Real-time affine region tracking and coplanar grouping *

Vittorio Ferrari[1], Tinne Tuytelaars[2], Luc Van Gool[1,2]
[1]Computer Vision Group (BIWI), ETH Zuerich, Switzerland
[2]ESAT-PSI, University of Leuven, Belgium
{ferrari,vangool}@vision.ee.ethz.ch, Tinne.Tuytelaars@esat.kuleuven.ac.be

Abstract

We present a novel approach for tracking locally planar regions in an image sequence and their grouping into larger planar surfaces. The tracker recovers the affine transformation of the region and therefore yields reliable point correspondences between frames. Both edges and texture information are exploited in an integrated way, while not requiring the complete region's contour. The tracker withstands zoom, out-of-plane rotations, discontinuous motion and changes in illumination conditions while achieving real-time performance for a region. Multiple tracked regions are grouped into disjoint coplanarity classes. We first define a coplanarity score between each pair of regions, based on motion and texture cues. The scores are then analyzed by a clique-partitioning algorithm yielding the coplanarity classes that best fit the data. The method works in the presence of perspective distortions, discontinuous planar surfaces and considerable amounts of measurement noise.

1. Introduction

Histogram-based region trackers have proven to run in real-time [6, 4]. The results of these trackers are impressive, but for some applications they lack certain properties. Firstly, they don't provide point correspondences under rotation. Secondly, they achieve high performance by considering a low dimensional search space for the region deformations during tracking (e.g. translation and size only), and are therefore unable to correctly handle the skew and anisotropic scaling effects caused by out-of-plane rotation. On the other hand, trackers that deal with the full set of affine motion parameters have been proposed [3, 13], but they rely on heavier techniques which makes them slow. Moreover, they often need the complete region's contour, what strongly limits the number of regions that can be tracked in a scene.

We present a region tracker that combines these two major properties: it tracks a region under complete affine deformation at real-time speed. Moreover, the tracker does not need closed contours in the images themselves to define the regions; it can deal with large displacements between subsequent frames, and it can recover from a temporary loss of the region. These properties are based on a particular way of searching the affine transformation space which exploits the nature of the two types of regions we are considering. By recovering the complete affine transformation, the tracker yields reliable point-wise correspondences as the region evolves. This is very useful for a number of applications.

In the second part of the paper, we investigate one such application: the detection of planar structures from video sequences. Planes play an important part in 3D reconstruction (e.g. buildings and indoor scenes). Parallax-based scene analysis methods [9, 7] and some special reconstruction techniques [5] need a reference plane. Navigation systems need to find free floor space, etc. For the detection of the planar structures, the set of all tracked regions is partitioned into disjoint *coplanarity classes*. Each class corresponds to a distinct plane in 3D space. Neither the size nor the number of classes is known beforehand. The basic grouping unit is the pair: with the selected, affine region model providing three independent coplanar points per region, a pair of regions is the smallest set sufficient for considering general planar motions (homographies). We define a *coplanarity score* between each pair of regions, based on the combination of a planar motion compatibility cue and a novel texture cue. The cues are computed from the point correspondences generated by the tracker. The score conveys information about the probability that the regions lie on the same plane. We exploit the intrinsic transitivity of the coplanarity property to resolve grouping ambiguities arising from noisy scores by formulating the coplanar grouping problem in *Clique Partitioning* [8] terms. This formulation offers an elegant approach for the treatment of general grouping problems. We introduce a simple, but effective, polynomial time heuristic for its solution.

The structure of the paper is as follows. Section 2 describes the regions that are the basis for the tracking. Section 3 describes the tracker. Section 4 discusses the coplanar grouping of the tracked regions. Section 5 shows some experimental results. Section 6 concludes the paper.

*This research was supported by EC project CIMWOS.

2. Affinely invariant regions

Tuytelaars and Van Gool [12] have proposed a method for the automatic extraction and wide-baseline matching of small, planar regions. These regions are extracted around anchor points and are *affinely invariant*: given the same anchor point in two images of a scene, regions covering the same physical surface will be extracted, in spite of the changing viewpoint. We deal with two of their region types (figure 1): parallelogram-shaped (anchored on corner points) and elliptical (anchored on local intensity extrema). The former are based on two straight edges intersecting in the proximity of the corner. This fixes a corner of the parallelogram (call it **c**) and the orientation of its sides. The opposite corner (call it **q**) is fixed by computing an affinely invariant measure on the region's texture. Elliptical regions are extracted based on the intensity profile along rays emanating from the intensity extremum, without needing edges.

Because of the nature of their respective anchor points and extraction methods, these two types complement each other well and experiments show that hundreds of uniformly distributed regions can be extracted from images of typical indoor and outdoor scenes.

One of the advantages of using these invariant regions is that tracked regions that have been lost due to large occlusions can be recovered by reverting to the matching techniques proposed by Tuytelaars and Van Gool (albeit not yet considered in this paper).

Figure 1: *A parallelogram-shaped and an elliptical region.*

3. Region tracking

Both the geometry-based and intensity-based regions are tracked using the same scheme. In the following we consider tracking a region R from a frame F_{i-1} to its successor frame F_i in the image sequence. First we compute a prediction $\hat{R}_i = A_{i-1}R_{i-1}$ of R_i using the affine transformation A_{i-1} between the two previous frames. An estimate $\hat{\mathbf{a}}_\mathbf{i} = A_{i-1}\mathbf{a_{i-1}}$ of the region's anchor point[1], is computed, around which a circular search space S_i is defined. The radius of S_i is proportional to the current translational velocity of the region. The anchor points in S_i are extracted. These provide potentially better estimates for the region's location. We *investigate* the point closest to $\hat{\mathbf{a}}_\mathbf{i}$ looking for the target region R_i. The anchor point investigation algorithm differs for parallelogram-shaped and elliptical regions and will be explained in the two following subsections. Since the anchor points are sparse in the image, the one closest to the predicted location is, in most cases, the correct one. If not, the anchor points are iteratively investigated, from the closest (to $\hat{\mathbf{a}}_\mathbf{i}$) to the farthest, until R_i is found. This first pruning of the search space helps achieving high speed while keeping the radius of S_i wide enough to ensure tolerance to large image displacements. In some cases it is possible that no correct R_i is found around any anchor point in S_i (e.g.: occlusion, sudden acceleration, failure of the anchor point extractor). When this happens the region's location is set to the prediction ($\mathbf{a_i} = \hat{\mathbf{a}}_\mathbf{i}$), and the tracking process proceeds to the next frame, with a larger S. In most cases this allows to recover the region in one of the next few frames, while avoiding the computationally expensive process of searching F_i further.

[1] Harris corners for geometry-based regions and intensity extrema for intensity-based regions

3.1. Parallelogram-shaped regions

Given a corner point $\mathbf{h}$, the region prediction $\hat{R}_i$, and the region in the previous frame R_{i-1} we want to test if R_i is anchored to $\mathbf{h}$ and, in that case, extract it. The idea is to construct at $\mathbf{h}$ the region most similar to R_{i-1}. The process follows two steps. The first tracks two of the straight region sides exploiting the *geometric information* (edges) of the image, and already yields partial information about R_i. The second step starts from the output of the first, and completes R_i by exploiting *intensity information* (texture).

In the first step a polyline snake with three-vertices recovers two of the sides, but not yet their lengths. We exploit the fact that translating $\hat{R}_i$ so that $\hat{\mathbf{c}} = \mathbf{h}$ automatically provides an estimation of the sides. We initialize the center vertex $\mathbf{v_c}$ of the snake at $\mathbf{h}$ and the other two vertices $\mathbf{v_1}, \mathbf{v_2}$ so that the line segments $\mathbf{v_c v_1}$ and $\mathbf{v_c v_2}$ have the orientation of the predicted region sides (figure 2). The three points are iteratively moved in order to maximize the total sum of gradient magnitudes along the two line segments:

$$E_S(\mathbf{v_c}, \mathbf{v_1}, \mathbf{v_2}) = \sum_{\mathbf{p} \in \mathbf{v_c v_1}} |\bigtriangledown I(\mathbf{p})| + \sum_{\mathbf{p} \in \mathbf{v_c v_2}} |\bigtriangledown I(\mathbf{p})|$$

where $|\bigtriangledown I(\mathbf{p})|$ is the image gradient magnitude at pixel $\mathbf{p}$. The snake can deform only by hinging around $\mathbf{v_c}$ and the length of the line segments is kept fixed (we are interested in their orientation only). These constraints reduce the number of DOF to four, thereby reducing the search space and improving efficiency.

The optimization process is efficiently implemented by a Dynamic Programming algorithm inspired by [1, 14]. The algorithm has a higher probability of being attracted toward

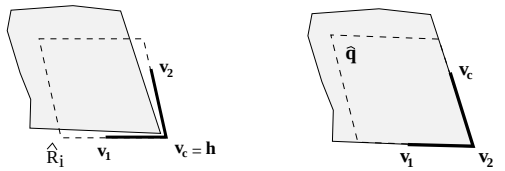

Figure 2: *Left: Polyline snake initialization. Right:* $\hat{\mathbf{q}}$ *initialization.*

contours than the traditional snake implementation [10], and it is ensured to converge [1]. In practice $\mathbf{h}$ is often very close (a few pixels) to the intersection point $\mathbf{c}$ of the target region sides. Hence our initialization is often very good and this reduces the number of iterations and the risk of being attracted by nearby distractor edges.

The tracked region sides lift four DOF: the two coordinates of $\mathbf{c} = \mathbf{v_c}$ and the orientations of the two sides. These correspond to the translation, rotation and skew components of the affine transformation A_i mapping R_{i-1} to R_i. This is all the information we can extract from the geometric features of the image. The two remaining DOF correspond to the position of the point $\mathbf{q}$ (they arise from the scale components of A_i) and are derived from the texture content of the region by the second step of the algorithm.

An initial estimate $\hat{\mathbf{q}}$ is obtained by aligning $\hat{R}_i$ on the structure formed by $\mathbf{v_1}, \mathbf{v_c}, \mathbf{v_2}$, so that $\hat{\mathbf{c}} = \mathbf{v_c}$ and the sides are oriented like $\mathbf{v_c v_1}$, $\mathbf{v_c v_2}$ (figure 2). This estimation is refined by moving $\hat{\mathbf{q}}$ so as to maximize the similarity between the resulting region $R_i(\hat{\mathbf{q}})$ and the region in the previous frame R_{i-1}. As a similarity measure we use the normalized cross-correlation between R_{i-1} and $R_i(\hat{\mathbf{q}})$ after aligning them via $A(\hat{\mathbf{q}})$, the affine transformation mapping R_{i-1} onto $R_i(\hat{\mathbf{q}})$. Therefore, $\mathbf{q}$ is obtained as the location of $\hat{\mathbf{q}}$ maximizing the objective function:

$$E_c = \mathrm{CrossCorr}(A(\hat{\mathbf{q}})R_{i-1},\ R_i(\hat{\mathbf{q}})) \qquad (1)$$

Notice that this similarity measure is invariant not only under geometric affine transformations, but also under linear transformations of the pixels intensities. This makes the tracking process relatively robust to changes in illumination conditions. The maximization process is implemented by Gradient Descent, initialized on $\hat{\mathbf{q}}$, where at each iteration $\hat{\mathbf{q}}$ is moved 1 pixel in the direction of maximal increase. Typically $\hat{\mathbf{q}}$ is initialized close to the absolute maximum, because most of the variability of the affine transformation is lifted by the sides tracking step. This strongly reduces the risk of converging toward a local maximum and keeps the number of iterations low. Extensive experiments confirm this consideration and indicate that, in most cases, 3 iterations are enough.

At the end of the second step, the most similar region to R_{i-1} anchored to $\mathbf{h}$ is constructed. This does not mean that it is the correct region though, as $\mathbf{h}$ could just be the wrong corner. Hence, as final verification we check if the maximum cross-correlation value is above a predefined threshold (typically 0.9), otherwise the algorithm proceeds to the next corner.

3.2. Elliptical regions

Let us now focus on the elliptical regions. Given an intensity extremum $\mathbf{i}$, the region prediction $\hat{R}_i$ and the region in the previous frame R_{i-1}, is R_i anchored to $\mathbf{i}$? Since the elliptical regions of [12] exploit only the raw intensity pattern of the image and do not rely on the presence of nearby edges, we can no longer devise a two-step search strategy like for the parallelogram-shaped case. A natural alternative would be to look for the complete set of 6 parameters of A_i simultaneously, by minimizing a cross-correlation based criteria similar to equation (1) (as proposed in [3]). The search process could be initialized from the affine transformation A_{i-1} between the two previous frames (possibly translated so that $\mathbf{c} = \mathbf{i}$). The problem is that searching for an optimum in this six-dimensional space, starting from an imprecise initialization, would probably take too much computation power to be achieved in real-time.

We exploit instead the property that R_i can be extracted from F_i *independently*, provided we are considering the correct intensity extremum. In a first step, R_i is extracted around $\mathbf{i}$ using an optimized implementation of the algorithm described in [12]. The second step could consist of verifying that R_i cross-correlates with R_{i-1} above a predefined threshold. Unfortunately, since an ellipse has only five DOF, it is not possible to directly compute A_i from R_{i-1} and R_i. The missing DOF corresponds to a free rotation in the ellipse plane around its center. We want to avoid the approach of [12], which consists of an exhaustive search for the rotation maximizing the cross-correlation, because of its inefficiency. We propose an alternative based on a photometric invariant version of the axis of inertia. First $\hat{R}_i$ is affinely mapped to a reference circular region O. The major and minor axes of inertia are then extracted (figure 3) as the lines passing from the center of O with orientations θ_{max}, θ_{min} defined by the solutions of:

$$\tan^2(\theta) + \frac{m_{20} - m_{02}}{m_{11}} \tan\theta - 1 = 0 \qquad (2)$$

with m_{pq} the (p, q) order moment *centered on the region's geometric center*. Equation (2) differs from the usual definition of the axis of inertia by the use of these moments instead of moments centered on the *center of gravity* weighted with image intensity. This makes them invariant to affine changes of the intensities. These axes are *invariant under rotation*, in the sense that they will cover the same part of the region after a rotation. Mapping the axis back to the original elliptical region will now provide two affinely invariant lines, and their intersection points with the ellipse. The mapping of the center of the ellipse and these intersections allow us to compute A_i; the cross-correlation test

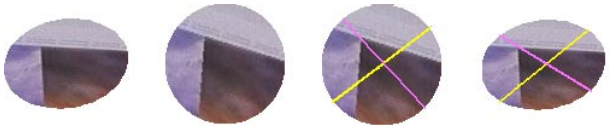

Figure 3: *From left to right: original elliptical region; mapped to circular region; axes of inertia in circular region; axes mapped back to elliptical region.*

can follow and, if failed, the tracker can proceed to the next intensity extremum.

4. Coplanar grouping

In this section, the coplanarity cues and score are introduced (subsection 4.1), followed by an alternative graph-theoretic approach (subsection 4.2) ensuring effective coplanar grouping. Note that from now on, perspective effects are fully taken into account. This is important, since the affine approximation is only valid on a local scale.

4.1. Coplanarity cues and score

Let R, S be two regions tracked in n frames. Consider three points $\mathbf{r}_i^1, \mathbf{r}_i^2, \mathbf{r}_i^3$ characterizing R in frame F_i. If R is parallelogram-shaped, these are three corners (all but $\mathbf{q}$). In the elliptical case, these are the center $\mathbf{c}$ and the two intersections of the axes of inertia with the ellipse.$\{\mathbf{r}_i^p\}_{p=1..3}$ completely define R in F_i and the correspondences between $\{\mathbf{r}_i^p\}$ and $\{\mathbf{r}_j^p\}$ implicitly encode the affine transformation of R between frames F_i, F_j. We assume analogous definitions for S.

We introduce two numerical coplanarity cues that will later be integrated in a single coplanarity score. The first cue is purely based on the motion of R and S between the first and last frames (F_1, F_n). We compute by least squares approximation the 2D homography H that best maps the 6 points in F_1 (the set $\{\mathbf{r}_1^p\}_{p=1..3} \cup \{\mathbf{s}_1^p\}_{p=1..3}$) to their corresponding points in F_n (the set $\{\mathbf{r}_n^p\}_{p=1..3} \cup \{\mathbf{s}_n^p\}_{p=1..3}$). If R, S are coplanar, H correctly describes the motion of both regions. We measure this via the following error:

$$c_m = \frac{1}{6}\sum_{i=1}^{3}(\mathrm{d}(H\mathbf{r}_i^1,\ \mathbf{r}_i^n) + \mathrm{d}(H\mathbf{s}_i^1,\ \mathbf{s}_i^n)) \qquad (3)$$

where $\mathrm{d}(\mathbf{p}_1, \mathbf{p}_2)$ is the euclidean distance between points $\mathbf{p}_1, \mathbf{p}_2$. Assuming noise-free data, if R, S are coplanar $c_m = 0$; c_m is related to the difference between the position and orientation of the R plane and the S plane in 3D space. Hence we use expression (3) as a cue about the potential coplanarity of two regions: the smaller c_m, the higher the chances of R, S being coplanar.

While the motion cue is based completely on local information, the second cue takes a larger view and considers the image data *between* R and S. The idea is to check if R, S are coplanar *and* located on a continuous, unoccluded

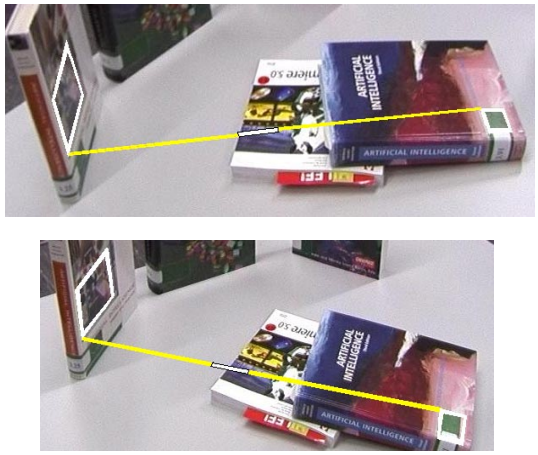

Figure 4: *Top: Non-coplanr pair of regions. Bottom: Other view. The central line segment is misaligned.*

physical planar surface in 3D space. In order to take into account a small, but representative, sample of the surface between R and S we consider three lines connecting corresponding characteristic points of the two regions.To keep the notation simple, we restrict the explanation to the first line. Consider the line $\mathbf{l}_1$ connecting the first characteristic points R_1^1, S_1^1 of the two regions in the first frame. Divide $\mathbf{l}_1$ in $s = \frac{\mathrm{d}(R_1^1, S_1^1)}{m}$ segments of equal length m, denote them $\{\mathbf{l}_1^j\}_{j=1..s}$. Let $\{\mathbf{l}_n^j\}_{j=1..s}$ be a list of segments in F_n, whose coordinates are obtained by projecting $\{\mathbf{l}_1^j\}_{j=1..s}$ via H. We are interested in the similarity between corresponding segments in the two frames, and in particular in the *least* similar one:

$$\min_{j=1..s} \mathrm{CrossCorr}(\mathbf{l}_1^j,\ \mathbf{l}_n^j) \qquad (4)$$

where $\mathrm{CrossCorr}(\mathbf{l}_1^j,\ \mathbf{l}_n^j)$ is the value of the normalized cross-correlation of the intensity profile on line segment $\mathbf{l}_1^i$ with the one on $\mathbf{l}_n^j$. Only if R, S are coplanar and located on a continuous, unoccluded planar surface, will *all* segments score well. If R, S are not coplanar, segments close to the region may still score well: H describes the motion in that zone best, and probably the neighboring area is planar. Nevertheless, central segments will tend to be misaligned as H can not correctly describe their motion, and therefore have low scores (figure 4). Taking the *least* scoring segment ensures the detection of exactly those significant cases.

We define the second coplanarity cue c_t as the average of expression (4) over the three lines connecting $\mathbf{r}_1^1$ with $\mathbf{s}_1^1$, $\mathbf{r}_1^2$ with $\mathbf{s}_1^2$ and $\mathbf{r}_1^3$ with $\mathbf{s}_1^3$. Coplanar pairs located on a discontinuous planar surface (e.g.: the surface is interrupted between the two regions) will tend to have a low c_t; clearly, this should not be interpreted as an indication that two regions are not coplanar. Hence, we use c_t only to *increment* the total coplanarity score. Nevertheless the role of this cue must not be underestimated, as we expect it to sub-

stantially reinforce the total score of a significant portion of the coplanar pairs, hence helping the forthcoming grouping algorithm.

From the above considerations, we define the *coplanarity score* of a pair:

$$w = (h_t - c_m) + \begin{cases} c_t h_t & \text{if } c_t > 0.6 \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

where $h_t > 0$ is an homography error threshold, acting like a splitting point between positive and negative scores ($w < 0$ suggests that R, S are not coplanar, while $w \geq 0$ suggests they are) and defining the maximal positive contribution of each cue. The range of w is $]-\infty, 2h_t]$. In practice though, for $h_t = 2.0$ scores above 1.0 already indicate very probable coplanarity.

4.2. Grouping algorithm

Taken one by one, the coplanarity scores are unreliable because they arise from very limited, noisy information. In practice it happens that a coplanar pair has $w < 0$ (false negative), and the contrary (false positive). Nevertheless, taken altogether, the scores clearly contain reliable information about the correct grouping. We want to be robust to misleading *local* information by exploiting the *transitivity* of coplanarity: if R, S are coplanar and S, T too, then R, T *must* be coplanar[2]. How can transitivity help us ? Consider a scene with three regions. Let w_{ij} be the score of the pair (i, j) composed of the ith and jth region. Given the scores $w_{12} = 9, w_{13} = 7, w_{23} = -3$, and the transitivity property, the best choice is to group the three regions together (w_{23} is a false negative score). Next, we formulate the coplanar grouping problem so as to exploit transitivity to detect and avoid false scores .

We propose to construct a complete graph G where each vertex represents a region and edges are weighted with the coplanarity scores. We partition G into completely connected disjoint subsets of vertices (cliques) so as to maximize the total score on the remaining edges (Clique Partitioning, or CP). The transitivity property is ensured by the clique constraint: every two vertices in a clique are connected, and no two vertices from different cliques are connected. Hence, the generated cliques correspond to the best possible coplanar grouping (given the cues). The CP formulation of coplanar grouping is made possible by the presence of positive *and* negative weights: they naturally lead to the definition of a *best solution* without the need of knowing the number of cliques (planes) or introducing any artificial stopping criteria like in other graph-based approaches to grouping based on strictly positive weights [11, 2]. On the other hand, our approach needs a parameter h_t that determines the splitting point between positive and negative scores. But, in our context, this parameter is easily determined and experiments show the optimal solution of CP to be generated for a wide range of h_t.

[2]Coplanarity is reflexive, symmetric and transitive (an *equivalence relation*).

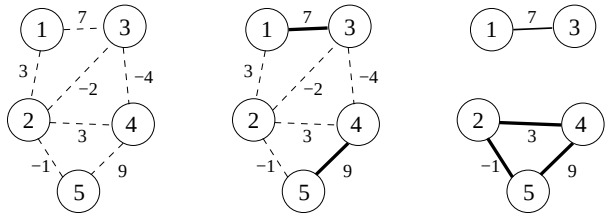

Figure 5: *An example graph and two iterations of CP. Not displayed edges have zero weight.*

CP can be solved by Linear Programming [8] (LP). Let w_{ij} be the weight of the edge connecting (i, j), and $x_{ij} \in \{0, 1\}$ indicate whether the edge exists in the solution. The following LP can be established:

$$\begin{array}{ll} \text{maximize} & \sum_{1 \leq i < j \leq n} w_{ij} x_{ij} \\ \text{subject to} & x_{ij} + x_{jk} - x_{ik} \leq 1, \quad \forall 1 \leq i < j < k \leq n \\ & x_{ij} - x_{jk} + x_{ik} \leq 1, \quad \forall 1 \leq i < j < k \leq n \\ & -x_{ij} + x_{jk} + x_{ik} \leq 1, \quad \forall 1 \leq i < j < k \leq n \\ & x_{ij} \in \{0, 1\}, \quad \forall 1 \leq i < j < k \leq n \end{array} \quad (6)$$

The inequalities express the clique constraints (transitivity), while the objective function to be maximized corresponds to the sum of the intra-clique edges. Unfortunately CP is an NP-hard problem [8]: LP (6) has worst case exponential complexity in the number n of vertices (regions), making it impractical for large n.

The challenge is to find a *practical* way out this complexity trap. The correct partitioning of the example in figure 5 is $\{\{1, 3\}, \{2, 4, 5\}\}$. A simple greedy strategy merging two vertices (i, j) if $w_{ij} > 0$ fails because it merges $(1, 2)$ as its first move. Such an approach suffers from two problems: the generated solution depends on the *order* by which vertices are processed and it looks only at *local* information. We propose the following iterative heuristic. The algorithm starts with the partition $\Phi = \{\{i\}\}_{1 \leq i \leq n}$ composed of n singleton cliques each containing a different vertex. The function $m(c_1, c_2) = \sum_{i \in c_1, j \in c_2} w_{ij}$ defines the cost of merging cliques c_1, c_2. We consider the functions $b(c) = \max_{t \in \Phi} m(c, t)$ and $d(c) = arg\max_{t \in \Phi} m(c, t)$ representing, respectively, the score of the best merging choice for clique c and the clique with whom to merge. We merge cliques c_i, c_j if and only if $d(c_i) = c_j$ and $d(c_j) = c_i$ and $b(c_i) = b(c_j) > 0$. In other words, two cliques are merged only if each one represents the best merging option for the other and if merging them increases the total score. At each iteration the functions $b(c), d(c)$ are computed, and all pairs of cliques fulfilling the criteria are merged. The algorithm runs until no two cliques can be merged.

Figure 5 shows an interesting case. In the first iteration $\{1\}$ is merged with $\{3\}$ and $\{4\}$ with $\{5\}$. Notice

how $\{2\}$ is, correctly, not merged with $\{1\}$ even though $m(\{1\},\{2\}) = 3 > 0$. In the second iteration $\{2\}$ is correctly merged with $\{4,5\}$, resisting the (false) attraction of $\{1,3\}$ ($b(\{1,3\},\{2\}) = 1$, $d(\{1,3\}) = \{2\}$). The algorithm terminates after the third iteration because $m(\{1,3\},\{2,4,5\}) = -3 < 0$. The second iteration shows the power of CP. Vertex 2 is connected to unreliable edges (w_{12} is false positive, w_{25} is false negative). Given vertices $\{1,2,3\}$ only, it is not possible to derive the correct partitioning $\{\{1,3\},\{2\}\}$; but, as we add vertices $\{4,5\}$, the *global information* increases and CP manages to get the correct partitioning out of it.

The proposed heuristic is *order independent*, takes a more global view than a direct greedy strategy, and resolves several ambiguous situations while maintaining polynomial complexity (worst case $O(n^3)$, but faster in practice). In the first iterations, being biased toward very positive weights, the algorithm risks to take wrong merging decisions. Nevertheless our particular merging criterion ensures this risk to quickly diminish with the size of the cliques in the correct solution (number of regions in a plane) and at each iteration, as the cliques grow and increase their resistance against spurious weights. Moreover, in our application, very positive scores arise only when *both* cues score well and are therefore much more *reliable* than negative scores, which are often due to large homography errors due to measurement noise. In summary, the algorithm uses reliable data as seeds, and then proceeds to the robust construction of the correct solution by filtering out spurious data.

5. Experiments

5.1. Tracking

We present two sequences demonstrating the tracker's qualities. In both sequences, the images of the tracked planar patches are put into *complete correspondence* along the frames. This allows to derive three reliable point correspondences per region between any pair of frames.

The *Book* sequence (figure 6) features a parallelogram-shaped and an elliptical region undergoing simultaneous rotation and scaling. The physical planar patch covered by each region is accurately tracked along the sequence, as proven by the constant high cross-correlation scores (figure 6). The axes of inertia of the elliptical region reliably follow the rotation of the book. The computational performance[3] meets the real-time expectations (figure 6). With an average cost per frame of 0.018 seconds, the parallelogram-shaped region is tracked particularly efficienlty. The difference with the elliptical one (average 0.034 seconds per frame) is mostly due to the different (and currently slower) algorithm needed for the anchor point extraction.

[3] All experiments performed on a Sun UltraSparc-IIi, 440 MHz.

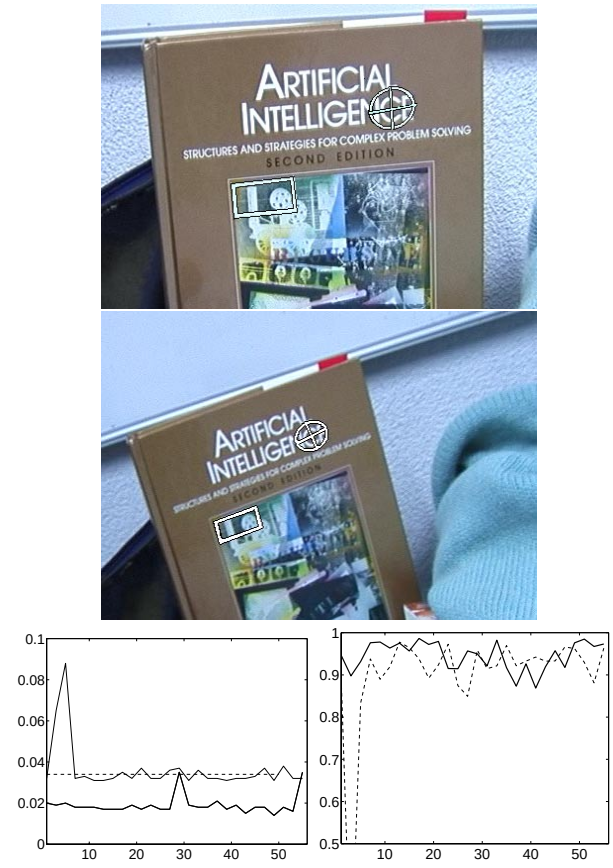

Figure 6: *Two frames of the Book scene. Bottom left: seconds (y axe) to track each frame (x axe) for parallelogram-shaped region (thick) and elliptical one (thin; peak correspond to a temporary loss). Bottom right: cross-correlation scores.*

Control of out-of-plane rotation and robustness to discontinuous motion are exemplified in the *Poster* sequence (figure 7). The region rotates significantly around the vertical axis, causing skew and anisotropic scaling effects in the image. The tracker was able to handle this situation by correctly transforming the 2D region's shape: despite the very different viewpoints of frames 1 and 200, the region is covering the same physical surface. In our application, this is a required feature: the region deformation yields precious information about a plane orientation. As it was taken with a handheld camera, the sequence contains a certain amount of irregular motion: the region sometimes bruskly changes direction and velocity, making it hard to predict the next location accurately. Moreover, we increased the average velocity even further by subsampling the sequence to contain only every fourth frame. The total effect is a very discontinuous motion (irregular and fast) where the region moves fast between each frame and where the predicted location is often far from the correct one. The tracker successfully managed to find the region in every frame despite predic-

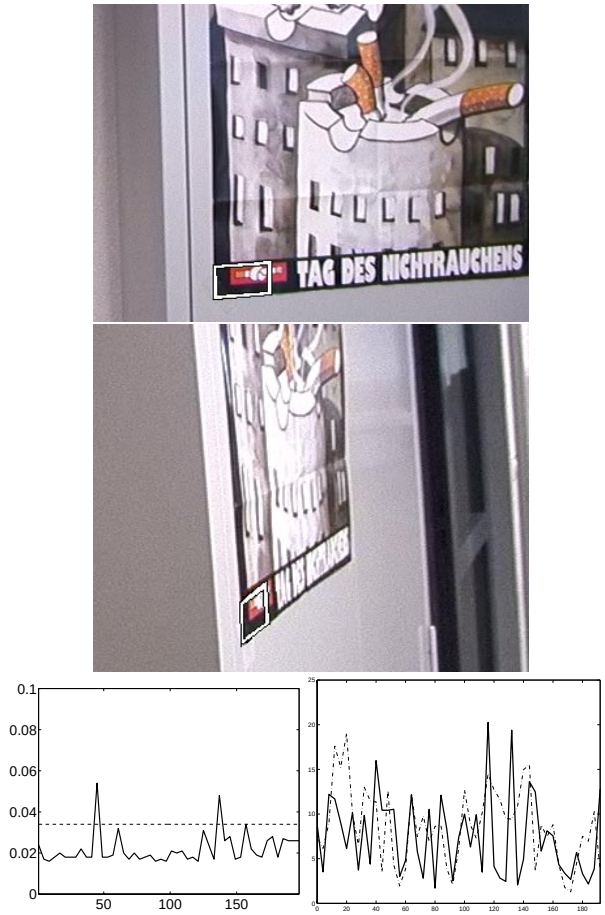

Figure 7: *First and last frames of the Poster scene. Bottom-left: seconds per frame; bottom right: prediction error (thick) and velocity.*

tions that were off by up to 20 pixels from the target (figure 7). The sequence was processed in 0.022 seconds per frame on average, and only in two frames the tracker needed more than 0.034 seconds (peaks in figure 7).

The ability to track fast moving regions is important when the whole sequence is available beforehand (offline tracking): by considering only a fraction of the frames, substantial reductions of the computational costs can be achieved. For online tracking, this allows to track several regions simultaneously.

The experiments confirmed that the tracker efficiently provides the reliable and accurate point correspondences needed by our application under general motion conditions.

5.2. Grouping

The coplanar grouping algorithm has been tested on several scenes; we exemplify its performance by the scene in figure 9. We consider 12 regions: 3 on the Artificial Intelligence book in the middle of the scene (AI plane), 4 on the journals lying on the table (Table plane) and 5 on the two cardboard

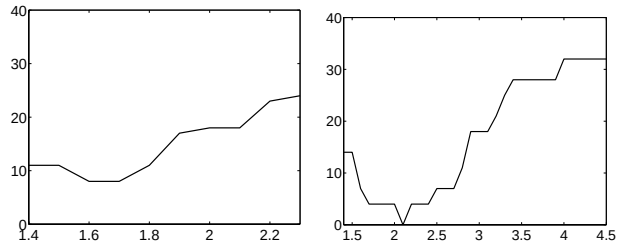

Figure 8: *Left: False score percentage in function of h_t in the complete scene case. Right: AI and Box case.*

boxes on the background (Box plane). The scene presents many difficulties: the Table and Box planes are not visible as continuous planar surfaces, the Table plane shows considerable perspective effects, the AI plane is slanted and intersects the table in the image. Moreover, the camera moves little during the sequence, resulting in a tight baseline and therefore less motion information for our cues. However, it is the small region at the bottom of AI that makes this scene particularly challenging. Since it is located very close to the Table regions and the two planes orientations (at the local scale) are similar, its motion is compatible with the Table motion, and the coplanarity scores are positive whereas they should be negative. Based on this local information only, we could not separate it from the Table regions.

Despite these difficulties, the algorithm produced the correct grouping, and exploited the attraction of the topmost AI regions to bring the bottom one in place. The correct solution for the whole scene was produced for h_t ranging in $[1.4, 2.3]$, which is a considerably wide interval, considering that the homography error for all pairs formed by regions in AI and Table ranges from 0.6 to 3.0. For $h_t < 1.4$ the AI plane was split (i.e.: not all AI regions were grouped together), and for $h_t > 2.3$ AI was merged with Table. As shown by figure 8, picking values for h_t far from the optimimum causes a deterioration of the coplanarity score, resulting in an increase of false scores (i.e.false positives and false negatives scores). Nevertheless, the algorithm resisted even significant increases: for $h_t = 2.3$, 24% of the coplanarity scores are false, with the grouper still delivering the correct partitioning. The range of h_t yielding the correct solution is even wider in simpler cases; considering only the AI and Box regions, this is $h_t \in [1.4, 4.5]$. For $h_t = 4.5$, 32% of the scores are false. Again, this is a wide interval, given the homography errors for this scene ranges in $[1.18, 5.22]$. These figures show Clique Partitioning's ability to extract the correct information in the presence of considerable amounts of noise. We verified the correctness of the clique paritioning solutions provided by our heuritstic by solving the problem also with LP. In both cases (complete scene, only AI and Box), for all values of h_t, LP generated the same solutions.

Various experiments on several scenes reinforce the re-

Figure 9: *First and last frame of test sequence for coplanar grouping. Notice the moderate baseline.*

ported observations about the h_t ranges for which the correct solution is produced; hence we assert our algorithm is little sensible to the exact choice of h_t. Setting $h_t = 2.0$ resulted in correct partitioning of almost all scenes.

Finally, we tested both LP and our heuristic on random instances of the clique partitioning problem. We generated 100 times a 21 vertices graph consisting of 3 cliques of 7 vertices each; intra-clique weights were uniformly distributed in $[-3, 9]$, while inter-clique weights in $[-9, 3]$. On the average, the number of missclassified vertices was very low: 0.5 for LP and 0.51 for the heuristic. Both algorithms produced the same partitioning (either correct or wrong) 96% of the times. The average percentage of false scores, computed over the set of correctly solved instances, was 24.92% (expected 25%). These encouraging results show CP robustness to noise and support our heuristic as a qualitative approximation.

6. Conclusions

In this contribution, a real-time affine region tracker was proposed. Among its most important features are its robustness to large out-of-plane rotations and discontinuous motion, its speed and the fact that it brings the tracked regions into complete correspondence. This last property is especially useful for the application described in the second half of the paper, where a robust method for grouping coplanar regions based on Clique Partitioning was proposed. A simple, but effective, polynomial time heuristic was introduced, that proved to yield the correct optimal solution in most practical situations. Although demonstrated on the application of coplanar grouping, the grouper is by no means restricted to this application.

References

[1] A.A. Amini, T.E. Weymouth and R.C. Jain. Using Dynamic Programming for Solving Variational Problems in Vision. In *IEEE Trans on Patt Analysis and Machine Intell*, 12(9):855-866, 1990.

[2] J. Aslam, A. Leblanc and C. Stein A new approach to clustering In *Workshop on Algorithm Engineering*, 2000.

[3] B. Bascle and R. Deriche Region tracking through image sequences. In *Int. Conf. on Comp. Vis.*, pp. 302-307, 1995.

[4] S. Birchfield Elliptical Head Tracking Using Intensity Gradients and Color Histograms. In *IEEE Conf. on Comp. Vis. and Pat. Rec.*, pp. 232-237, 1998.

[5] S. Carlsson and C. Rother Linear multi-view reconstruction and camera recovery In *Int. Conf. on Comp. Vis.*, 2001.

[6] D. Comaniciu, V. Ramesh and P. Meer Real-Time Tracking of Non-Rigid Objects using Mean Shift. In *IEEE Conf. on Comp. Vis. and Pat. Rec.*, pp. 142-149, 2000.

[7] A. Criminisi, I. Reid and A. Zeissermann Duality, Rigidity and Planar Parallax In *5th European Conf. On Comp. Vis.*, pp. 846-861, 1998.

[8] R. L. Graham, M. Groetschel and L. Lovasz (editors) Handbook of Combinatorics volume 2, Elsevier, pp. 1890-1894, 1995

[9] M. Irani, P. Anandan, and D. Weinshall From Reference Frames to Reference Planes: Multi-View Parallax Geometry and Applications. In *5th European Conf. On Comp. Vis.*, pp. 829-845, 1998.

[10] M. Kass, A. Witkin and D. Terzopoulos Snakes: active contour models In *Int. Journal of Comp. Vis.*, pp. 321-331, 1988

[11] J. Shi and J. Malik Normalized Cuts and Image Segmentation In *IEEE CVPR*, pp. 731-737, 1997..

[12] T. Tuytelaars and L. Van Gool Wide Baseline Stereo based on Local, Affinely invariant Regions *British Mach. Vis. Conf.*, pp. 412-422, 2000.

[13] C. Toklu, A. M. Tekalp A.T. Erdem Simultaneous alpha map generation and 2-D mesh tracking for multimedia applications In *Int. Conf. on Image Proc.*, pp. 113-120, 1997.

[14] N. Ueda and K. Mase Tracking Moving Contours Using Energy-Minimizing Elastic Contour Models. In *2nd European Conf. On Comp. Vis.*, pp. 453-457, 1992.