Scalable Active Matching

Ankur Handa, Margarita Chli, Hauke Strasdat and Andrew J. Davison

Department of Computing, Imperial College London, UK

{ahanda, mchli, strasdat, ajd}@doc.ic.ac.uk

Abstract

In matching tasks in computer vision, and particularly in real-time tracking from video, there are generally strong priors available on absolute and relative correspondence locations thanks to motion and scene models. While these priors are often partially used post-hoc to resolve matching consensus in algorithms like RANSAC, it was recently shown that fully integrating them in an 'Active Matching' (AM) approach permits efficient guided image processing with rigorous decisions guided by Information Theory.

AM's weakness was that the overhead induced by intermediate Bayesian updates required meant poor scaling to cases where many correspondences were sought. In this paper we show that relaxation of the rigid probabilistic model of AM, 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 two variations, CLAM and SubAM, in the context of sequential camera tracking. These algorithms are 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.

1. Introduction

Matching is the problem of obtaining correspondence between images, or between a single image and a model constructed from previously processed data. It is a task at the core of almost all computer vision systems which process image sequences, including those which tackle the camera tracking problem we focus on in this paper.

Ultimate performance in matching is represented by the determination of a fully dense correspondence field between images, or at least between the parts of them which observe common parts of the scene. It is feasible to aim to obtain such dense correspondence information in cases where it
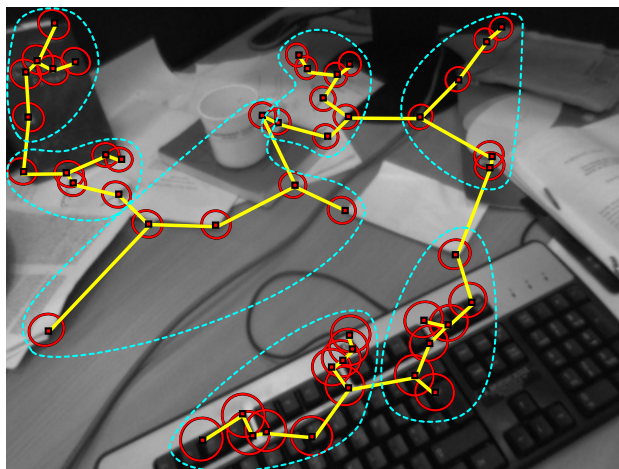

Figure 1. Approximating the joint prior distribution over feature predictions in matching using graphs. In our algorithms we simplify a fully connected graph to a unit-width tree (CLAM), or a tree of subsets (SubAM) to achieve real-time matching of many features.

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 [19]. 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 consists 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 Chli and Davison's Active Matching (AM) algorithm [2, 3] was demonstrated, as the matcher in a filtering-based monocular Simultaneous Localisation and Mapping (SLAM) system [8]. 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. It uses a mixture of Gaussians representation to represent and refine multiple hypotheses, and can also take 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) [12].

AM is a very different paradigm from the matching methods which dominate computer vision currently, where any use of priors is generally non-probabilistic, relying on fixed thresholds to check on matching consensus, and post-hoc as matching candidates gathered using blanket image processing are later refined. RANSAC [9] and more efficient variants [6, 14] are the most widely used algorithms of this type, relying on random sampling and voting to hypothesise and test matching combinations.

Attempts have been made to retro-fit RANSAC algorithms with probabilistic tests and updates in the loop, leading to semi-probabilistic variants like KALMANSAC [18], Guided-MLESAC [17] and Civera et al.'s recent 1-point RANSAC method [7]. Raguram et al.[15] 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 and we do not see any reason not to aim for deterministic, probabilistic algorithms which do not
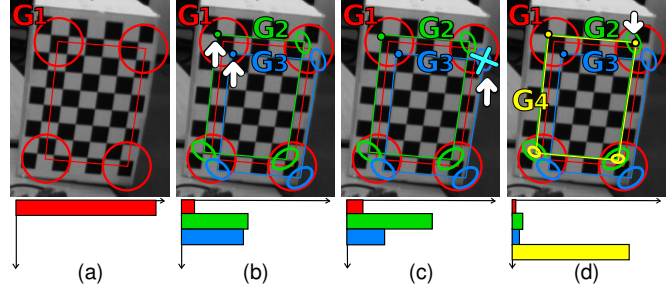

Figure 2. A mini example of AM [2]. The initial search-state $\mathbf{G}_1$ in (a) describes the expected configuration of matches. In (b) a search is made for the top-left of the four predicted features, and the two candidate matches found cause the spawning of new Gaussian hypotheses $\mathbf{G}_2$ 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.

need any random sampling or fixed thresholds.

While AM is therefore technically appealing, detailed performance analysis presented in [3] has revealed its poor scalability with the number of features 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 on speed of processing while sacrificing probabilistic detail.

The aim of this paper is to take a deeper look at the structure of the probabilistic priors fed to matching algorithms, and using graph-theoretic principles to make conservative approximations and sparsifications to the joint measurement density used by AM such that scalability is much improved while matching accuracy is maintained. We propose new algorithms, CLAM and SubAM, based on a tree structure (CLAM) and a tree of feature subsets (SubAM). We demonstrate the performance of our new algorithms in extensive tests in the context of sequential camera tracking using a keyframe-optimisation based SLAM system, and show that SubAM in particular permits highly scalable AM-style matching performance such that these methods are now highly competitive with the best other techniques.

2. The Active Matching Paradigm

Given a new image, AM [2] sets its initial matching search-state to 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:

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

The algorithm follows a predict-measure-update loop which terminates when all features have been searched for

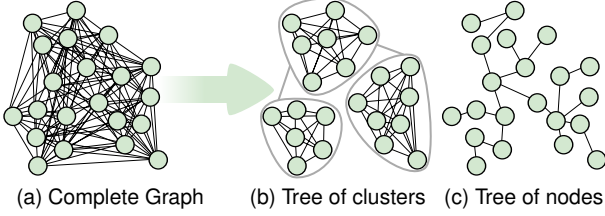(a) Complete Graph   (b) Tree of clusters   (c) Tree of nodes

Figure 3. 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.

and the mixture converges to a probabilistically dominant Gaussian. Below we describe each of these stages briefly which can be visualised in the example of Figure 2:

• **Predict:** Evaluate the expected utility (in terms of mutual information — see Section 3.2) 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.

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 we attack the scalability of matching studying sparsifications of the joint input prior as illustrated in Figure 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.

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 Section 4.2). Note that in the language of Kalman filter tracking, $\mathtt{S}$ is the 'innovation covariance' and is explicitly available at every frame.

3.2. Mutual Information of Candidates

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.

The **Pairwise MI** score quantifies the information a potential measurement for $\mathbf{z}_b$ predicted to provide to prediction $\mathbf{z}_a$ as:

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

where $\mathtt{S}_{a,b}$ is the joint covariance of both $\mathbf{z}_a$, $\mathbf{z}_b$. As explained in [4] 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 is different as there we consider the information shared between $\mathbf{z}_b$ and the rest of the candidates in $\mathbf{z}$ (*i.e.* $I(\mathbf{z}_b; \mathbf{z}_a, \mathbf{z}_c, \dots)$).

4. CLAM: Chow Liu Active Matching

As a first attempt to mitigate the computational overhead of AM, we considered thinning the complete graph of the joint prediction $p(\mathbf{z})$ into a singly-connected tree as in Figure 3(c). While this can indeed be a big approximation, careful selection of the edges preserved can be very beneficial to the closeness of approximation.

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 a tree-shaped model via factorisation:

---

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

$$p(\mathbf{z}) = p(\mathbf{z}_n) \prod_{i=1}^{n-1} p(\mathbf{z}_i | \mathbf{z}_{i+1} \ldots \mathbf{z}_n) \approx p(\mathbf{z}_n) \prod_{i=1}^{n-1} p(\mathbf{z}_i | \mathbf{z}_{i+1}). \quad (3)$$

Out of all such tree factorisations, Chow and Liu [5] 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 3.2).

Inspired by the power of the Chow-Liu (CL) tree to capture the most representative correlation structure in the scene in [4], 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.

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 [1] discussed how a full update requires two passes of the tree (from the leaves to the root and back) so that every node receives updates from all its neighbours. In our case, observing one node at a time permits updates to rest of the nodes by propagating messages all the way to the leaves in a single pass.

$$\mathbf{z}_a \text{ —— } \mathbf{z}_b \text{ —— } \mathbf{z}_c$$

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 $\mathbf{z}_c$.

4.2.1 Propagating Updates

Considering that the joint distribution $p(\mathbf{z})$ of this tree described by a Gaussian $\mathbf{G} = \{\hat{\mathbf{z}}, \mathbf{S}\}$ can be partitioned as follows:

$$\hat{\mathbf{z}} = \begin{pmatrix} \hat{\mathbf{z}}_a \\ \hat{\mathbf{z}}_b \\ \hat{\mathbf{z}}_c \end{pmatrix}, \ \mathbf{S} = \begin{bmatrix} \mathbf{S}_{aa} & \mathbf{S}_{ab} & \mathbf{S}_{ac} \\ \mathbf{S}_{ba} & \mathbf{S}_{bb} & \mathbf{S}_{bc} \\ \mathbf{S}_{ca} & \mathbf{S}_{cb} & \mathbf{S}_{cc} \end{bmatrix}. \quad (4)$$

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

$$\begin{bmatrix} \mathbf{S}_{bb|a} & \mathbf{S}_{bc|a} \\ \mathbf{S}_{cb|a} & \mathbf{S}_{cc|a} \end{bmatrix} = \begin{bmatrix} \mathbf{S}_{bb} & \mathbf{S}_{bc} \\ \mathbf{S}_{cb} & \mathbf{S}_{cc} \end{bmatrix} - \begin{bmatrix} \mathbf{S}_{ba} \\ \mathbf{S}_{ca} \end{bmatrix} \mathbf{S}_{aa}^{-1} \begin{bmatrix} \mathbf{S}_{ab} & \mathbf{S}_{ac} \end{bmatrix}, \quad (5)$$

while similar update-rules apply for the means vector:

$$\begin{pmatrix} \hat{\mathbf{z}}_{b|a} \\ \hat{\mathbf{z}}_{c|a} \end{pmatrix} = \begin{pmatrix} \hat{\mathbf{z}}_b \\ \hat{\mathbf{z}}_c \end{pmatrix} - \begin{bmatrix} \mathbf{S}_{ba} \\ \mathbf{S}_{ca} \end{bmatrix} \mathbf{S}_{aa}^{-1} (\hat{\mathbf{z}}_a - \mathbf{a}). \quad (6)$$

---

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

However, the block $\mathbf{S}_{ca}$[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 $\mathbf{S}_{ca|b} = \mathbf{0}$ (since $\mathbf{z}_{a|b}$ and $\mathbf{z}_{c|b}$ become independent), one can arrive to the expression $\mathbf{S}_{ca} = \mathbf{S}_{cb} \mathbf{S}_{bb}^{-1} \mathbf{S}_{ba}$. Substituting for $\mathbf{S}_{ca}$ back to (5) and (6) it becomes evident that:

$$\mathbf{S}_{cc|a} = \mathbf{S}_{cc} - \mathbf{S}_{cb} \mathbf{S}_{bb}^{-1} \left(\mathbf{S}_{bb} - \mathbf{S}_{bb|a} \right) \mathbf{S}_{bb}^{-1} \mathbf{S}_{bc} \quad (7)$$

$$\mathbf{S}_{bc|a} = \mathbf{S}_{cc} - \mathbf{S}_{cb} \mathbf{S}_{bb}^{-1} \left(\mathbf{S}_{bb} - \mathbf{S}_{bb|a} \right) \quad (8)$$

$$\hat{\mathbf{z}}_{c|a} = \hat{\mathbf{z}}_c - \mathbf{S}_{cb} \mathbf{S}_{bb}^{-1} \left(\hat{\mathbf{z}}_b - \hat{\mathbf{z}}_{b|a} \right). \quad (9)$$

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 $\{\hat{\mathbf{z}}_{b|a}, \mathbf{S}_{bb|a}\}$ of $p(\mathbf{z}_b | \mathbf{z}_a)$. Hence, $\mathbf{S}$ needs to contain explicit entries only for nodes sharing a direct link in the CL-tree. Interestingly, upon successful measurement of a feature $\mathbf{z}_b$, the Gaussian spawned to represent the hypothesis that the match obtained is a true-positive will have zero $\mathbf{S}$-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.

4.2.2 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:

$$I(\mathbf{z}_b; \mathbf{z}_a, \mathbf{z}_c) = \int_{\mathbf{z}} p(\mathbf{z}_a, \mathbf{z}_b, \mathbf{z}_c) \log_2 \frac{p(\mathbf{z}_b, \mathbf{z}_c | \mathbf{z}_a)}{p(\mathbf{z}_b, \mathbf{z}_c)} d\mathbf{z}. \quad (10)$$

Applying Bayes' rule to the ratio inside the logarithm:

$$\frac{p(\mathbf{z}_a, \mathbf{z}_b, \mathbf{z}_c)}{p(\mathbf{z}_a) p(\mathbf{z}_b, \mathbf{z}_c)} = \frac{p(\mathbf{z}_a) p(\mathbf{z}_b | \mathbf{z}_a) p(\mathbf{z}_c | \mathbf{z}_b)}{p(\mathbf{z}_a) p(\mathbf{z}_b) p(\mathbf{z}_c | \mathbf{z}_b)} = \frac{p(\mathbf{z}_b | \mathbf{z}_a)}{p(\mathbf{z}_b)}. \quad (11)$$

Substituting (11) back in (10), it is straightforward to show that $I(\mathbf{z}_b; \mathbf{z}_a, \mathbf{z}_c) = I(\mathbf{z}_b; \mathbf{z}_a)$. 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 is equal to the MI it shares with its immediate neighbours alone. As a result, the costly overhead of updating a dense graph in AM is 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.

4.3. CLAM: A Step-By-Step Example

Figure 4 illustrates a step-by-step example of CLAM within a given frame. Given the joint prior $p(\mathbf{z})$, the dense

---

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

(a) Initial state: the CL tree (b) Propagate 1st measurement

(c) Failed search for a match (d) Measure a hub-like feature

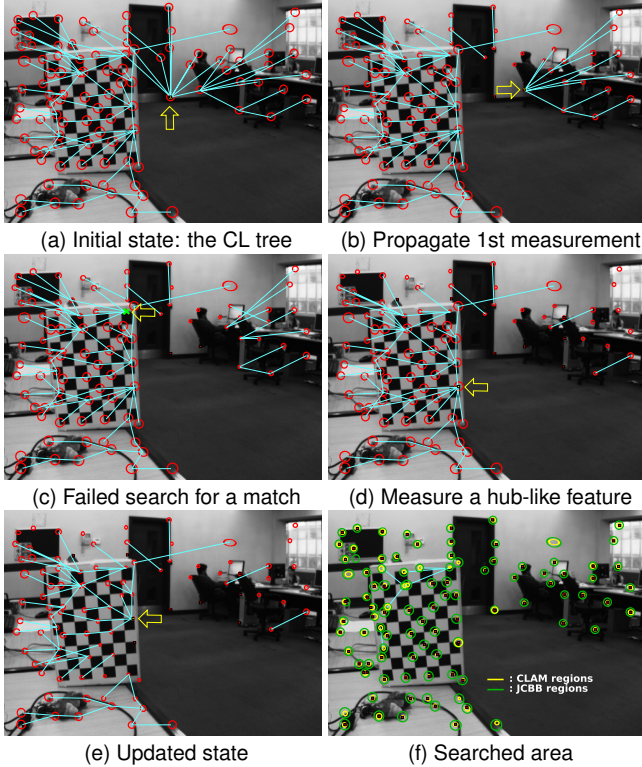(e) Updated state (f) Searched area

Figure 4. Matching using CLAM. The prior distribution and the computed CL tree are illustrated in (a). The arrow points to 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 a match 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.

MI graph is sparsified into the CL tree as shown in 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 4(b), we propagate updates causing reductions in uncertainty of different magnitude at all other nodes depending on their closeness in the tree structure. Since no more information can be passed through a matched node, any links connecting it to the rest of the tree can be cut. As a result, matching is partitioned into subtrees which are highly intercorrelated. Subsequent measurements and updates result in successful matching as shown in (f) where the searched regions of CLAM and traditional methods like JCBB are superimposed.

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 6 allowing real-time matching for hundreds of features. However, due to 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(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.

5.1. The SubAM Algorithm

As in CLAM, in SubAM we construct the CL tree from the input prior $\mathbf{G}_1$ and then form groups of features ('subsets') by considering their proximity in this tree: given a target group-cardinality c (set to 10 for all experiments presented), we place partitions at nodes where the number of their descendants not already grouped is greater than or equal to $c - 1$. Note that this strategy can lead to subsets smaller than c, but we fix the minimum size to $c_{min} = 3$.

Similar ideas to that in SubAM have been used in other recent matching algorithms which perform matching cluster-by-cluster. The N3M algorithm [10] 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. GroupSAC [13], 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.

	SubAM($\mathbf{G}_1$)
1	mixture = $[[1, \mathbf{G}_1]]$ *(each entry is a [λ_i, $\mathbf{G}_i$] tuple)*
2	T = find_tree_of_subsets($\mathbf{G}_1$)
3	$\mathbf{V}$ = [] *(to hold all subsets visited by SubAM)*
4	**for** $\forall \, \mathbf{s}_i \in$ T *(selecting $\mathbf{s}_i$ in a depth-first manner)*
5	mixture = AM(mixture, $\mathbf{s}_i$)
6	$\mathbf{V}$ = append($\mathbf{s}_i$, $\mathbf{V}$)
7	$\mathbf{G}_{best}$ = get_most_probable_$\mathbf{G}$(mixture)
8	**while** $\exists \, \mathbf{f} \in \mathbf{V}$: is_unmeasured($\mathbf{f}, \mathbf{G}_{best}$)
9	$\mathbf{s}_j$ = get_subset_of_$\mathbf{f}$($\mathbf{f}$)
10	mixture = AM(mixture, $\mathbf{s}_j$)
11	$\mathbf{G}_{best}$ = get_most_probable_$\mathbf{G}$(mixture)
12	**end while**
13	**end for**
14	**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 by

considering subsets in tandem. Starting with the root subset, we traverse T in a depth first manner and perform full AM but **only 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 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 all of them. 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 $\mathbf{G}_{best}$.

5.2. SubAM: A Step-By-Step Example

Figure 5 illustrates a step-by-step example of SubAM in action. The CL tree and the subset structure for this frame are shown in (a). Running AM on subset $\mathbf{s}_1$ creates a new AM process to operate on $\mathbf{s}_2$ as shown 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 look for matches. It is worth noting that the bigger the subsets, the closer the approximation but also the more time AM needs to complete. However, if subsets are very small it becomes more likely to generate erroneous hypotheses, so one has to select a suitable subset size c to compromise the desirable speed with the quality preserved.

6. 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 [8] 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 [11]. In all experiments presented we detect FAST features [16] 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 $2\mathrm{ms}$ 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.

(a) CL tree and subsets   (b) Initial AM state in subset $\mathbf{s}_2$

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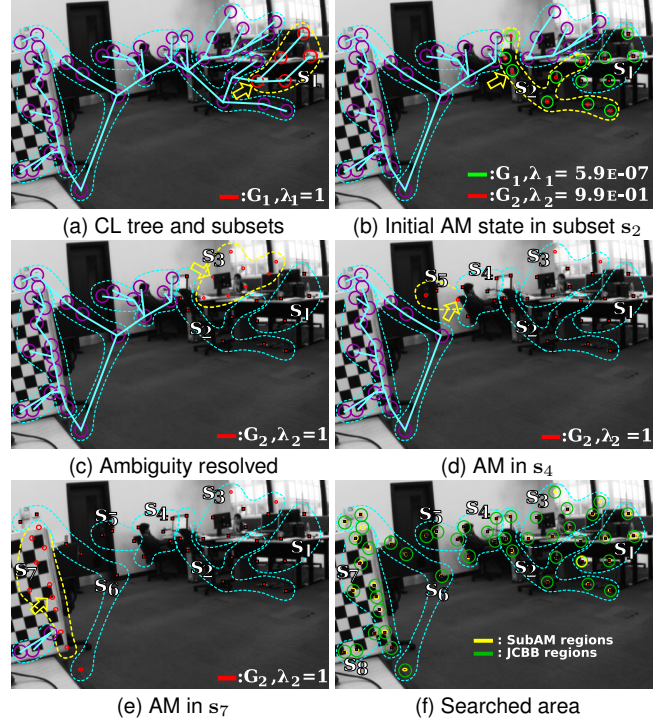
(e) AM in $\mathbf{s}_7$ (f) Searched area

Figure 5. 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 (v) 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 would need to search.

6.1. 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 [8], we need to work a little to obtain them from the alternative keyframe optimisation trackers in the style of PTAM [11] 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 $\mathtt{S}$:

$$\mathtt{S} = \frac{\partial \mathbf{h}(\mathbf{y}_{1:n})}{\partial \mathbf{x}_v} \mathtt{P}_{\mathbf{x}_v}^{(\mathrm{rel})} \frac{\partial \mathbf{h}(\mathbf{y}_{1:n})}{\partial \mathbf{x}_v}^T + \mathtt{R} , \qquad (12)$$
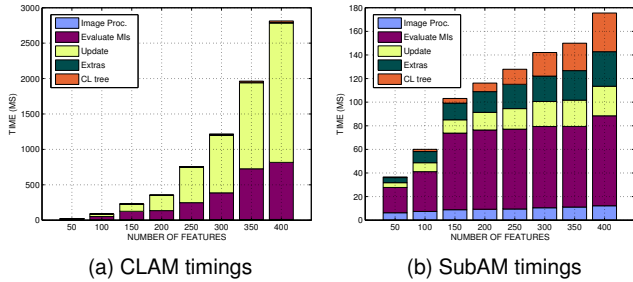
(a) CLAM timings (b) SubAM timings

Figure 7. 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.

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.

6.2. Time Requirements

We have tested the scalability of our algorithms with respect to the number of features matched per frame in keyframe-based SLAM. Figure 6 illustrates the time required to perform matching using AM, CLAM and SubAM for frames where the number of features predicted to be visible ranges from 20 up to 420. As suggested in [3], it is evident that AM is not suitable for real-time matching of more than 50 features per frame, with its curve soon disappearing off the top of the graph. CLAM exhibits a vast reduction in processing time, with real-time performance looking feasible up to the 100–200 feature level and a relatively modest loss of speed beyond this.

However, it is SubAM which really takes performance into the real-time domain for large numbers of features per frame. SubAM demonstrates nearly constant runtime across the range of numbers of features tried, achieving matching of 400 features in only 170ms. Up to 150 features, CLAM and SubAM are comparable but as shown in Figure 7(a) both the Update and the Evaluation of MIs stages consume increasing processing time in CLAM. As explained earlier in Section 5 this is due to the maintenance costs of the tree representation, which gets partitioned into smaller subtrees but these are not explicitly balanced, the decision is instead being driven by MI. The timings breakdown for SubAM in Figure 7(b) suggests that the most significant factor then is the Evaluation of MIs. This is expected as SubAM performs full AM on small subsets of features and we have already seen that this is the most expensive step in AM [3].

Note that the jagged nature of the processing time results in both Figures 6 and 7 is due to the variation in our real
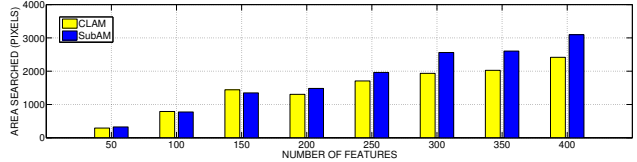

Figure 8. The number of pixels searched for ZNCC matches with respect to the input uncertainty regions. While conventional methods like JCBB and RANSAC need to look for matches within the regions corresponding to the input prior, AM and variants exploit correlations of features to reduce this as shown in Figures 4(f) and 5(f).

dataset. Some frames can be matched very quickly due to inherent lack of ambiguity or a fortunate choice of initial matching candidates, while others require more effort.

6.3. 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. Our new approximations of AM still reduce the searched areas significantly as shown in Figures 4(f) and 5(f), which is to be accredited to the use of the CL tree to identify highly correlated links to preserve. Figure 8 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 in fact rejects some of the matches that CLAM and SubAM seem to accept (comprising no more than 6% of the features matched). This is because of the rigid distribution of AM, whereas both CLAM and SubAM relax this distribution allowing some extra (conservative) freedom in the expected configuration of matches.

7. Conclusions

This paper presented two conservative approximations which enable the efficient use of probabilistic priors in an active matching approach: CLAM approximates the prior distribution by 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 much lower processing time than standard Active Matching. In fact, our SubAM method is able to complete matching of 400 features in 170ms which to our knowledge is faster than what any other fully probabilistic method. Future plans involve further study into graph-theoretic relaxations 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
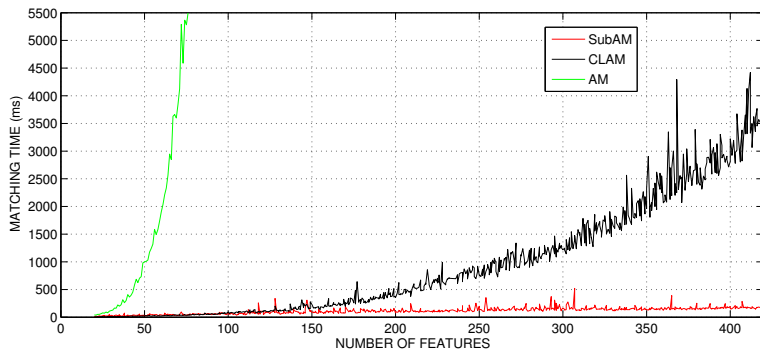
(a) CLAM vs SubAM w.r.t. AM
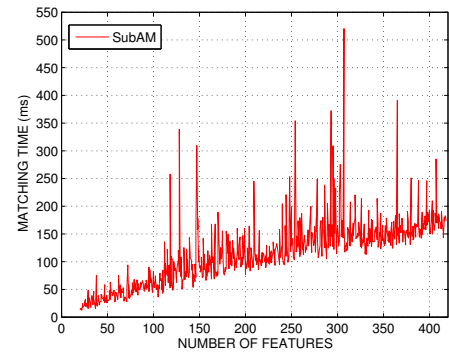
(b) SubAM close-up

Figure 6. Timing requirements per frame for AM, 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 it computationally unfeasible beyond 76 features. CLAM demonstrates a vast speed improvement over AM, but becomes expensive when matching around 200 features per frame and beyond. 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.

question of top-down versus bottom-up methods.

Acknowledgements

References

[1] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., 2006.

[2] M. Chli and A. J. Davison. Active Matching. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2008.

[3] M. Chli and A. J. Davison. Active Matching for visual tracking. *Robotics and Autonomous Systems*, 57(12):1173 – 1187, 2009. Special Issue 'Inside Data Association'.

[4] M. Chli and A. J. Davison. Automatically and efficiently inferring the hierarchical structure of visual maps. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2009.

[5] C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, 1968.

[6] O. Chum and J. Matas. Optimal randomized RANSAC. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 30(8):1472–1482, 2008.

[7] J. Civera, O. G. Grasa, A. J. Davison, and J. M. M. Montiel. 1-point RANSAC for EKF-based structure from motion. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2009.

[8] A. J. Davison, N. D. Molton, I. Reid, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 29(6):1052–1067, 2007.

[9] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[10] S. Hinterstoisser, S. Benhimane, and N. Navab. N3M: Natural 3D markers for real-time object detection and pose estimation. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2007.

[11] G. Klein and D. W. Murray. Parallel tracking and mapping for small AR workspaces. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2007.

[12] J. Neira and J. D. Tardós. Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on Robotics and Automation*, 17(6):890–897, 2001.

[13] K. Ni, H. Jin, and F. Dellaert. GroupSAC: Efficient consensus in the presence of groupings. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2009.

[14] D. Nistér. Preemptive RANSAC for live structure and motion estimation. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2003.

[15] R. Raguram, J.-M. Frahm, and M. Pollefeys. Exploiting uncertainty in random sample consensus. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2009.

[16] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2006.

[17] B. J. Tordoff and D. W. Murray. Guided-MLESAC: Faster image transform estimation by using matching priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 27(10):1523–1535, 2005.

[18] A. Vedaldi, H. Jin, P. Favaro, and S. Soatto. KALMANSAC: Robust filtering by consensus. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2005.

[19] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime TV-L1 optical flow. In *Proceedings of the DAGM Symposium on Pattern Recognition*, 2007.