 2006] Nieto, J., Bailey, T., and Nebot, E. (2006). Scan-slam: Combining ekf-slam and scan correlation. In *Field and service robotics*, pages 167–178. Springer. 33

[Nieto et al., 2007] Nieto, J., Bailey, T., and Nebot, E. (2007). Recursive scan-matching slam. *Robotics and Autonomous Systems*, 55(1):39–49. 33

[Nir et al., 2008] Nir, T., Bruckstein, A. M., and Kimmel, R. (2008). Over-parameterized variational optical flow. *International Journal of Computer Vision (IJCV)*, 76(2):205–216. 92

[Nistér, 2003] Nistér, D. (2003). Preemptive RANSAC for Live Structure and Motion Estimation. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 35

[Nistér, 2004] Nistér, D. (2004). An Efficient Solution to the Five-Point Relative Pose Problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(6):756–777. 26

[Nistér et al., 2004] Nistér, D., Naroditsky, O., and Bergen, J. (2004). Visual Odometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 25, 29, 31

[Panin and Knoll, 2008] Panin, G. and Knoll, A. (2008). Mutual information-based 3d object tracking. *International Journal of Computer Vision (IJCV)*, 78(1):107–118. 96

[Park et al., 2009] Park, Y., Lepetit, V., and Woo, W. (2009). ESM-Blur: Handling & rendering blur in 3D tracking and augmentation. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*. 90

[Park et al., 2012] Park, Y., Lepetit, V., and Woo, W. (2012). Handling motion-blur in 3d tracking and rendering for augmented reality. *Visualization and Computer Graphics, IEEE Transactions on*, 18(9):1449–1459. 90

[Peris et al., 2012] Peris, M., Martull, S., Maki, A., Ohkawa, Y., and Fukui, K. (2012). Towards a simulation driven stereo vision system. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 1038–1042. IEEE. 141, 199

[Posch, 2011] Posch, C. (2011). Next generation bio-inspired vision. *ERCIM News*, pages 24–24. 202

[Posch et al., 2011] Posch, C., Matolin, D., and Wohlgenannt, R. (2011). A qvga 143 db dynamic range frame-free pwm image sensor with lossless pixel-level video compression and time-domain cds. *Solid-State Circuits, IEEE Journal of*. 202

[Press et al., 1992] Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T. (1992). *Numerical Recipes in C: The Art of Scientific Computing, Second Edition*. Cambridge University Press, 2 edition. 107

[Prisacariu and Reid, 2012] Prisacariu, V. A. and Reid, I. D. (2012). PWP3D: Real-time segmentation and tracking of 3d objects. *International Journal of Computer Vision (IJCV)*, 98(3):335–354. 22

[Quam, 1984] Quam, L. (1984). Hierarchical warp stereo. In *Image Understanding Workshop*, pages 149–155. 109

[Rademacher et al., 2001] Rademacher, P., Lengyel, J., Cutrell, E., and Whitted, T. (2001). Measuring the perception of visual realism in images. In *In Rendering Techniques 2001*, pages 235–248. 140

[Raguram et al., 2009] Raguram, R., Frahm, J.-M., and Pollefeys, M. (2009). Exploiting Uncertainty in Random Sample Consensus. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 64

[Reid, 1979] Reid, D. (1979). An algorithm for tracking multiple targets. *Automatic Control, IEEE Transactions on*, 24(6):843–854. 34

[Reinhard et al., 2001] Reinhard, E., Shirley, P., and Troscianko, T. (2001). Natural image statistics for computer graphics. *Univ. Utah Tech Report UUCS-01-002 (Mar. 2001)*. 140

[Ren and Reid, 2012] Ren, C. Y. and Reid, I. (2012). A unified energy minimization framework for model fitting in depth. In *ECCV Workshops (2)*, pages 72–82. 38

[Rosten and Drummond, 2006] Rosten, E. and Drummond, T. (2006). Machine learning for high-speed corner detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 81

[Rousseeuw and Leroy, 1987] Rousseeuw, P. J. and Leroy, A. M. (1987). *Robust regression and outlier detection*. John Wiley & Sons, Inc., New York, NY, USA. 106

[Rumpler et al., 2011] Rumpler, M., Irschara, A., and Bischof, H. (2011). Multi-view stereo: Redundancy benefits for 3d reconstruction. In *Proceedings of the 35th Workshop of the Austrian Association for Pattern Recognition, AAPR/OAGM*. 202

[Rushmeier et al., 1995] Rushmeier, H., Ward, G., Piatko, C., Sanders, P., and Rust, B. (1995). Comparing real and synthetic images: Some ideas about metrics. In *Sixth Eurographics Workshop on Rendering*, pages 82–91. Citeseer. 137

[Rusinkiewicz and Levoy, 2001] Rusinkiewicz, S. and Levoy, M. (2001). Efficient Variants of the ICP Algorithm. In *Proceedings of the IEEE International Workshop on 3D Digital Imaging and Modeling (3DIM)*. 29

[Sawhney, 1994] Sawhney, H. (1994). 3d geometry from planar parallax. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 929–934. IEEE. 95

[Sawhney et al., 1995] Sawhney, H., Ayer, S., and Gorkani, M. (1995). Model-based 2d&3d dominant motion estimation for mosaicing and video representation. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 583–590. 96, 105

[Scaramuzza and Fraundorfer, 2011] Scaramuzza, D. and Fraundorfer, F. (2011). Visual odometry [tutorial]. *Robotics & Automation Magazine, IEEE*, 18(4):80–92. 25

[Scaramuzza et al., 2009] Scaramuzza, D., Fraundorfer, F., and Siegwart, R. (2009). Real-time monocular visual odometry for on-road vehicles with 1-point ransac. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4293–4299. IEEE. 35

[Scharstein and Szeliski, 2001] Scharstein, D. and Szeliski, R. (2001). A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. *International Journal of Computer Vision (IJCV)*, 47:7–42. 140

[Senoo et al., 2006] Senoo, T., Namiki, A., and Ishikawa, M. (2006). Ball control in high-speed batting motion using hybrid trajectory generator. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1762–1767. IEEE. 42

[Senoo et al., 2008] Senoo, T., Namiki, A., and Ishikawa, M. (2008). High-speed throwing motion based on kinetic chain approach. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 3206–3211. IEEE. 42

[Shahrokni et al., 2004] Shahrokni, A., Drummond, T., and Fua, P. (2004). Texture boundary detection for real-time tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 566–577. Springer. 20

[Shi and Tomasi, 1994] Shi, J. and Tomasi, C. (1994). Good Features to Track. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 92

[Shiokata et al., 2005] Shiokata, D., Namiki, A., and Ishikawa, M. (2005). Robot dribbling using a high-speed multifingered hand and a high-speed vision system. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 2097–2102. IEEE. 42

[Shum and Szeliski, 1997] Shum, H. and Szeliski, R. (1997). Panoramic image mosaics. *Microsoft Research, MSR-TR-97*, 23. 96, 107

[Shum and Szeliski, 1998] Shum, H.-Y. and Szeliski, R. (1998). Construction and Refinement of Panoramic Mosaics with Global and Local Alignment. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 96

[Steinbrucker et al., 2011] Steinbrucker, F., Sturm, J., and Cremers, D. (2011). Real-Time Visual Odometry from Dense RGB-D Images. In *Workshop on Live Dense Reconstruction from Moving Cameras at ICCV*. 30

[Stewart, 1999] Stewart, C. V. (1999). Robust Parameter Estimation in Computer Vision. *SIAM Reviews*, 41:513–537. 103

[Sturm et al., 2012] Sturm, J., Engelhard, N., Endres, F., Burgard, W., and Cremers, D. (2012). A benchmark for RGB-D SLAM evaluation. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*. 141

[Szeliski and Coughlan, 1997] Szeliski, R. and Coughlan, J. (1997). Spline-based image registration. *International Journal of Computer Vision (IJCV)*, 22(3):199–218. 108

[Szeliski and Kang, 1995] Szeliski, R. and Kang, S. (1995). Direct methods for visual scene reconstruction. In *Representation of Visual Scenes, 1995.(In Conjuction with ICCV'95), Proceedings IEEE Workshop on*, pages 26–33. IEEE. 96

[Szeliski and Shum, 1997] Szeliski, R. and Shum, H. Y. (1997). Creating full view panoramic image mosaics and environment maps. In *Proceedings of SIGGRAPH*. 96

[Taalebinezhaad, 1992a] Taalebinezhaad, M. A. (1992a). Direct recovery of motion and shape in the general case by fixation. *IEEE Trans. Pattern Anal. Mach. Intell*, 14:847–853. 94

[Taalebinezhaad, 1992b] Taalebinezhaad, M. A. (1992b). Towards autonomous motion vision. *Laboratory, Massachusetts Institute of Technology*. 94

[Taylor and Kriegman, 1994] Taylor, C. and Kriegman, D. (1994). Minimization on the lie group so(3) and related manifolds. *Yale University*. 98

[Toldo and Fusiello, 2009] Toldo, R. and Fusiello, A. (2009). Automatic estimation of the inlier threshold in robust multiple structures fitting. In *Proceedings of the 15th International Conference on Image Analysis and Processing (ICIAP)*, volume 5716 of *Lecture Notes in Computer Science*, pages 123–131, Vietri sul Mare, Italy. Springer. 106

[Tomasi and Shi, 1993] Tomasi, C. and Shi, J. (1993). Direction of heading from image deformations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 422–427. IEEE. 94

[Tommasini et al., 1998] Tommasini, T., Fusiello, A., Trucco, E., and Roberto, V. (1998). Making good features track better. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 178–183, Santa Barbara, CA. IEEE Computer Society Press. 106

[Tordoff and Murray, 2005] Tordoff, B. J. and Murray, D. W. (2005). Guided-MLESAC: Faster image transform estimation by using matching priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 27(10):1523–1535. 35, 64

[Torr and Zisserman, 2000] Torr, P. H. S. and Zisserman, A. (2000). MLESAC: a new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding (CVIU)*, 78(1):138–156. 35

[Triggs et al., 1999] Triggs, B., McLauchlan, P., Hartley, R., and Fitzgibbon, A. (1999). Bundle Adjustment — A Modern Synthesis. In *Proceedings of the International Workshop on Vision Algorithms, in association with ICCV*. 30

[Triggs et al., 2000] Triggs, B., Zisserman, A., Szeliski, R., Sawhney, H. S., Peleg, S., Irani, M., Torr, P., Knight, J., Anandan, P., and Malik, J. (2000). Discussion for Direct versus Features Session. In *Vision Algorithms: Theory and Practice*. 88

[Tykkala et al., 2011] Tykkala, T., Audras, C., and Comport, A. I. (2011). Direct Iterative Closest Point for real-time visual odometry. In *ICCV Workshops*. 28, 30

[Valgaerts et al., 2012] Valgaerts, L., Bruhn, A., Mainberger, M., and Weickert, J. (2012). Dense versus sparse approaches for estimating the fundamental matrix. *International Journal of Computer Vision (IJCV)*, 96(2):212–234. 27

[Vedaldi et al., 2005] Vedaldi, A., Jin, H., Favaro, P., and Soatto, S. (2005). KALMANSAC: Robust filtering by consensus. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 64

[Whelan et al., 2013] Whelan, T., Johannsson, H., Kaess, M., Leonard, J. J., and McDonald, J. (2013). Robust real-time visual odometry for dense rgb-d mapping. 30

[Yamakawa et al., 2011] Yamakawa, Y., Namiki, A., and Ishikawa, M. (2011). Motion planning for dynamic folding of a cloth with two high-speed robot hands and two high-speed sliders. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 5486–5491. IEEE. 42

[Yu et al., 1999] Yu, Y., Debevec, P., Malik, J., and Hawkins, T. (1999). Inverse global illumination: Recovering reflectance models of real scenes from photographs. In *International Conference on Computer Graphics and Interactive Techniques: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, volume 1999, pages 215–224. 139

[Zach et al., 2007] Zach, C., Pock, T., and Bischof, H. (2007). A duality based approach for realtime TV-L1 optical flow. In *Proceedings of the DAGM Symposium on Pattern Recognition*. 65

[Zhang, 1997] Zhang, Z. (1997). Parameter Estimation Techniques: A Tutorial with Application to Conic Fitting. *Image and Vision Computing (IVC)*, 15:59–76. 105