Optical Flow based person following behaviour of a robot

Ankur Handa

Robotics Research Center

IIIT-Hyderabad

a thesis submitted in partial fulfillment
of the requirements for the degree of

Bachelor of Technology

in

Electronics and Communication Engineering

February 2008

Acknowledgements

I am very thankful to my advisors, Dr. Jayanthi Sivaswamy and Dr. Madhava Krishna for giving me complete freedom and flexibility to work on this topic. Both of them have been very encouraging and motivating and the intensity of encouragement has always increased with time. I am greatly indebted to them.

I owe a huge debt of gratitude to Mr. Sartaj Singh, Scientist, Center for Artificial Intelligence and Robotics(CAIR) for his invaluable help and guidance in designing the robotic platform which I used to test algorithms. I am also very thankful to all the people who I have talked to, at CAIR. Those small discussions were very wonderful and provided me a really valuable information.

I also owe a big 'thank-you' to my lab mates, Vijay for creating an entertaining and humorous atmosphere around and Mahesh and Rahul for their encouragement.

I would also like to thank my friend Ankul Batra for his immense contribution and help he has provided me. Thank you Jairaj, Rishab and Anshul for helping me test the algorithm. These people (2^{nd} year ECE) stayed there in the lab till very late in the night inspite of having an exam two days later.

Lastly, thanks to Gururaj for intimating me about the initial errata in the report.

Abstract

Tracking features between two consecutive images captures the essence of motion in order to categorize objects (either static or moving) in the scene. There has been a lot of literature on tracking features (sparse or dense) and lot of improvements have also been proposed over time. Many of these methods try to extract motion either through global optic flow methods, Horn-Schunck or local optic flow methods, Lucas-Kanade. The analysis is more on scenes taken from a static camera in which background remains stationary but it becomes more challenging to extract motion from a moving camera as the motion of camera is also inherited into the objects. We examine the problem of tracking and tailing single as well as multiple people from a camera mounted on a mobile robot and present a solution for the same. In the proposed method, an alternative approach to optic flow computation is taken by formulating it in an energy minimization framework. The computed flow field is filtered using a spatial relative velocity based filter to determine the potential moving objects. Color and depth information is then used to finally segment and correctly classify the moving objects. The approach works for different testing environments including change in illumination, presence of many textured static objects and similar background color.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background and Motivation

Tracking of moving people finds many applications including survellience, security, guidance and monitoring. We deal with the problem of tracking and tailing multiple people with a camera mounted on a mobile platform or a robot. While tracking a single target or person from a moving robot has been studied [9,10,11,18,19], there has not been much attention on tracking multiple people from a mobile robot. In [20], a method for moving a robot to keep maximum number of targets in field of view is presented. However this is an exercise more at the higher decision making level of how to allocate robots to areas where there are more targets than at the lower level of sensor data interpretation to detect motion and finding objects of interest. Tracking multiple people with a stationary camera is possible by modeling the background with a mixture of Gaussians [12]. However, for moving backgrounds, such a method is not suitable. In general, motion extraction is more challenging when both the camera and the objects of interest are in motion as it requires separating the ego-motion of the camera from object motion. One solution determines a transformation function for the static background from a pair of images,to compensate for the ego-motion of the camera [17,7]. The moving objects are then taken to be those points which do not obey the transformation. However, when there are multiple moving objects and the static background forms a smaller area in the image, it is difficult to estimate the transformation function.

An alternative approach is based on range information obtained from a laser on a mobile robot [6] where the positions of multiple moving objects are tracked from successive laser scans. Changes in local occupancy grid map are used to detect leg motion of people and a joint probability based data association to track objects in subsequent scans. Cameras on the other hand can provide a denser depth map that can more robustly cluster objects of interest along with other criteria such as color.

With this method, we present a solution for tracking multiple people, from a moving robot. The solution is based on the fact that there is a difference between the motion of points on a moving object and the motion of static objects when viewed from a moving camera. Specifically, the points on the static objects in the scene should have a relative velocity which is different from that of the points on the moving objects. This is due to the fact that the motion of the static objects is inherited from the camera whereas the motion of the moving objects is not. We propose a modified optic flow-based technique for computing this relative motion using which, the moving objects can then be segmented robustly. To compute the optical flow for each feature, we include the information of motion contained in its neighbors. Most approaches for feature tracking consider each feature independently of the other features, thus neglecting important information that is available in determining the motion of a feature. By incorporating the motion information of neighbors, for calculating flow, a smoothness term is added to the formulation to penalize the deviation of the direction of a feature from its expected value. This leads to a smooth flow of motion vectors. Finally the objects of interest are identified as follows: First, based on the flow vectors, we segment regions that have an abrupt change in spatial relative velocities and intensity profile, in a neighborhood. We call this as a *spatial relative velocity* (SRV) filter. Next, the color models and depth information are incorporated into the flow field to accurately extract the moving objects of interest. The information about moving objects is then used to control the robot motion such that it moves towards the direction where there are more number of people.

The current method has been tested with our camera equipped mobile robot, called SPAWN, in environments where one or more number of people are in motion. The tested environments include moderate changes in ambient light,

presence of many stationary objects having similar disparity and color as that of moving object.

1.2 Structure of the report

The organization of this report follows the flow of the pipeline shown in Figure 1.1. The images captured from the camera mounted on the mobile robot are first processed by the motion segmentation phase of the pipeline. The components of this phase, namely the modified optical flow computation and SRV filter are discussed in Chapter 4. Once motion is detected, in the next phase (presented in Section 4.2), color and depth information are used to cluster the motion segments into moving objects (people) and a robot control law is used to move the robot to follow the cluster that has the maximum person count. In section 4.4, we present results of testing the algorithm in different environments. We finally close with a discussion about the performance of the proposed method and some concluding remarks in Section 4.5.

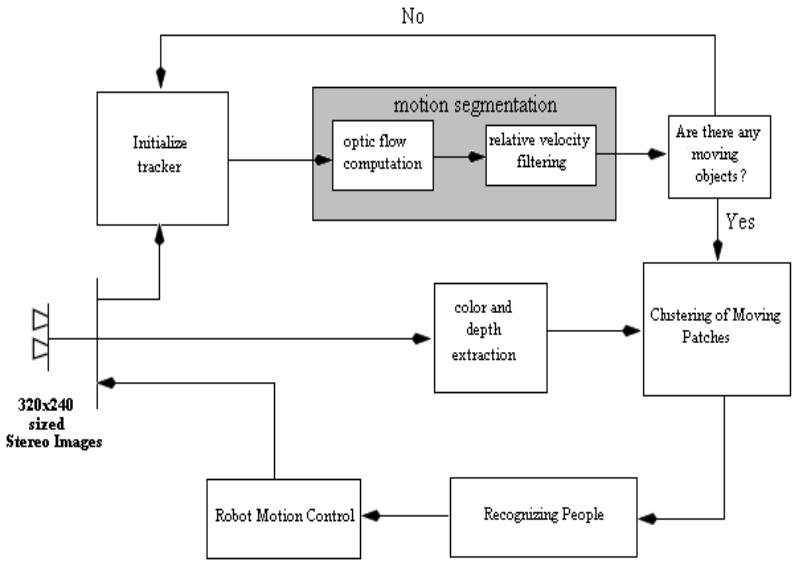


Figure 1.1: Basic Pipeline of the algorithm

Chapter 2

Optic Flow

Optical flow is the distribution of apparent velocities of movement of brighness pattern in an image. The basic assumption for optical flow calculation is that of conservation of pixel intensity. It is assumed that the intensity, or the color of the objects has not changed significantly between two frames. Following this idea, we can derive a constraint equation for an image to compute the flow of motion. Images taken from a camera separated by a small time interval δt may be observed to have changes in point intensity $\delta I(x, y)$ value during that time. The optical flow is a vector field describing the intensity change by indicating the motion of features from one image to another.

In general, the optical flow will not be same as the true the 2-D projection of the 3-D motion field. A typical example can be of a perfectly featureless sphere rotating about its vertical axis. This sphere does not induce any optical flow, but the 2-D projection of its motion field is non-zero everywhere on it except at the occluding boundries. In other words, if the sphere is stationary but a light source moves, the changes in shading will induce an optical flow field even though the motion field is zero everywhere.

There is a lot of literature available on the techniques to calculate the optical flow but these techniques work well under certain constraints. These can be classified into gradient based and correlation based. We will look at both of these techniques in detail in the following sections.

2.1 Gradient Based Optical flow

A common technique used to compute optical flow assumes that the total spatial and temporal derivates of the image brightness remains constant. For small motions, this assumption seem to work pretty well unless the case is severe like occluding boundaries. Assuming that the brightness remains constant over small interval of time δt then we can say that :

$$\frac{\delta I}{\delta t} = 0$$

This essentially means that :

$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t)$$

Assuming the movement to be small enough, the image constraint at $I(x, y, t)$ with Taylor series can be developed to get :

$$I(x + \delta x, y + \delta y, t + \delta t) = I(x, y, t) + \frac{\delta I}{\delta x}\delta x + \frac{\delta I}{\delta y}\delta y + \frac{\delta I}{\delta t}\delta t + ...$$

ignoring the higher order terms we can say that

$$\frac{\delta I}{\delta x}\delta x + \frac{\delta I}{\delta y}\delta y + \frac{\delta I}{\delta t}\delta t = 0$$

The spatial derivates $\frac{\delta I}{\delta x}$ and $\frac{\delta I}{\delta y}$ and the temporal derivate at an image point $\frac{\delta I}{\delta t}$ can be estimated by using two or more images. Thus we have two motion variables $(u = \frac{\delta I}{\delta x}, v = \frac{\delta I}{\delta y})$ And one constraint. Also the motion along the direction of gradients $(\frac{\delta I}{\delta x}, \frac{\delta I}{\delta y})$is available. This is known as *aperture problem* i.e. motion along an edge is ambiguous. To find the optical flow another set of equations is needed, given by some additional constraint. The solution as given by Lucas and Kanade is a non-iterative method which assumes a locally constant flow. Assuming that the flow (u, v) is constant in a small window of size m $\times$ m with

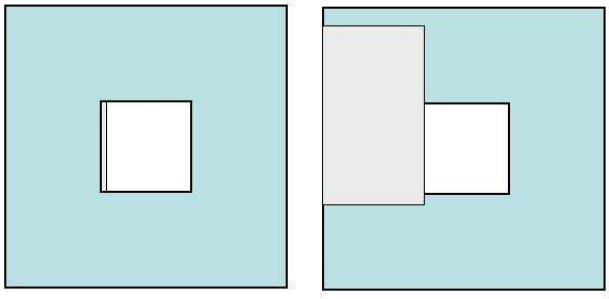


Figure 2.1: The *aperture problem* is illustrated in the image sequence; only the motion *normal* to the translating straight contour can be determined. Left image shows only a part of the translating patch while right image exposes the *aperture problem*. The motion can never be determined, either the translating patch is moving upwards or downwards since the patch is featureless or uniform. Image courtesy: http://robots.stanford.edu/cs223b

m ≥ 2, which is centered at voxel x,y and numbering the pixels within as 1...n, n $= m^2$, a set of equations can be found:

$$\begin{aligned} I_{x1}u + I_{y1}v &= -I_{t1} \\ I_{x2}u + I_{y2}v &= -I_{t2} \\ I_{x3}u + I_{y3}v &= -I_{t3} \\ &\cdot \\ &\cdot \\ I_{xn}u + I_{yn}v &= -I_{tn} \end{aligned}$$

Here I_{xi} denotes i^{th} pixel's gradient along the x direction, I_{yi} denotes pixel's gradient along the y direction and I_{ti} denotes pixel's temporal gradient. If we assume

$$A = \begin{pmatrix} I_{x1} & I_{y1} \\ I_{x2} & I_{y2} \\ . & . \\ . & . \\ I_{xn} & I_{yn} \end{pmatrix}.$$

and

$$b = \begin{pmatrix} I_{t1} \\ I_{t2} \\ . \\ . \\ I_{tn} \end{pmatrix}.$$

then we can write the equation as :

$$A\vec{v} = -b$$

To solve the over-determined system of equations, the least squares method is used in the Lucas Kanade optical flow estimation:

$$\begin{aligned} A^T A\vec{v} &= A^T(-b) \\ \vec{v} &= (A^T A)^{-1} A^T(-b) \end{aligned}$$

here

$$A^T A = \begin{pmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{pmatrix}.$$

and

$$A^T(-b) = \begin{pmatrix} -\sum I_x I_t \\ -\sum I_y I_t \end{pmatrix}.$$

Since this derivation requires the invertiblity of matrix for solution, we may encounter situations in which the whole m × m patch is more or less uniform and has no texture and leading to non-invertibility of $A^T A$. Therefore, we look at the properties of the matrix $A^T A$ and see how it affects the solution and other conditions of solvability of this equation.

$$A^T A = \left(\sum (I_g)(I_g)^T \right).$$

where

$$I_g = \begin{pmatrix} I_x \\ I_y \end{pmatrix}.$$

Gradient away from the edge will have a small magnitude while gradient along the edge all point the same direction. Therefore

$$\begin{aligned}\left(\sum I_g(I_g)^T\right) &\approx kI_g(I_g)^T \\ \left(\sum I_g(I_g)^T\right)I_g &= k||I_g||I_g\end{aligned}$$

Here I_g is an eigenvector with eigenvalue k$||I_g||$. Hence the eigenvectors of A^TA relate to edge direction and magnitude. If we compute the eigen values of this

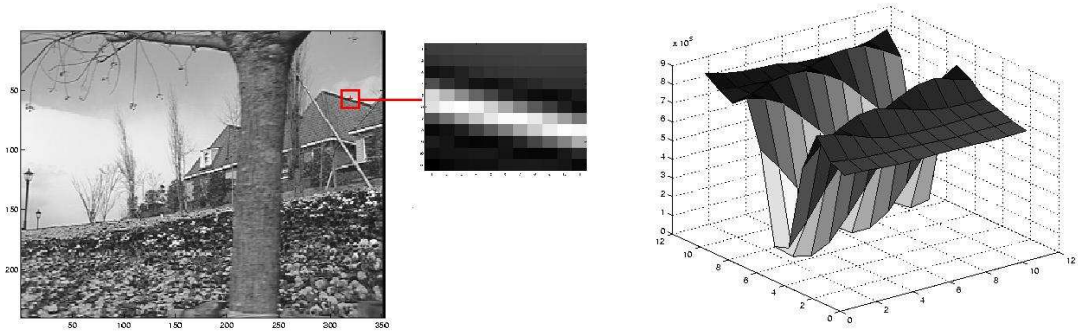


Figure 2.2: The Left image shows the intensity pattern of a block centered around an edge and the right image shows the magnitude of the gradient plot of the intensity in that image block taken.

matrix (A^TA) we may classify the quality of pixel for tracking on the basis of the magnitude of these eigenvalues (say λ_1, λ_2). Since :

$$\begin{aligned}\lambda_1 + \lambda_2 &= trace(A^TA) \\ \lambda_1\lambda_2 &= det(A^TA)\end{aligned}$$

If λ_1 and λ_2 both are very small, it means that the gradient of pixel is very small therefore it may lie in a region of uniform intensity pattern. In case both of them are not very small, the chances are more that the pixel lies in a textured region. If the ratio $\frac{\lambda_1}{\lambda_2}$ is more than a thershold, it confirms that the pixel is good for tracking. Fig. 2.2, Fig. 2.4 and Fig. 2.3 show the quality of a pixel and hence show whether the pixel is good for tracking or not. Now that it is known which pixel is good for tracking, the next thing is to find out the correspondence of

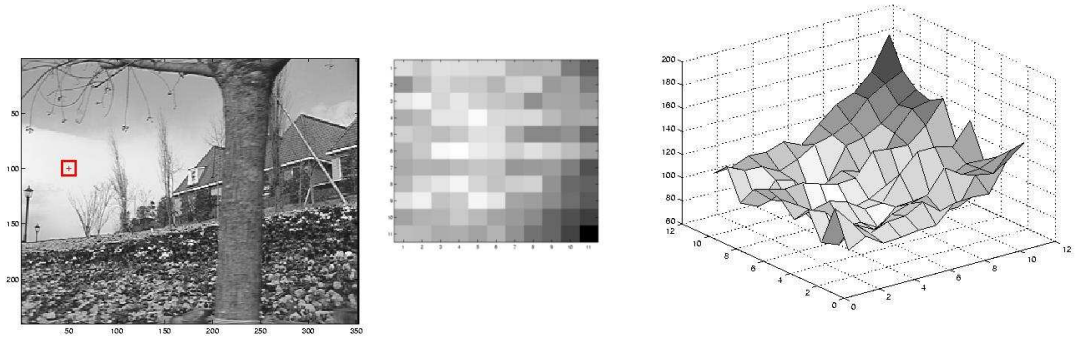


Figure 2.3: The Left image shows a block taken from a non-textured uniform intensity region and right image shows the gradient plot of that block. As it is quite evident from the graph that the gradient values are quite small and hence it does not make for a good pixel to be tracked.

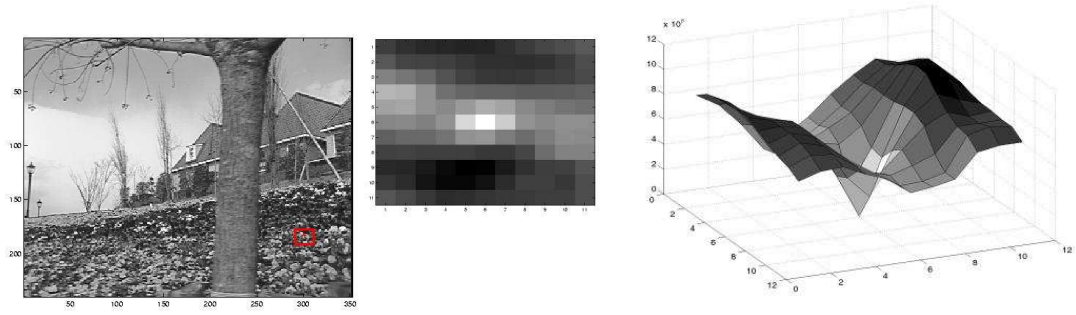


Figure 2.4: The Left image shows the intensity pattern of an image block taken from textured region and the right image shows the gradient plot. The gradient plot completely shows that the magnitude of gradient values is high and hence this is a good pixel to track.

pixel is the next image. The pyramidal LK feature tracker implementation [8] by Birchfield tracks the pixels in a base image to other images in sequence.

2.2 Correlation Based optical flow

In general it is not possible to determine the correct optical flow field given a pair of images due to aperture problem. Under certain constraints, the problem becomes well posed and can be solved significantly. The gradient based optical flow methods generally suffer from noise since they depend upon the gradient of pixel. A relatively noise-resistant method to determine the optical flow would be to find the best possible direction of patch over a given search window in the next image. The maximum possible displacement w is limited to the size of the window. The value of w depends on the expected values of pixel displacement in the image plane. This is shown in the Figure 2.5 Since we consider motion of

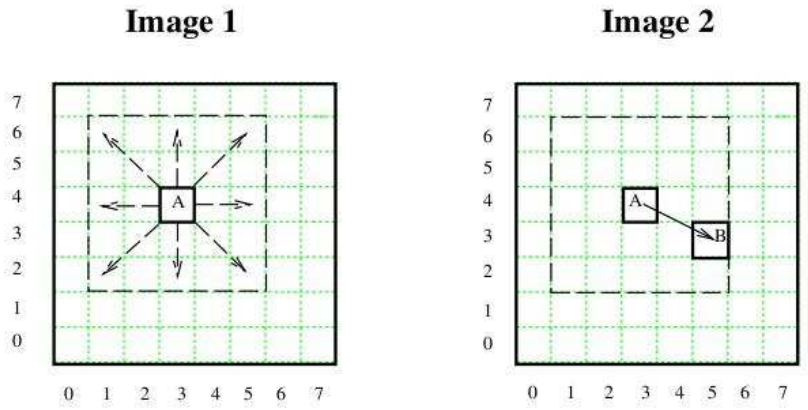


Figure 2.5: Image 1 shows the search space of pixel A for w=2 and Image 2 shows the expected direction $\vec{AB}$ of pixel A to B.

a pixel as the motion of a patch it is assumed that the all the pixels belonging to the patch have similar motion. In other words, the motion vectors of pixel adjacent to the given pixel will be similar. An example is shown in Figure 2.6. The motion of pixels around a given pixel is assumed to be the same due to rigid body assumption and hence a patch centered around that pixel is considered. The patch is moved in a given search window centered around that pixel and for

each location (i, j) in the window, a match strength based on a given criteria function is computed. The final flow direction is the one which minimizes the criteria function. A typical criteria function would be correlation (SSD) of patch

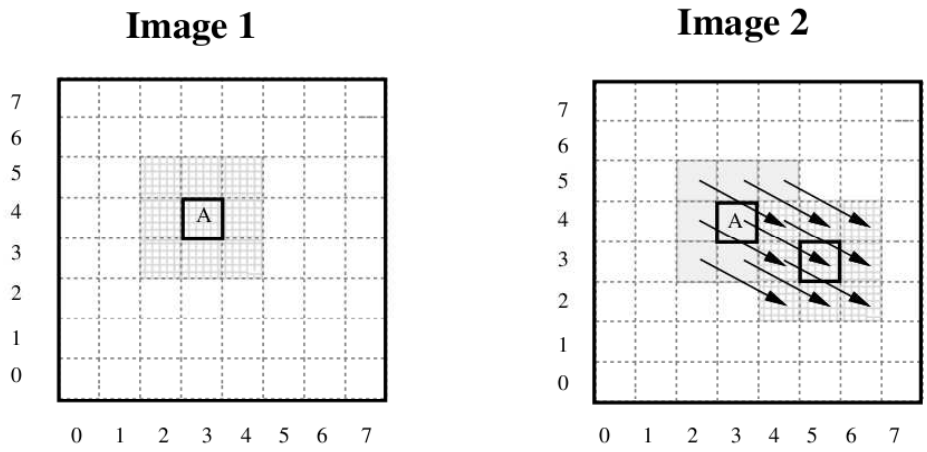


Figure 2.6: Motion vectors of pixels adjacent to a given pixel are assumed to be constant.

in Image 1 and Image 2. We denote the correlation function as I_{corr}.

$$I_{corr}(i,j) = \sum_{y=\frac{-pH}{2}}^{y=\frac{pH}{2}} \sum_{x=\frac{-pW}{2}}^{x=\frac{pW}{2}} (I_1(x,y) - I_2(x+i, y+j))^2$$

Here pW and pH denote the width and the height of the patch and (i, j) denote a location in the search window W of size wHxwW defined as

$$W(i,j) = \begin{cases} 1 & \text{if } \frac{-wW}{2} \leq i \leq \frac{wW}{2} \\ & and \\ & \frac{-wH}{2} \leq j \leq \frac{wH}{2} \\ 0 & otherwise \end{cases}$$

Therefore the direction which minimizes the energy function $E(i, j)$ is defined as:

$$\begin{aligned} E(i,j) &= I_{corr}(i,j)W(i,j) \\ (\hat{i},\hat{j}) &= argmin_{i,j}(E(i,j)) \end{aligned}$$

The best direction is $(\hat{i}, \hat{j})$ which minimizes the energy function.

Selim Temizer *et al.* [25] at MIT have controlled the robot motion to navigate through obstacles by using correlation based optical flow. Their method to obtain the optical flow field is basically as follows: First, a gaussian filter is applied to the raw input images. This is a low pass filter, and has a blurring (smoothing) effect on the image. Then, laplacian filter is applied to obtain the second derivative information from the images. Theoretically, both of these filtering operations are 2-D convolutions, but practically they implement them as two 1-D and one 2-D convolutions. The same effect of applying a 2-D, gaussian filter - an NxN square matrix - is obtained in two 1-D steps (which helps us reduce the number of necessary operations from N^2 to $2N + 1$), and then the laplacian is applied as usual. The combined effect of these two filters are referred to as a LoG filter (Laplacian of Gaussian). The result of this operation is the detection of the edges in the images. After the LoG filter is applied to an image, the zero crossings of the intensity values show the position of the edges. Therefore, it suffices to look at the sign changes to detect the edges. They then produce binary sign of laplacian of gaussian (SLOG) images by using the sign information. Once they have two successive binary SLOG images, in order to find the displacements of features, they apply a procedure called patch matching [23] [24]; For each needle of the flow field, a patch centered around the origin of that needle in the first image is taken. Then this patch is compared with all of the same sized patches that have their centers in a search area in the second image. The search area is a rectangle whose center has the same coordinates with the origin of the needle, and whose sizes can be adjusted on the fly. If the number of the matching pixels in two patches are above some (percentage) threshold, then the two patches are considered to match. The vector defined by the needle origin and the center of the best matching patch is the displacement to be found. They have tested their programs both in simulated environments, and in real environments by using a physical robot.

Hence this approach's "winner-takes-all" nature does not require that the calculated match strengths have any relation whatsoever to what their actual values should theoretically be, it is only necessary that their relative ordering remains same. For example, a change in illumination between frames would

certainly affect the individual match strengths, but need not change the best matching pixel shift. Conversely, any noise in gradient-based method usually directly results in errors in basic optical flow measurements. In the case of change in illumination, the image intensity constraint does not apply since total image intensity does not remain constant.

2.3 Conclusions

The gradient based optical flow tehcniques as said earlier, tend to suffer from noise and hence affect the acual motion computation. The correlation based optical flow techniques have a limit of maximum allowable displacement of the patch, increasing which may affect the performance in real time. Also in correlation based techniques, each patch is allowed to move independently, without even incorporating the motion flow information of the neighboring patches which may increase the accuracy in computing motion.

Chapter 3

Motion extraction from sparse features

3.1 Motion detection

Tracking with sparse features has find a lot of attraction in literature [7,9]. We examine here the problem of extracting motion of external objects from a moving robot. It is relatively difficult to extract the motion since there are two independent motions involved in the scene: the motion of robot and the motion of moving objects in the environment. These motions are blended together when measured through a camera. Once the motion is segregated, moving objects need to be tracked over image sequences. The motion detection process involved is performed in two steps: the ego-motion compensation of camera images, and the position estimation of moving objects in the image space.

3.2 Segregating Camera motion and Object motion

A generic and most intuitive approach for moving object detection assuming camera is static, would be Frame differencing, which compares two consecutive image frames and finds moving objects based on the difference. However, when the camera itself moves (eg. when it is mounted on a mobile robot), simply taking

the difference in frames is not applicable because a big difference is generated by simply moving the camera even if nothing moves in the environment. There are two independent motions involved in the moving camera scenario: motions of moving objects and the ego-motion of the camera. Since these two motions are blended into a single image, the ego-motion of the camera should be eliminated so that the remaining motions, which are due to moving objects, can be detected.

The ego-motion of the camera can be estimated by tracking features between images [1, 2, 3]. When the camera moves, two consecutive images, I_t (the image at time t) and I_{t-1} (the image at time t-1),are in different coordinate systems. Ego-motion compensation is a transformation from the image coordinates of I_{t-1} to that of I_t so that the two images can be compared directly. The transformation can be estimated using two corresponding feature sets: a set of features in I_t and a set of corresponding features in I_{t-1}. However, since there are independently moving objects in the images, a transform model and outlier detection algorithm needs to be designed so that the result of ego-motion compensation is not sensitive to object motions.

The feature selection algorithm introduced in [4] for corresponding feature set selection. The Lucas-Kanade method [5] is applied to track those features in the subsequent image (I_t) to find the corresponding set of features f^t . For efficiency,the search range was limited to a small constant distance (assuming a bounded robot speed). Once the correspondence $< f^{t-1} , f^t >$ is known, the ego-motion of the camera can be estimated using a transformation model and an optimization method. We have studied three different models: affine model, bilinear model, and pseudo-perspective model. When the interval between consecutive images is very small, most ego-motion of the camera can be estimated using an affine model, which can cover translation, rotation, shearing, and scaling motions. However, when the interval is long, the camera motion in the interval cannot be captured by a simple linear model. For example, when the robot moves forward, the features in the image center move slower that those near the image boundary, which is a projection, not a zoom. Therefore, a nonlinear transformation model is required for our case. On the other hand, an over-fitting problem may be caused when a model is highly nonlinear, especially when some

of the selected features are associated with moving objects (outliers). There is clearly a trade-off between a simple, linear model and a highly nonlinear model.

$$\begin{bmatrix} f_x^t \\ f_y^t \end{bmatrix} = \begin{bmatrix} a_0 & a_1 \\ a_2 & a_3 \end{bmatrix} \begin{bmatrix} f_x^{t-1} \\ f_y^{t-1} \end{bmatrix} + \begin{bmatrix} c \\ d \end{bmatrix}$$

Given a transformation model $T_t = [R|t]$, the cost function for least square optimization is defined as :

$$C = \frac{1}{2}\sum_{i=1}^{N}(f_i^t - T_{t-1}^t(f_i^{t-1}))^2$$

where N is the number of features. The model parameters for ego-motion compensation are estimated by minimizing the cost. However, as mentioned before, some of the features are associated with moving objects, which lead to the inference of an inaccurate transformation. Those features (outliers) should be eliminated from the feature set before the final transformation is computed. The model parameter estimation is thus performed using the algorithm described below: It is

Algorithm 1 Algorithm for computing the Transformation

(a) Compute the initial estimate T_0 using the full feature set F.

(b) Divide the feature set into two subsets F_s and F_m as:

$$\begin{cases} f_i^t \in F_s & \text{if } |f_i^t - T_{0_{t-1}}^t(f_i^{t-1})| < \epsilon \\ f_i^t \in F_m & otherwise \end{cases}$$

(c) Re-compute the final estimate T using the subset F_s only.

assumed for outlier detection that the portion of moving objects in the images is relatively small compared to the background; the features which do not agree with the main motion are considered as outliers. This assumption will be violated when the moving objects are very close to the camera or when there are many moving objects in the scene so that they occlude the static background.

Jung and Sukhatme [7] have tested this algorithm on different robotic platforms and results obtained are good. Recently [9] have also proposed their solution to track people with a mobile robot based on the computation of Tranformation function T_t in a different way. They conclude that due to the possible

distraction caused by other moving objects in the scene, along with errors from the tracker and approximation errors in the motion model, the background motion cannot be estimated by simply fitting a model to all the features. Even a robust fitting that discards outliers will not be reliable, because the number of outliers may exceed the number of inliers.

Instead they apply the random sample consensus (RANSAC) algorithm [22] to find small groups of features (containing at least five features) with consistent motion. They repeatedly select five random features from among the background features (determined by disparity), enforcing a minimum distance between the features to ensure that they are well spaced in the image. From these features they compute an initial estimate to the model T_t, which is then applied to all the background features to record the number of inliers. This process is repeated several times, and the motion model with the largest number of inliers is taken to be the background motion. Once the background motion has been estimated, the foreground features that do not match this motion model are discarded.

3.3 Conclusions

This approach works fairly well for a static camera and in other cases when the moving object viewed from a moving camera occupies a small area or remains cosiderably far from the camera, this approach tends to satisfactorily classify the two independent motions: the camera motion and the moving object motion. Since this approach assumes that the most number of inliers are from a static background, the T_t model is more biased towards these static features, but if the background is occluded by various independent moving objects or when the moving object occupies more area in the image, it becomes significantly more difficult to segregate the camera motion and other moving object motion(s).

Chapter 4

The proposed algorithm and Implementation results

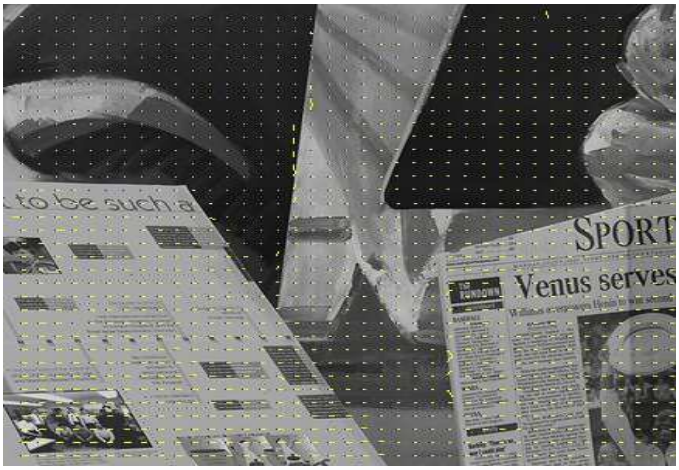
4.1 Motion detection

The task of tailing multiple people from a mobile robot requires good discrimination between the motion of a moving object and that of a static background. Optic-flow techniques have been widely used to extract the motion information [15,16]. However, they are susceptible to noise since they also depend on intensity gradient. In general, the flow vectors tend to drift away from their actual direction if allowed to move independently. Hence, we formulate the flow field determination in an energy minimization framework. The energy function is based on the correlation of an intensity patch in two successive frames and is defined in such a way that it smoothly aligns the flow vectors of textured as well as non-textured static objects. Next, we describe the details of this modified method of flow field computation. Given an image pair I_t and I_{t-1}, we consider a patch of size pWxpH at location (x, y) in image I_t and define an energy function as follows

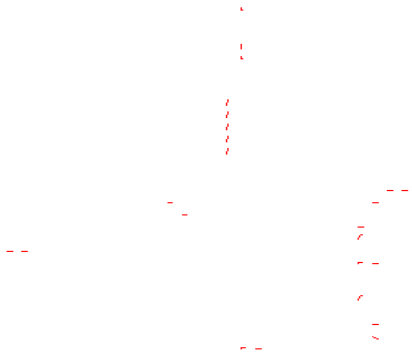
$$E_{corr}(i,j) = \sum_{y=\frac{-pH}{2}}^{y=\frac{pH}{2}} \sum_{x=\frac{-pW}{2}}^{x=\frac{pW}{2}} (I_t(x,y) - I_{t-1}(x+i, y+j))^2 ,$$

(a) Image 1

(b) Image 2



(c) Flow field computation using our enery minimization method



(d) Final Segmentation after SRV filtering

Figure 4.1: The results of energy minimization algorithm on two images. Note the smoothness in the computed flow field on the addition of smoothness term which penalizes any deviation from the expected field. Image courtesy: http://vision.middlebury.edu/flow

This represents the correlation of the intensity patch in I_t with intensity patches in I_{t-1} at locations (i, j) within a window W centered around the (x, y). Next by associating with every patch at (i, j) a direction d_p^{t-1} we define a second energy function as

$$\begin{aligned} E_{dir}(i,j) &= \sum_{k=1}^{m} \alpha_k (d_p^{t-1} - d_{n_k}^t)^2 , \\ d_p^{t-1} &= \tan^{-1}\left(\frac{j}{i}\right) \end{aligned}$$

and $d_{n_k}^t$ is the direction of k^{th} neighbour of the patch at (x, y) in I_t. This function represents a penalty imposed on the flow direction as it takes into account the directions of neighbouring patches (those patches in the neighborhood for which the flow has already been determined). Finally we define a net energy function at each (i, j) as

$$E_{net}(i,j) = E_{corr}(i,j) + E_{dir}(i,j)$$

This is illustrated in Figure 4.2. The final direction d_p^t and net spatial displacement $(\hat{i},\hat{j})$ of a patch is the one which minimizes the net energy function.

$$\begin{aligned} (\hat{i},\hat{j}) &= \arg\min_{i,j}(E_{net}(i,j)W(i,j)) \\ d_p^t &= \tan^{-1}\left(\frac{\hat{j}}{\hat{i}}\right) \end{aligned}$$

Here, the window W of size wHxwW is defined as

$$W(i,j) = \begin{cases} 1 & \text{if } \frac{-wW}{2} \le i \le \frac{wW}{2} \\ & and \\ & \frac{-wH}{2} \le j \le \frac{wH}{2} \\ 0 & otherwise \end{cases}$$

and α_k is a smoothening constant. The images are smoothened with Gaussian filters before computing the energy minimization. Figure 4.1(c) shows the results of our energy minimization technique.

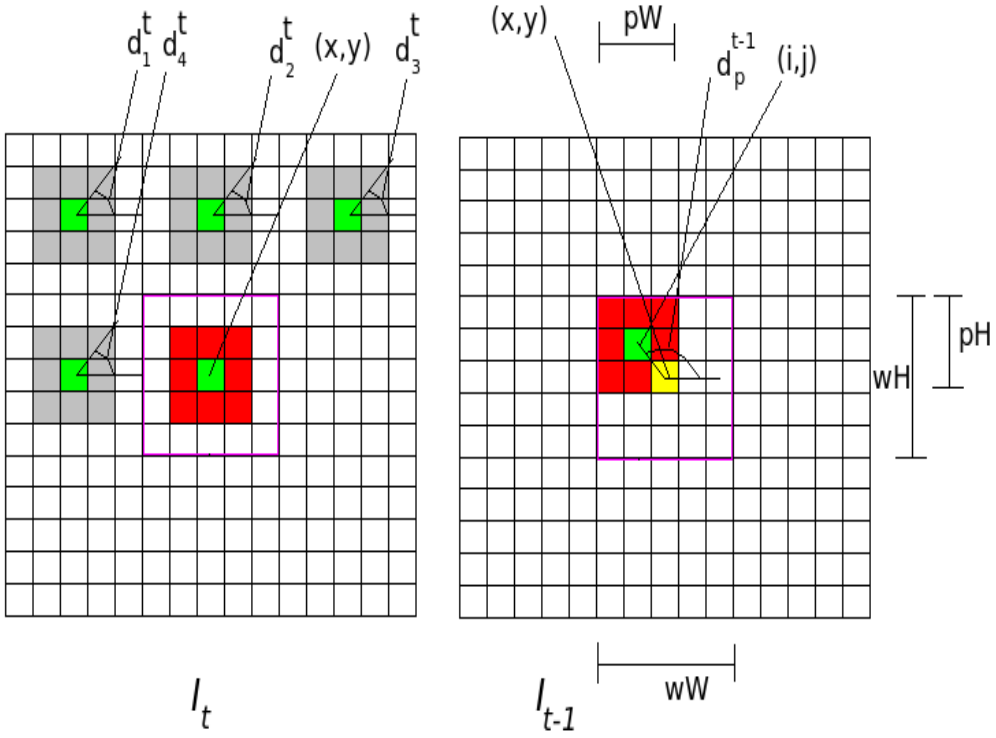


Figure 4.2: Scheme for energy minimization computation. For each location (i, j) in the search window W direction d_p^{t-1} is computed. The super script $t-1$ indicates that the patch is moved in Image I_{t-1} for best match search. This direction is compared absolutely with the directions of neighbors $d_{n_k}^t$ of a given patch in the Image I_t. The super-script t here denotes the directions of patches which have been already calculated. This may correspond to only 4 neighbors of that patch as shown in the figure. Note that this smoothness term tends to accurately classify the motion of each patch. Considering each patch as independent may lead to motion in any direction. Such smoothness functions tend to smoothen the flow field by putting a constraint on their motion flow field.

Now that a flow field has been determined, we can derive the candidate moving objects by examining the distribution of the directions of the flow vectors. Since the background pixels inherit the motion from the camera (which is on a moving platform) their motions will be locally similar. On the other hand, pixels on a moving object will have motions that is dissimilar to the background. Thus the boundary of moving objects should correspond to discontinuity in their relative displacements in a local neighbourhood. Hence, we construct a filter which we call as spatial relative velocity (SRV) filter that detects the moving object boundary as follows. We assign a label for each patch based on the relative velocity distribution. Specifically, if the sum of the relative displacements of a patch with respect to its neighbouring patches, is below a threshold, then it is unlikely to belong to a boundary of a moving object and hence we label that patch as 0. If the sum surpasses the threshold, then it is likely to belong to a moving object. Hence, it is labelled, along with all the neighboring patches, as 1. This is illustrated as follows. Denoting the patch label as $\mathbf{L}$ and sum of displacements as δ:

$$\mathbf{L} = \begin{cases} 0: & \text{if}((\delta_x + \delta_y < th_1) \vee ((\delta_x + \delta_y > th_1) \wedge (\sigma_i < th_2))) \\ 1: & otherwise \end{cases}$$

where δ_x and δ_y are the sum of relative displacements in x and y directions respectively and σ_i is the standard deviation in intensity. Patches labelled 1 are further processed to check for false alarms. This is done based on the variation in the intensity profile. Since the goal is to extract the boundary of moving objects, discontinuity in motion field should correlate with discontinuity in intensity as well. Accordingly, we check for the intensity profile of a patch and if it is smooth, it is unlikely to be part of a moving edge and it is therefore labelled as 0. This filtering tends to accurately classify the motion of every patch, leading to the selection of potential moving patches in the image. Figure 4.1(d) shows the result after SRV filtering.

4.2 Color modeling and person identification

In order to accurately classify each candidate patch as belonging to a moving person, we incorporate color and depth information. After the motion segmenta-

tion, patches are clustered first on the basis of their depth and then classified as belonging to a person or background. We model the color density of the upper body of each person using non-parametric kernel density estimation. Given a sample data for color values $D^c = \{c_i\}$ where $i = 1...N$ and c_i is a k-dimensional vector, kernel density estimation is used to estimate the probability that a given color sample C is from the distribution given by D^c as

$$P(C) = \frac{1}{N}\sum_{i=1}^{N} K(C - c_i)$$

Choosing a zero mean and $\sum$ bandwidth Gaussian function as a kernel estimator function K, we assume independence between the different k channels. Then for each kernel, the bandwidth is

$$\sum = \begin{pmatrix} \sigma_1^2 & 0 & 0 & 0 \\ 0 & \sigma_2^2 & 0 & 0 \\ 0 & . & . & . \\ 0 & . & . & \sigma_k^2 \end{pmatrix}.$$

Hence, the density can be written as

$$P(C) = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{(2\pi)^{\frac{k}{2}}|\sum|^{\frac{1}{2}}} e^{-\frac{1}{2}(C-c_i)^T \sum^{-1}(C-c_i)}$$

A Bayesian classification is used to classify a pixel $\mathbf{p}$ as belonging to a particular person's color model D_i^c.

$$P(D_i^c|\mathbf{p}) = \zeta P(D_i^c)P(\mathbf{p}|D_i^c)$$

wherein the $P(\mathbf{p}|D_i^c)$ is given by the color model of the person and the *prior* probability is obtained from the disparity d and the height h of the pixel in the image and ζ is a normalizing factor. The classification is done using a MAP estimation process:

$$\hat{i} = \max_i(P(D_i^c|\mathbf{p}))$$

Each patch is classified by computing the posterior probability for each pixel in the patch and using a majority rule. The centroid of a person is computed from

the patches belonging to the same class for finally identifying the moving person. Color models are periodically updated by adding new values once identification is done.

In our implementation, the bandwidths were estimated offline from image regions of the upper part of person. The bandwidth for the Gaussian function was estimated as $\sigma \approx 1.06\widehat{\sigma}n^{-1/5}$ where $\widehat{\sigma}$ is the standard deviation and n is the sample size. To speed up the computation of the probabilities, the values of the Gaussian kernel, given the color value difference and kernel function bandwidth, were precalculated and stored in a Look Up Table (LUT). Thus, the values could be fetched in $O(1)$, avoiding excessive floating point computations. Also, the color values for the models were stored as $< r_i^j, g_i^j, b_i^j, n_i^j >$ where $< r_i^j, g_i^j, b_i^j >$ is the sample color data for i^{th} person and n_i^j denotes the number of times the j^{th} color tuple has occured in the sample data. Hence, the likelihood of the pixel to a particular person was computed efficiently as

$$P(\mathbf{p}|D_i^c) = \frac{1}{N}\sum_j n_i^j K_{\sigma_r^i}(r - r_i^j) K_{\sigma_g^i}(g - g_i^j) K_{\sigma_b^i}(b - b_i^j)$$

$$\textit{where}$$

$$\mathbf{p} = < r, g, b > .$$

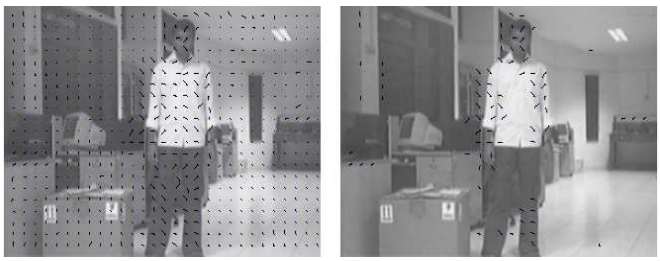
4.3 Robot control

Once the moving persons in the scene are identified, the robot moves in the direction where the density of people is high. The robot velocities are controlled by the disparities and the angles of the centroids of persons, in the image plane. Proximity of the vector (x_c^t, y_c^t, d_c^t), where (x_c^t, y_c^t) are the image coordinates of the centeroid of the person and d_c^t is the disparity, at instant t with its previous location,is used as a consistency check for continuity in motion tracking for every person. All the computed angles are sorted and clustered based on their proximity to each other. The robot's rotation velocity, v_r, is made proportional to average of all the angles in the person cluster having maximum person count and the translation velocity, v_t, is proportional to the average of disparities of persons of

the same cluster.

$$\begin{aligned} v_t &= c_1 d_{mean} \\ v_r &= c_2 \theta_{mean} \end{aligned}$$

and d_{mean} is the mean disparity and θ_{mean} is the mean angle of the persons in the person cluster having maximum person count. In the current implementation, the robot design could only permit an operational rate of 0.2 m/sec and maximum possible was 0.4 m/sec.



(a) Motion Segmentation. Left: the flow field and right: segmented motion.

(b) person identification. Left: depth based clustering and right: the segmented person.

Figure 4.3: Results of the various stages of the algorithm

4.4 Experimental results

The media files of the results obtained are available on this web-site [27]. The proposed method was implemented in C++ on a Linux platform (FC7) with AMD Athlon 64 bit Processor. The image resolution used was 320x240. The entire algorithm was tested comprehensively on our lab robot, SPAWN, in indoor

environments under different conditions. Figure 4.4 shows the trajectories of three among the several expriments performed on the the robot. Paths 1 and 3 correspond to the robot tailing multiple persons while paths 2 and 4 correspond to the robot tracking single person. Path 4 was obtained when the robot tracked a person who moved without facing the camera. Other exprimenrts involved people moving along zig-zag paths in environments cluttered with stationary objects (furniture and persons) and changing lighting conditions which varied from high to low brighness regions. The tracking performance was invariant to how the person faced the robot and similarity of the color of a person's clothing to that of the background. Figure 4.3 shows the various stages of the pipeline described

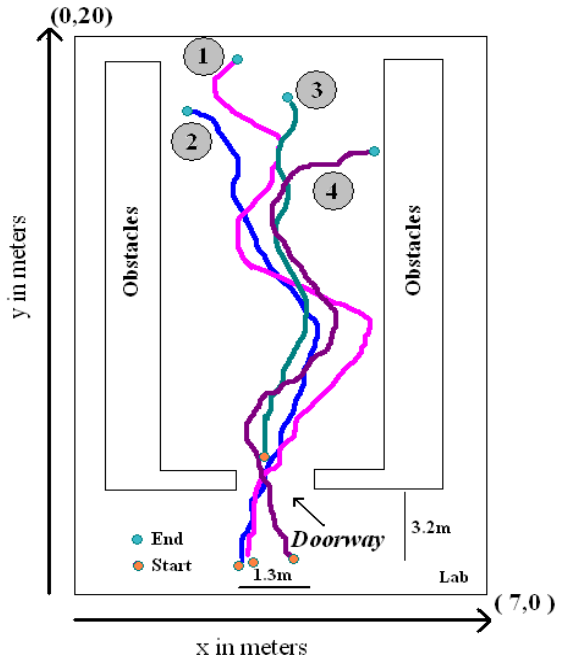
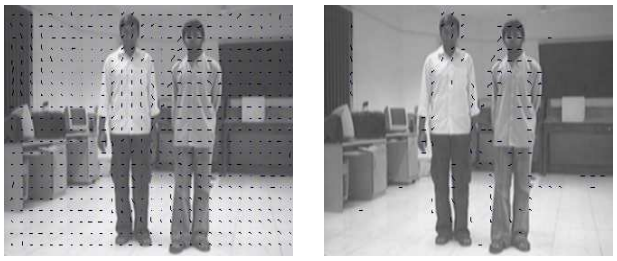


Figure 4.4: Trajectories of the robot as it followed the moving persons. Path 1 and 3 correspond to the robot tailing multiple persons while path 2 and 4 correspond to the robot tracking single person. The robot was made to pass through a narrow doorway and follow a zig-zag path as well.

in Chapter 1. Figure 4.3(a) shows the motion segmentation based on the energy minimization and SRV filtering whereas Figure 4.3(b) shows the clustering of flow vectors based on the depth and the final segmented person. Figure 4.5 shows

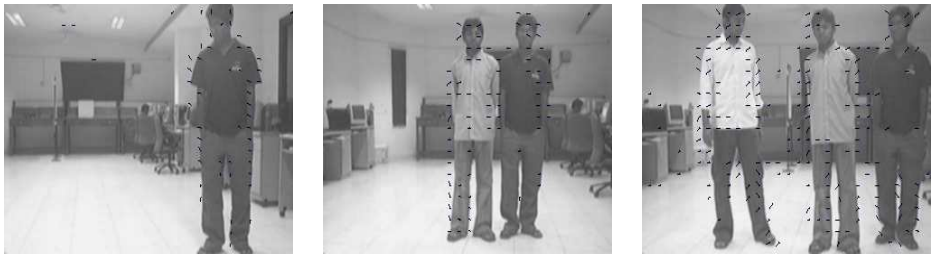
results of motion segmentation on two persons. Figure 4.6 illustrates the tracking



(a) The flow field (b) Segmented moving patches

Figure 4.5: Tracking results for two persons in the view

experiment involving three persons who appeared in the scene at different time intervals. Figure 4.6(a) shows the results when there was only one person in the scene Figure 4.6(b) shows the situation when the second person got introduced in the scene,with the robot beginning to track both and Figure 4.6(c) shows the results when the third person also joined the group.



(a) Single person being tracked (b) Two persons being tracked (c) Three persons being tracked

Figure 4.6: Frames from the experiment where people entered the scene at different times

We also show results of tracking under poor lighting conditions (Figure 4.9), similar background color and depth (Figure 4.7) and when the person is not facing the camera (Figure 4.8). Figure 4.10 shows some results of experiments done on single as well as multiple people.

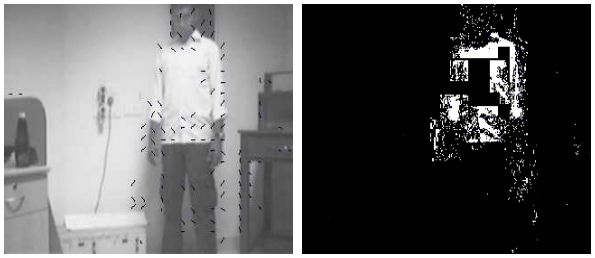
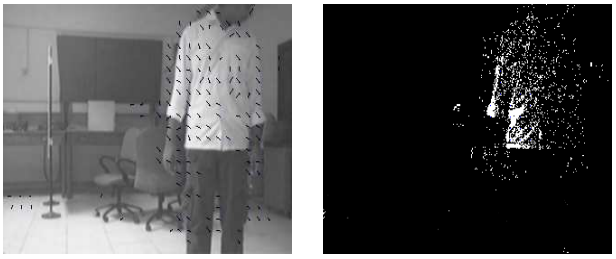


Figure 4.7: Robot is able to track the person when the background color is same. Since we look for patches which have discontinuity in motion and intensity as well, the background patches having similar color may tend to be ruled out due to their uniform intensity profile and continuity in motion as the flow vectors are smoothly aligned after the energy minimization process.

(a) Segmented moving patches (b) Segmented moving person

Figure 4.8: Person being tracked when he is facing away from the camera. The robot is able to track the person even when it is facing away from the camera. This ensures that no facial or skin color based features are being tracked. Also note that there is another person sitting next to the person being tracked and he is wearing a decently textured shirt. Since the motion produced by the person who is sitting, is small, the filtering process tends to discard this from being a potential moving object.



(a) Segmented moving patches (b) Segmented Person

Figure 4.9: Person being tracked under poor lighting conditions. Note the smooth flow vectors. The color model is updated by adding new values from the identified regions of person detection. Since the model is getting updated each step, the robot is able to track the person accurately.

4.5 Discussion and conclusions

We have presented a solution for tailing multiple moving people using a camera on a mobile platform. The method circumvents the need to extract the ego-motion of the camera by devising a novel method for motion segmentation. The segmentation is achieved through an energy-based minimization technique for flow field computation and SRV filtering. The solution also uses statistical color models and depth to improve the accuracy of segmenting the moving objects. The results of various experiments conducted show that, the tailing of multiple people moving in a cluttered environment, was achieved despite challenges imposed during the motion. For instance, the motion included people moving from well-lit to ill-lit zones through a narrow passageway (1.2 m wide). It was also found that the motion segmentation was robust to conditions where the background color was similar to person's clothing, as a result of the adopted motion segmentation technique. In our current implementation, the color models are initialized offline and updated as the motion progresses. Eliminating this offline initialization will increase the scope of the presented method, however it is challenging to devise a fully online process of color modeling, given the camera is also moving.

Finally we note that the problem of tailing multiple people with a robot using only visual information has received little attention in literature despite its potential application in important areas including security and health care

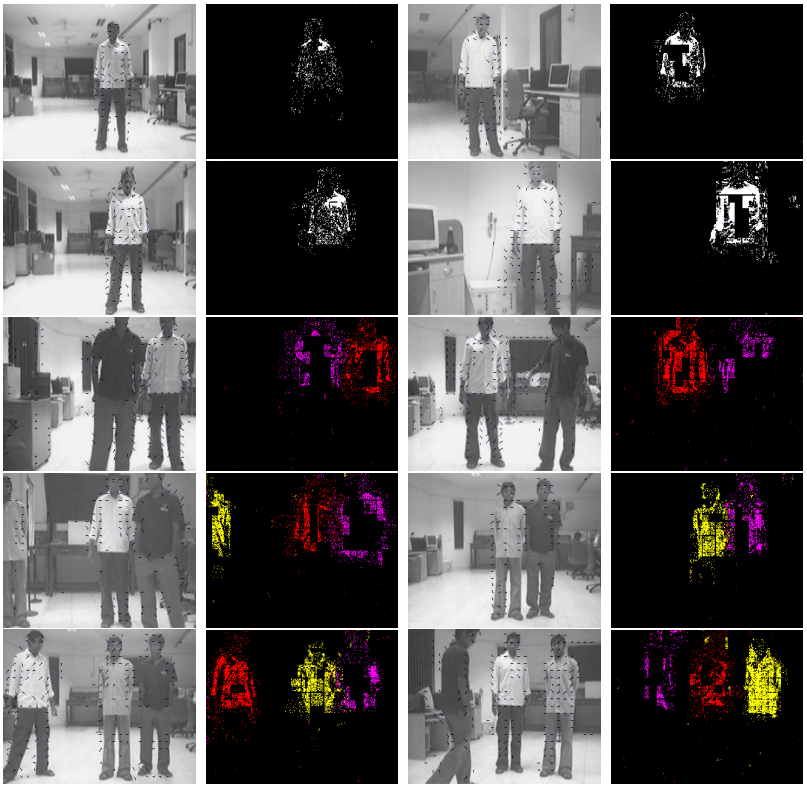


Figure 4.10: Tracking of single as well as multiple people in different frames. First two rows show the results of tracking when there is only single person in the view. Next two rows show the results when there are two persons in the view and last two rows illustrate the tracking performance when there are three people in the field of view.

applications e.g. robotic aid for following group of doctors on ward-rounds in hospitals.

Chapter 5

The Robot

5.1 The Anatomy

The robot, SPAWN [26] used in testing this algorithm was build by the author at the *Center for Artificial Intelligence and Robotics*(CAIR) in summer of 2007. The following sections describe the anatomy of the robot in detail.

1. ATMEL atMega-16.
2. Caster wheels.
3. 12V, 7Ah Lead Acid battery pair.
4. USB cable.
5. MAX-232.
6. Johnson 150 $\times$ 40mm wheels.
7. DC-to-DC converters.
8. PITTMAN motors.
9. Main board.

Figure 5.1: The robot (Spawn) we used to test the algorithm.

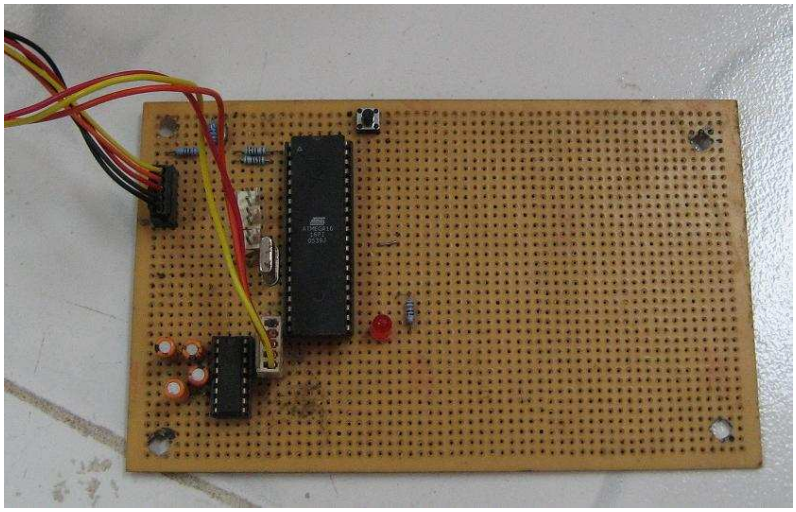


Figure 5.2: This is the avr-atMega16-usb board used for serial to USB data transfer.

5.2 ATMEL atMega-16

The laptops (in this case it is *AMD Turion 64*) today in general don't have serial ports in their motherboard. The ATMEL atMega-16 is used as simple serial to USB converter to exchange the data from the laptop with the robot. A seperate soldered board with an atMega-16, a MAX-232, a red LED and a switch, finds its place on the second acrylic sheet placed in order. A white colored USB wire which is attached to the board has one connector at the other end which fits into the USB port. The red LED flashes as the connection is established with the main board. If this doesn't happen in one pass, pressing the reset switch will ensure another attempt to establish a connection. The main board (the green colored) has a 3 PIN-Male connector for serial data transfer (Tx, Rx and Gnd). The atMega-16 utilizes UART to connect to the main board through a TTL-RS232 level shifter IC, MAX-232 which sits on a 16 PIN IC base soldered on the atMega-16 board. There is another 6 PIN connector comprised of 3 separate 2 PIN connectors placed in series. This connector connects with STK-200 programming dongle to download the hex code from PC into atMega-16. The atMega-16 is running at 12Mhz with external oscillator. At this frequency each bit on the USB takes 8 clock cycles and thus the rate of transfer is 1.5M bits/sec. The circuit is designed

and soldered as explained in [28]. We are using the connection at a low speed data rate.

5.3 Caster wheels

Two caster wheels, one at the front and other at the rear end, are fixed on the acrylic sheet at the bottom. These caster wheels move freely and the positioning is such that they maintain the center of gravity of the robot at the center of the sheet, unlike the case when only one caster wheel is used. Typically such type of wheels are found on shopping carts and rolling chairs.

Figure 5.3: Front caster wheel.

5.4 12V, 7Ah Lead Acid battery pair

Two 12V, 7Ah Lead acid batteries provide the main power supply needed by the robot. These batteries rest on the acrylic sheet at the bottom, above each wheel. The batteries when fully charged can supply 24.5-25.5 Volts. After running the robot for reasonably longer period of time, if the voltage drops to 22V or less, the batteries need charging.

Figure 5.4: A typical 12V, 7Ah Lead Acid battery.

5.5 USB cable

A white colored cable is attached to the atMega-16 board which has USB Series A plug [29] at the other end. This plug is the most common plug and fits into any of the typical USB ports available at the host. When the robot is switched on, this plug is connected into the USB port of the host. The other end of the cable is a 5 PIN female connector fixed into its male counterpart soldered on the atMega-16 board.

Figure 5.5: The figure shows USB cable and plug A at the end.

5.6 MAX-232

The basic purpose of this IC is to shift the voltage levels from TTL to RS232 and vice-versa. Generally this IC is used to connect to the serial port of the PC with any other device having a UART. Since the voltage levels may be different at the device and PC, the IC thus shifts the voltage levels to either logic.

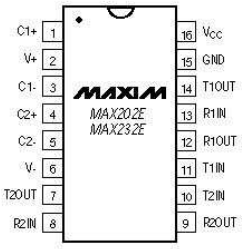


Figure 5.6: Pin-out of MAX232.

5.7 Johnson 150x40mm wheels

Two 150mm diameter and 40mm thick Johnson wheels are used with which the robot runs over. These wheels are quite tough in strength and provide non-shaky and stable movement. Fig. 5.7 shows a snapshot of wheel.

Figure 5.7: Johnson 150x40 wheel used on the robot.

5.8 DC-to-DC converters

Three different DC-to-DC converters derive different isolated voltages from the input voltage supplied by the batteries. A 24V to 24V converter supplies power to the SICK laser, one 24V to 12V converter supplies power to the motors. There are two 24V to 5V converters. One of them supplies power to the motor driver circuit and other provides power to the PIC microcontroller. Fig. 5.8 shows the DC-to-DC converters on board.

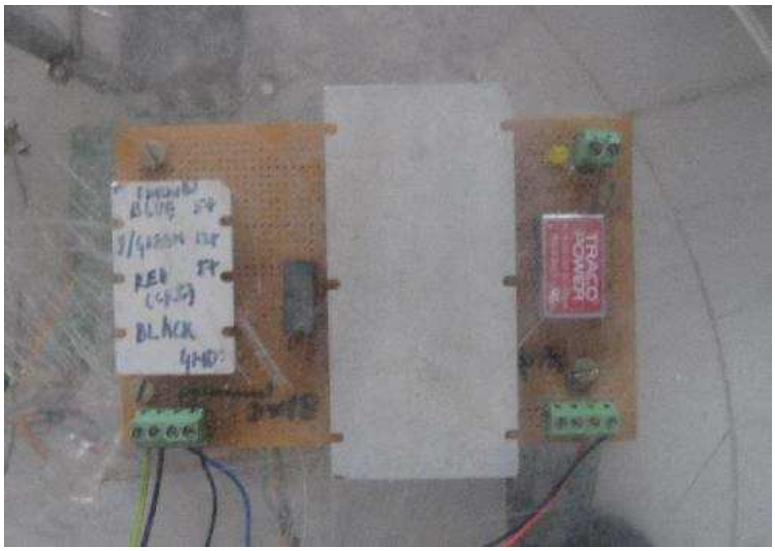


Figure 5.8: DC to DC converters on acrylic sheet.

5.9 PITTMAN motors

Two PITTMAN motors (as shown in Fig. 5.9) with built-in encoders are used to run the wheels. The various specs mechanical and electrical are listed as:

1. DC Brush Gearmotor.
2. 1.37" Diameter.
3. 218.4:1 ratio.
4. 500 oz-in Maximum Continuous Torque.

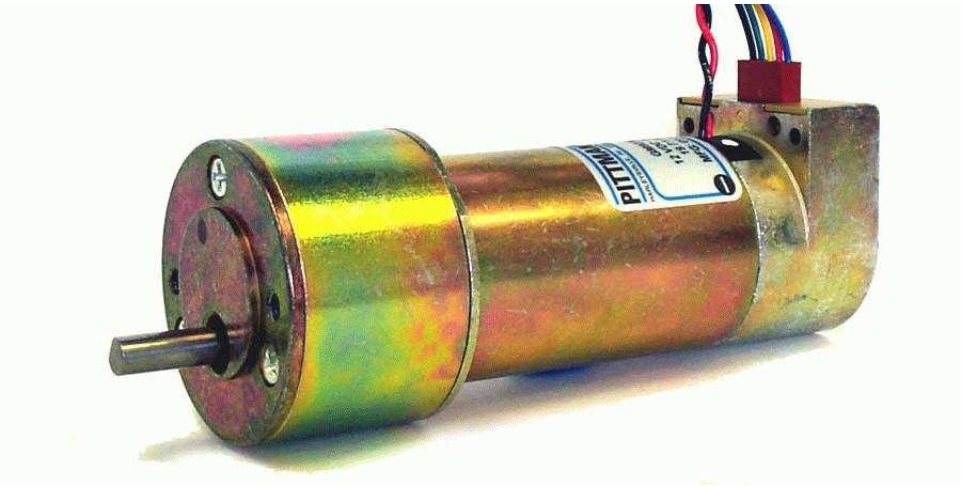


Figure 5.9: The figure shows a sample of PITTMAN GM9234 motor which is used on the robot. These motors belong to the GM9000 series of brushed commutated DC gearmotors.

5. 4199 oz-in Peak Torque (Note: Peak torque is provided for the purpose of performance calculations only. Operation near, or at, a stalled condition will result in motor and/or gearhead damage).
6. 21 rpm No load speed.
7. Torque Constant (Kt) = 3.29 oz-in / amp.
8. Voltage Constant (Ke) = 2.43 v/krpm.
9. Resistance (R) = 1.26 ohms.
10. Inductance = 1.02 mH.
11. Rated voltage: 12 volts.
12. Encoder: 500 CPR.
13. Length (motor) = 3.67".
14. The gearhead will be damaged when operating at the Peak Torque.
15. Unit supplied with ball bearing output shaft.

5.10 Main Board

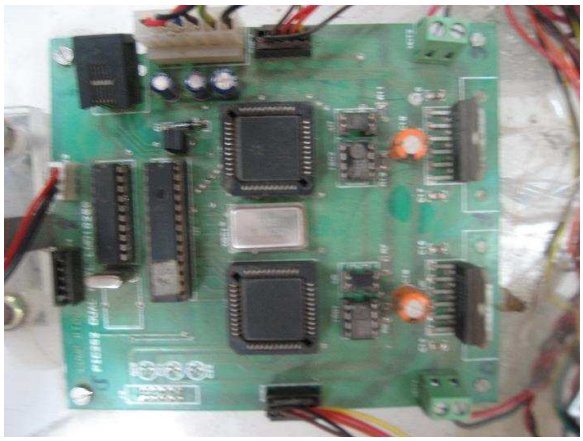


Figure 5.10: The main board (placed at the bottom).

The main board which is lying on the acrylic sheet at the bottom is comprised up of the following ICs:

1. PIC18F452 microcontroller.
2. MAX232 level shifter (TTL to RS232 and vice-versa).
3. LMD18200T motor driver.
4. HCTL-1100 motion controller.

PIC18F452 controls HCTL-1100, LMD18200T and MAX232. Apart from these components, there is one slot for programming PIC18F452, two 5 pin connectors and a 6 pin connector. There are two jumpers just aligned to the placement of micrcontroller on the board. These jumpers have to be taken out when programming is to be done and placed into the same positions after programming. The color coding for the 6 pin connectors is as described below:

Yellowish Green (thick)	+12V volts
Black(thick)	GND(of 12Volts)
Blue	+5V (drive)
Black	GND(of 5Volts drive)
Red	+5V (ckt)
Black	GND(of 5Volts ckt)

5.11 Commands

The various functions to control the motion of the robot included:

5.11.1 robot.connectUsb()

This function is called to connect the host to the robot after the main power switch is turned on, failing which may lead to pop-up message on the terminal **Could not find USB device 'PowerSwitch' with vid=0x16c0 pid=0x5dc**. Once the connection is established, the robot is able to follow the instructions given by the user.

5.11.2 robot.resetOdometery()

This function is called on default just after the attempt to connect to the robot goes successful. It is a sort of good practice to reset the odometery just after the connection is established to avoid any kind of garbage values of odometery affecting the algorithm. Although in our case odometery is not used, but this function is called for the sake of being a good technician.

5.11.3 robot.setVelocity(v_t,v_r)

Here v_t is the translation velocity of the robot and v_r is the rotational velocity of the robot. The robot motion is controlled by the translation as well as rotational velocity. The translational velocity tries to make the robot move forward/backward in a straight line path while the rotational velocity tries to make the robot move along a curve.

Chapter 6

Conclusions

We have demostrated a tracking algorithm with a moving platform. The robot is able to track people under different environment testing conditions. The whole system was set up in the lab and we have restricted the testing to indoors, although the performace will not be greatly affectd in outdoors. The robot design and other mechanical issues did not permit us to test the algorithm over considerably fast moving objects. Owing to this the robot speed was limited to maximum of 1200mm/sec.

The field of view of cameras used (Logitech Notebook Pro) is small, therefore for a given scene only limited number of people can be tracked. Also, we need to build the color models of person before they can be tracked, hence it can track only those people which have been introduced to it in the beginning.

Such type of robotic systems may find application in guiding someone in a big hall, providing physicians with ready access to charts, supplies and patient data and digital assitance for medical personnel in hospital environments. A much advanced application is that of automating time-and-motion studies for increasing the clinical efficency in hospitals.

References

[1] Alberto Censi, Andrea Fusiello, and Vito Roberto. Image stabilization by features tracking. *In Proceedings of the 10th International Conference on Image Analysis and Processing*, pages 665667, Venice, Italy, September 1999.

[2] I. Zoghlami, O. Faugeras, and R. Deriche. Using geometric corners to build a 2d mosaic from a set of images. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 420425, 1997

[3] Gian Luca Foresti and C. Micheloni. A robust feature tracker for active surveillance of outdoor scenes. *Electronic Letters on Computer Vision and Image Analysis*, 1(1):2134, 2003.

[4] Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, Pittsburgh, PA, April 1991.

[5] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. *In Proceedings of the 7th International Joint Conference on Artificial Intelligence*, pages 674697, 1981.

[6] Dirk Schultz, Wolfram Burgard, Dieter Fox, and Armin B. Cremers. Tracking multiple moving targets with a mobile robot using particle filters and statistical data association. In *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, pages 11651170, 2001.

[7] B.Jung, and G.Sukhatme. Detecting Moving Objects using a Single Camera on a Mobile Robot in an Outdoor Environment *In the 8th Conference on Intelligent Autonomous Systems* pp. 980–987, Amsterdam, The Netherlands, March 10-13, 2004.

[8] KLT Tracker Implementation, Stanley Birchfield: http://www.ces.clemson.edu/ stb/klt/

[9] Zhichao Chen and Stanley T. Birchfield, Person Following with a Mobile Robot Using Binocular Feature-Based Tracking *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* San Diego, California, October 2007

[10] M. Piaggio, P. Fornaro, A. Piombo, L. Sanna and R. Zaccaria. An optical flow based person following behaviour.*In Proceedings of the IEEE ISIC/CIRNISAS Joint Conference*, 1998.

[11] C. Schlegel, J. Illmann, H. Jaberg, M. Schuster and R. Worz. Vision based person tracking with a mobile robot. *In The British Machine Vision Conference*, 1998.

[12] Z.Zivkovic. Improved adaptive Gausian mixture model for background subtraction. *International Conference Pattern Recognition*, Vol.2, pages: 28-31 , 2004.

[13] Wren C., A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real Time Tracking of the Human Body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.19, pages:780-785, 1997

[14] Y. Raja, S. McKenna, S. Gong. Object Tracking Using Adaptive Colour Mixture Models,*Proc. ACCV 98*, Vol. I, pp 615-622 66.

[15] B.K. Horn and B.G. Schunck. Determining optical flow. Artificial Intelligence,Vol.17,pages:185-203,1981.

[16] B. Lucas and T. Kanade. An iterative image registeration technique with an application to stereo vision.*In International Joint Conference on Artificial Intelligence (IJCAI)*, pages: 674-679, 1981.

[17] A. Behrad, A. Shahrokni, S. A. Motamedi and K. Madani.A Robust Vision-based Moving Target Detection and Tracking System. *In Pro- ceedings of Image and Vision Computing conference (IVCNZ2001)*,University of Otago, Dunedin, New Zealand, November 26-28, 2001

[18] H. Kwon, Y. Yoon, J. B. Park and A. C. Kak. Person tracking with a mobile robot using two uncalibrated independently moving cameras. *In Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2005

[19] H. Sidenbladh, D. Kragik and H. I. Christensen. A Person following behaviour of mobile robot.*In Proceedings of the IEEE International Conference on Robotics and Automation*, 1999.

[20] B. Jung and Gaurav S. Sukhatme. A Region-based Approach for Cooperative Multi-Target Tracking in a Structured Environment. *In Proceedings of IEEE International Conference on Robotics and Systems* ,2002.

[21] D. Scott. *Multivariate Density Estimation*, 1992.

[22] M. A. Fischler, R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. of the ACM*, 24: 381-395, 1981.

[23] Nishihara, H. K., *Practical Real-Time Imaging Stereo Matcher*, OptEng(23), No. 5, September/October 1984, pp. 536-545

[24] Nishihara, H. K., *Real-Time Implementation of a Sign-Correlation Algorithm for Image-Matching*, Technical report, Teleos Research, February 1990

[25] Selim Temizer, *Optical Flow Based Local Navigation*, MIT

[26] Ankur Handa, A report on SPAWN, IIIT-Hyderbad.

[27] Ankur Handa, Media Files, http://students.iiit.ac.in/~ankurhanda/robot.html/ Note: Type in the browser the same link instead of cut-copy-paste or clicking from here.

[28] http://www.obdev.at/products/avrusb/index.html

[29] http://en.wikipedia.org/wiki/USB