Markerless Augmented Reality with a Real-time Affine Region Tracker

V. Ferrari[1], T. Tuytelaars[2] and L. 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 system for planar augmented reality based on a new real-time affine region tracker. Instead of tracking fiducial points, we track planar local image patches, and bring these into complete correspondence, so a virtual texture can directly be added to them. Moreover, the local image patches can be extracted in an invariant way, even without any a priori information from previous frames. Hence it is possible to use them as natural beacons, that can be used to recognize the scene and to identify the individual patches. This results in a powerful system, that can work without artificial markers or fiducial points and with a minimal amount of user interference.

1. Introduction

Augmented Reality superimposes information – textual or figural – onto parts of the real world. Issues to be resolved are *what* should be shown *where*, and *how*. The latter problem is especially important when the visual appeal of the result is crucial. Then much effort has to go into seamlessly fitting the information into the scene, both geometrically and photometrically (occlusions, shadowing, mutual reflections, chromatic adaptation to scene illumination, ...). In this paper we only deal with the two other issues: what to place where. We even simplify the problem further by assuming that planar textures are to be superimposed onto planar parts in the scene (but these planar parts can be small, so, in contradistinction to e.g. Simon *et al.* [13], we do *not* assume that dominant planes are present.)

Yet, even under such simplified conditions these problems do not become trivial. Suppose a field worker has to perform a job somewhere in a building. The engineer has gone through the building the previous day, and has prepared virtual annotations that indicate what is to be done at different locations. When the worker enters a room, his AR system should superimpose this information through a head-mounted display, e.g. a todo list, or a video demonstrating the expected manipulation. The system should therefore recognize where it is, and which information is to be superimposed where. As mentioned, we assume that planar textures are to be superimposed onto planar surface patches. In fact, the information need not be projected right on top of these patches, but it will be rigidly connected to them. The following tasks have to be solved:

1. recognize the overall position of the user (context)
2. recognize the individual patches
3. track the patches and superimpose their virtual textures

The first two steps can be considered to be a kind of initialization. The first step, finding the user's overall position within a possibly large scene, need not be solved entirely based on visual information. Vision could be combined with GPS and we plan such integration. In our current implementation this step is carried out pure visually. In fact, the first two steps are combined. The joint recognition of a set of patches as a configuration yields the location. This step is not carried out in real-time. Once the patches have been recognized, the system displays their annotations. The user can select a region, and then the system allows to track and augment the selected region in real-time. Hence, the correct, superimposed information rigidly moves with the patch.

Due to the real-time demands of AR, researchers often have to take refuge to putting 'markers' or 'landmarks' into the scene [1]. They are designed in a way that they are easy to detect. Detection may be eased by giving them distinctive photometric characteristics (e.g. colored markers [9, 10], LEDs [6], and retroreflective markers [5, 12]) or special shapes (like squares [9, 15], circles [14], dot codes [8] or even 2D barcodes [11]). The novelty of the proposed work is that we attempt to use patches of the natural scene as landmarks. The extraction of these landmarks is automatic and robust against changes in viewpoint and illumination. A single patch can be tracked in real-time, where out-of-plane rotations as well as major shifts between frames are allowed. During tracking accurate correspondences between points of the landmark patches are established, because the complete affine transformations between frames are recovered

(in contradistinction with other real-time region trackers, often resorting to translation and scale only [20, 19]). This allows to annotate the landmark patch with a texture, that is locked to it and deforms as if it were part of the physical scene, mimicing the same 3D motion as the patch undergoes.

This differs with earlier contributions with natural landmarks in that there is no initialization stage based on fiducial markers [7] or interactively indicated points needed for calibration [13], in that the planarity assumption is limited to the small landmark patch instead of an extended part of the image [13], in that no 3D model of the object is required to predict the deformations of the natural landmark regions [16], in that the 3D camera/object motion is not constrained [2, 3], and in that the computations run real-time on a single, mainstream workstation processor (Sun UltraSparc-IIi, 440 MHz) [16]. On the other hand, the virtual texture is purely planar and is simply mapped onto the landmark plane. Nevertheless, for quite a few AR applications this is sufficient. 3D shapes could be added based on the joint tracking of several natural landmarks. However, this is not yet considered.

The landmarks correspond to regions that are 'good for tracking' according to the criteria of Shi and Tomasi [4] (similar work in [26]). A difference with their regions is that the affine deformations are not determined through a reference to the initial frame. Frames play a more symmetric role. This seems better as the first frame could be a frame that was taken from a particularly large distance or under a grazing angle. In that case the texture within the region does not support cross-correlations well. Also, in our case there is no double process where tracking assumes the flow field to amount to a translation whereupon a more precise affine deformation is determined, but, instead, affine deformations are determined directly by the tracking process.

The paper is organized as follows. Section 2 explains the specific characteristics of the image patches used and how to extract them from an image without a priori information. Section 3 explains the scene recognition and patch identification based on affine moment invariants. Section 4 looks at the real-time tracking. Section 5 deals with the virtual pattern overlay. Section 6 discusses some experimental results, and section 7 concludes the paper.

2. Affinely invariant regions

In order to use patches of the natural scene as beacons instead of artificial markers, a method is required to extract *exactly the same* scene patch, in spite of changing viewpoint and/or illumination conditions. This can be achieved through the use of *affinely invariant regions* [23, 24]. With 'invariant' we mean that they automatically deform their shape with changing viewpoint as to keep on covering identical physical parts of a scene (under the assumption of local planarity). In addition to the affine geometric invariance, they are also invariant to linear photometric changes, with a different offset and scale-factor for each colorband.

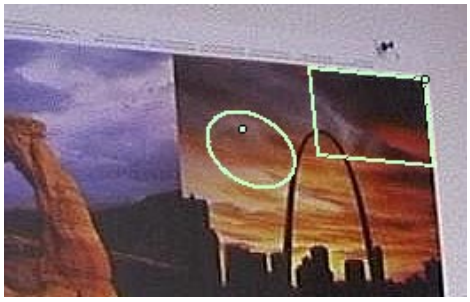

Figure 1. *A parallelogram-shaped and an elliptical region.*

These local invariant regions can be distinguished based on a feature vector of moment invariants; hence they can be recognized on an individual basis. Moreover, they also turn out to be good features to track. While tracking, we can exploit specific knowledge about the regions, which allows to obtain real-time performance, even under out-of-plane rotation and large discontinuous motion. Finally, since the affine transformation is completely retrieved, tracking a single invariant region suffices for augmenting the scene with a virtual texture (in the same plane as the tracked region, and close to it such that perspective effects can be neglected).

In this section, we summarize two algorithms for extracting invariant regions – although for a more in depth discussion we refer to earlier work [23, 24]. We deal with two different region "types" (figure 1). The first one is based on a geometry-based approach, exploiting corners and edges in the image, and yields parallelogram-shaped regions, while the second one is purely intensity-based and yields elliptical regions (anchored on local intensity extrema). 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.

2.1. Geometry-based region extraction

The first method starts from a Harris corner point $\mathbf{h}$ and its nearby edges (extracted using the Canny edge detector). Although a general method for curved edges has been described in [23], we focus in this work on the special case of straight edges only. The two straight edges intersecting in the proximity of $\mathbf{h}$ fix one corner of the parallelogram-shaped region (call it $\mathbf{c}$) and the orientation of its sides. The remaining degrees of freedom (DOF) are lifted by examining some photometric quantities, evaluated over the

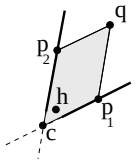

Figure 2. *Parallelogram-shaped region construction.*

parallelogram-shaped regions spanned by $\mathbf{c}$, a point moving along the first edge $\mathbf{p_1}$ and a point moving along the second edge $\mathbf{p_2}$ (figure 2).

The photometric quantities we use are:

$$f_1(\Omega) = abs(\frac{|\mathbf{p_1} - \mathbf{p_g} \quad \mathbf{p_2} - \mathbf{p_g}|}{|\mathbf{p} - \mathbf{p_1} \quad \mathbf{p} - \mathbf{p_2}|}) \frac{M^1_{00}}{\sqrt{M^2_{00}M^0_{00} - (M^1_{00})^2}}$$

$$f_2(\Omega) = abs(\frac{|\mathbf{p} - \mathbf{p_g} \quad \mathbf{q} - \mathbf{p_g}|}{|\mathbf{p} - \mathbf{p_1} \quad \mathbf{p} - \mathbf{p_2}|}) \frac{M^1_{00}}{\sqrt{M^2_{00}M^0_{00} - (M^1_{00})^2}}$$

with M^n_{pq} the *n*th order, *(p+q)*th degree moment computed over the neighborhood Ω, $\mathbf{p_g}$ the center of gravity of the region, weighted with image intensity $I(x, y)$ (one of the three color bands R, G or B), and $\mathbf{q}$ the corner of the parallelogram opposite to $\mathbf{c}$.

$$M^n_{pq} = \int_\Omega I^n(x, y)x^p y^q \; dxdy, \qquad \mathbf{p_g} = (\frac{M^1_{10}}{M^1_{00}}, \frac{M^1_{01}}{M^1_{00}})$$

Since these functions are invariant under the considered geometric and photometric transformations, a possible strategy could be to select the parallelogram-shaped region for which one of these functions reaches a (local) extremum. However, $f_1(\Omega)$ and $f_2(\Omega)$ do not show clear, well-defined extrema. Rather, we have some shallow *valleys* of low values (corresponding to cases where the center of gravity lies on or close to one of the diagonals). Instead of taking the inaccurate local extrema of one function, we combine the two functions and take the *intersections* of the two valleys, as shown in figure 3. The special case where the two valleys (almost) coincide must be detected and rejected, since the intersection will not be accurate in that case Although this method seems rather ad-hoc and not a very principled one, the regions we obtain prove to be much more stable than those based on a 2D local extremum.

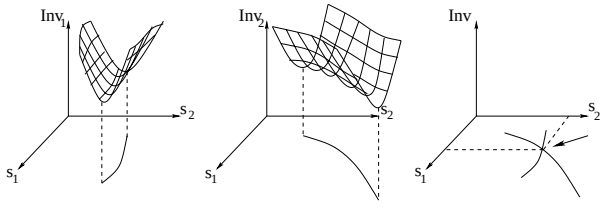

Figure 3. *Parallelogram-shaped regions: the intersection of the "valleys" of two different functions is used to determine the* $\mathbf{q}$ *point.*

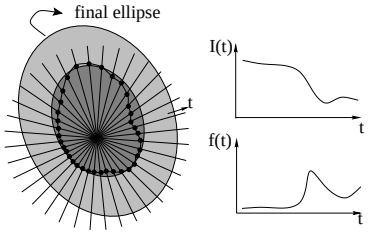

Figure 4. *An elliptical region is constructed based on the analysis of the intensity profile along rays emanating from* i

2.2. Intensity-based region extraction

The second region extraction method starts from local extrema in the image intensity, extracted with a non-maximum suppression algorithm. Given such a local extremum **i**, the intensity function along rays emanating from **i** is studied, as shown in figure 4. The following invariant function is evaluated along each ray:

$$f(t) = \frac{|I(t) - I_0|}{max(\frac{\int_0^t |I(t) - I_0| dt}{t}, d)}$$

with t the Euclidean arclength along the ray, $I(t)$ the intensity at position t, I_0 the intensity of the extremum **i** and d a small number which has been added to prevent a division by zero. The point for which $f(t)$ reaches an extremum is invariant under the aforementioned affine geometric and photometric transformations (given the ray). Typically, such extrema occur at positions where the intensity suddenly increases or decreases dramatically compared to the intensity changes encountered on the line up to that point.

Next, all points corresponding to maxima of $f(t)$ along rays originating from **i** are linked to enclose an (affinely invariant) region (figure 4). This often irregularly-shaped region is then replaced by an ellipse having the same shape moments up to the second order. This ellipse-fitting is affinely invariant as well. Finally, the area of the ellipse. is doubled. This leads to a higher distinctive power of the regions, due to a more diversified texture pattern within the regions and hence facilitates the recognition process, at the cost of a higher risk of non-planarity due to the less local character of the regions. Note that the ellipse is not centered on **i**. From now on, we will use **c** to refer to the center of elliptical regions, and to the edge intersection for parallelogram-shaped regions.

3. Scene recognition

Once local, invariant regions have been extracted, a scene can be recognized by recognizing the invariant regions it contains. By recognizing we mean determining

whether the current scene contents match a pre-processed model view of the scene. This is achieved by means of a nearest neighbor classification scheme, based on feature vectors containing moment invariants computed over the affinely invariant regions. As in the region extraction step, we consider invariance both under affine geometric changes and linear photometric changes, with different offsets and different scale factors for each of the three color bands.

Each region is characterized by a feature vector of moment invariants. The moments we use are *Generalized Color Moments* [22]. They contain powers of the image coordinates and of the intensities of the different color channels.

$$M_{pq}^{abc} = \iint_{\Omega} x^p y^q [R(x,y)]^a [G(x,y)]^b [B(x,y)]^c \, dxdy$$

with *order* $p+q$ and *degree* $a+b+c$. In fact, they implicitly characterize the shape, the intensity and the color distribution of the region pattern in a uniform manner. We use a feature vector containing 18 moment invariants, composed of moments up to the first order and second degree. As an additional invariant, we use the region type (the method that has been used for the region extraction). Only if the type of two regions corresponds, they can be matched.

Similarity is quantified using the Mahalanobis distance:

$$d_M(\mathbf{b}, \mathbf{a}) = \sqrt{(\mathbf{b}-\mathbf{a})^\top \mathbf{C}^{-1} (\mathbf{b}-\mathbf{a})}$$

where $\mathbf{a}$ and $\mathbf{b}$ represent two feature vectors, and $\mathbf{C}$ denotes the covariance matrix (estimated based on a set of tracking experiments).

After the invariant-based matching, a cross-correlation is performed as a final check. To this end, the region in the image R_i and the model region R_m from the model view are first aligned with one another by applying an affine transformation A. For the parallelogram-shaped regions, A is completely defined by putting into correspondence the corners of R_m with the corners of R_i. However, an ellipse having only five DOF, it is not possible to directly compute A from R_i and R_m. The missing DOF corresponds to a free rotation in the ellipse plane around its center. We propose here a new solution to this problem which is both more elegant and more time-efficient than our earlier approach [24] (consisting of an exhaustive search for the rotation maximizing the cross-correlation). The new solution is based on a photometric invariant version of the axis of inertia. First R_i is affinely mapped to a reference circular region O. The major and minor axes of inertia are then extracted (figure 5) as the lines passing through the center of O with orientations θ_{max}, θ_{min} defined by the solutions of:

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

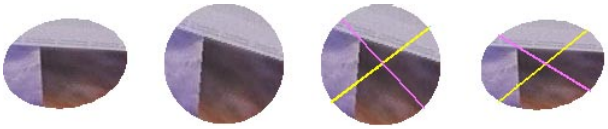

Figure 5. *Original elliptical region; mapped to circular region; axes of inertia in circular region; axes mapped back to elliptical region.*

with m_{pq} the $p+q$th order, first degree moment *centered on the region's geometric center*. Equation (1) 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 linear intensity changes. These axes are *invariant under rotation*, in the sense that they will cover the same part of the region after a rotation. Mapping the axes back to the original elliptical region provides 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; the cross-correlation test can follow and, if failed, the region match will be rejected. If a sufficient number of regions are recognized, the system considers the scene as a whole as recognized.

4. Region tracking

This section describes the new affine region tracker, which is the heart of the proposed AR system. 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 $\hat{\mathbf{a_i}} = A_{i-1}\mathbf{a_{i-1}}$ 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. This allows to continuously adapt the range of S_i so as to maximize the chances of finding R_i, while keeping it as small as possible (efficiency). The anchor points in S_i are extracted. These provide potentially better estimates for the region's location. We *investigate* the point closest to $\hat{\mathbf{a_i}}$ looking for the target region R_i. The anchor point investigation algorithm differs for geometry-based and intensity-based 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 $\hat{\mathbf{a_i}}$) to the farthest, until R_i is found (figure 6). Hence, the investigation process runs on a sparse subset of points in S_i. This

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

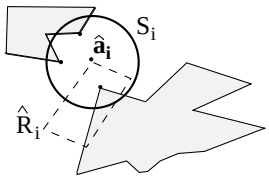

Figure 6. *Anchor points (thick dots) are extracted in the search space S_i, defined around the predicted anchor point* $\hat{\mathbf{a}}$.

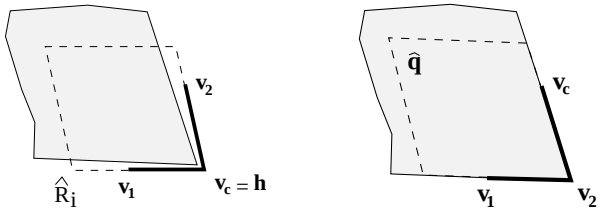

Figure 7. *Left: Polyline snake initialization. Right:* $\hat{\mathbf{q}}$ *initialization.*

way of *pruning* the search space is made possible by the region model we consider and allows to keep the radius of S_i wide enough to ensure tolerance to large image displacements, while staying within the tight time bounds imposed by real-time requirements.

In some cases it is possible that no correct R_i is found around any anchor point in S_i. This can be due to several reasons, including occlusion of the region, sudden acceleration (the anchor point of R_i is outside S_i) and 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 simple mechanism allows to recover the region in one of the next few frames, while avoiding the computationally expensive process of searching F_i further.

Our tracking scheme can be summarized as follows:

1. Predict the target region $\hat{R}_i = A_{i-1}R_{i-1}$ and its anchor point $\hat{\mathbf{a}}_\mathbf{i} = A_{i-1}\mathbf{a_{i-1}}$.
2. Define the search space S and extract anchor points.
3. Investigate the closest anchor point searching for R_i.
4. Loop to 3 for the next closest anchor point if needed.

4.1. Geometry-based 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 7). 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 [17, 25]. The algorithm has a higher probability of being attracted toward contours than the traditional snake implementation [21], and it is ensured to converge [17]. 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 7). 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, the objective function to be maximized is:

$$E_C(\mathbf{q}) = CrossCorr(A(\mathbf{q})R_{i-1},\ R_i(\mathbf{q})) \qquad (2)$$

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.

4.2. Intensity-based regions

Let us now focus on the tracking of intensity-based 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 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 (2) (as proposed in [18]). The search process could be initialized from the affine transformation A_{i-1} between the two previous frames. 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 section 2.2. In a second step, the extracted region R_i is submitted to the two following *geometric continuity* tests:

$$\frac{1}{\epsilon_e} < \frac{e(R_i)}{e(R_{i-1})} < \epsilon_e, \qquad \frac{1}{\epsilon_a} < \frac{a(R_i)}{a(R_{i-1})} < \epsilon_a \quad (3)$$

where $e(R)$, $a(R)$ are, respectively, the eccentricity and area of an elliptical region R and ϵ_e, ϵ_a are thresholds (typical values $\epsilon_e = 1.6, \epsilon_a = 1.6$). The two tests in (3) quickly detect the case where the extracted region can *not* be the correct one, and in practice reveal very effective in keeping the tracker away from the attraction of spurious elliptical regions located in the proximity of the correct one.

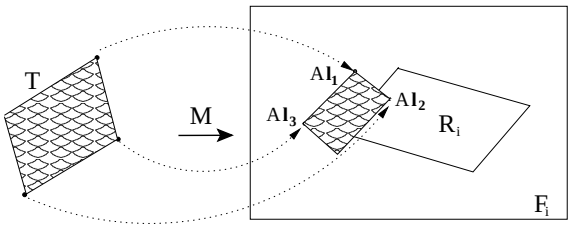

Figure 8. *Mapping the texture T to F_i.*

The geometric continuity tests represents a *necessary*, but not sufficient condition: a last step consisting of verifying that R_i cross-correlates with R_{i-1} above a predefined threshold (after aligning them using A_i) is still required. A_i is computed by putting in correspondence the center and the intersection points of the axes of inertia with the ellipse between regions R_{i-1} and R_i (section 2.2).

5. Augmenting a region

The previous section explained how to track a region through the image sequence. In this section, we move to the next goal: attaching a texture to the region, in every frame of the sequence. The texture will be projected in every frame on the same plane as R.

The user provides a parallelogram-shaped texture T and a set of three points $\Psi = \{\mathbf{l_1}, \mathbf{l_2}, \mathbf{l_3}\}$ indicating where it should be placed. These points have to be selected in a user-defined reference frame F_r only (e.g.: first frame) and define a parallelogram-shaped area $\Omega(\mathbf{l_1}, \mathbf{l_2}, \mathbf{l_3})$ where the texture should be mapped. This gives the user valuable creative freedom. Nevertheless Ψ should be chosen in the proximity of an affinely invariant region R, so as to maximize the quality of the results (the affine transformation of R may only hold in a limited neighborhood). Besides, it must not be forgotten that T will appear in the images as if it was on the same plane as R.

We restrict the explanation to a single frame F_i. The algorithm consists of two main steps: the first constructs the affine transformation M mapping the texture to F_i, the second does the actual texture-mapping. The algorithm is iteratively applied to every frame in the sequence. In order to construct M, we first compute A: the affine transformation mapping the region from the reference frame (R_r) to the current frame (R_i). The placement of the texture is obtained by warping $\{\mathbf{l_j}\}_{j=1..3}$ by A . By enforcing the correspondence between three texture corners and the obtained points $\{A\mathbf{l_j}\}_{j=1..3}$, we find the desired affine transformation M. It is now possible to continue with the second step: augment the region by mapping T in the image via M (figure 8):

$$W(\mathbf{p}) = T(M^{-1}\mathbf{p}), \quad \forall \mathbf{p} \in \Omega(M\mathbf{l_1}, M\mathbf{l_2}, M\mathbf{l_3})$$

where $W(\mathbf{p}), T(\mathbf{p})$ is the color of the augmented region, respectively of the texture, at pixel $\mathbf{p}$. Since the region's points are in correspondence along the frames, the texture will be projected always on the same physical surface.

Partially transparent textures are easily integrated in this approach: all pixels of T of a certain predefined color will not be projected on F_i. This simple process allows artificial textures of any shape, and with holes, which often blend much better in the real image. Animated textures are also supported by projecting on each frame a different texture from a user-given *texture sequence*.

6. Experimental results

We present four image sequences demonstrating the qualities of the proposed system. The images of the tracked planar patches are put into *complete correspondence* along the frames, allowing to define in every frame an affine reference coordinate system where to project the artificial texture (section 5).

The *Cinema* sequence (figure 9) shows the tracker robustness to discontinuous motion. A parallelogram-shaped region on a window of the background building is tracked and two advertising neon-light characters *AR* are added on the wall. Notice the texture is not projected on top of the region, but to the right of it, because this better fits in the scene. This is allowed by the mechanism explained in section 5. The *AR* characters change color, from red to blue every 20 frames, to simulate a flashing advertising neon-light. The sequence was taken by a fast moving hand-held camera and therefore contains considerable amounts of irregular motion: the region often bruskly changes direction and velocity, making it hard to predict the next location accurately. Moreover, the region moves fast: there are often large displacement between subsequent frames. As a result, the predicted location is often far from the correct one. The region was successfully tracked in every frame despite displacements of up to 22 pixels between subsequent frames and predictions that were off by up to 22 pixels from the target. The physical planar patch covered by the region is accurately tracked along the sequence, as proven by the constant very high cross-correlation score (average 0.97). This is very important for the quality of the final visual impact of the augmented scene, as it directly influences the accuracy by which the artificial texture is mapped into the real environment. As a result, the texture is perceived as strongly sticking on the building's wall and relative displacements are hardly noticeable. The computational performance [2] meets the real-time expectations: the sequence was processed in 22 ms per frame on average, and in no frames the tracker needed more than 34 ms (which corresponds to 30 Hz, the real-time bound). Since the time to map the texture was 9 ms per frame, the total average is still in the real-time limits.

Figure 9. *Cinema Scene. Top: tracked region and* AR *texture in frame 1; middle: texture in Frame 72; bottom left: cross-correlation score (y axe) in each frame (x axe); bottom right: velocity (dotted) and prediction errors (in pixels).*

The *Graffiti* sequence shows the combined application of scene recognition, region tracking and augmentation. The topmost image of figure 10 shows the *model view*. The second image is the first frame of the sequence to be augmented; the system successfully extracts and recognizes in this frame 8 regions of the model view and therefore knows this is the sequence it has to augment. The particular region to be augmented is found in the set of recognized regions (sunflower's eye). We see here the importance of elliptical regions: the Graffiti is drawn with very round lines and the wall contains several blob-like structures, but no strong corners or straight lines. The augmentation consists in the pictogram of a fish sprayed over the eye of the sunflower.

[2] All experiments performed on a Sun ULTRA10 workstation, with UltraSparc-IIi 440Mhz processor.

The scene was tracked at average 24 ms per frame, and the average cross-correlation between frames is 0.85.

Control of out-of-plane rotation is exemplified in the *Desk* sequence (figure 11). The goal is to attach the *BIWI* logo to the top-right corner of a poster. The challenge is posed by the region's motion, which contains significant rotation around the vertical axis, causing skew and anisotropic scaling effects in the image. The tracker was able to handle the situation by correctly transforming the 2D region's shape: despite the very different viewpoints of frames 1 and 168, the region is covering the same physical surface. This feature of the tracker allows to deform the virtual texture so as to reproduce the effects of viewpoint changes, and is therefore crucial for the quality of the resulting augmented scene. A frame of the sequence was tracked in on average 19 ms, with 0.97 average cross-correlation score.

The last sequence illustrates a potential application of the proposed AR system (figure 12). The goal is to visually help a worker, who is supposed to wear an head-mounted display, in the task of sticking a lightning metal plate on an electricity control panel. As the panel comes into the field of view, a glowing signal attracts the worker's attention to the area where the plate has to be stuck. As the worker approaches, the lightning plate appears and starts blinking exactly where it should be attached, looking like it was already stuck on the panel. As the worker moves, the superimposed plate moves and deforms with the viewpoint, in order to align with the image of the panel. The sign blinks faster as the worker gets closer, until it stops and it is continuously shown. The quality of the augmentation alignment is once again ensured by the accurate correspondences of the tracked region between frames (cross-correlation average 0.94). Each frame was tracked in average 24 ms, which, once added 8 ms for the texture mapping, fulfills the real-time promises.

In the current implementation we use a simple texture mapping algorithm carried out by the main processor. This allows to augment image areas of the size of the above examples (about 80x60 pixels) in about 8 ms. Since the tracker runs in average faster than real-time, this is still an acceptable overhead, and allows to perform the complete *track-augment* cycle in real-time. However, for much larger areas to be augmented, the texture-mapping overhead can become unacceptable. Fortunately cheaply available 3D accelerator chips have proven extreme texture mapping performances (e.g.: 3D games), providing an easy solution to this limitation.

Notice how the system needs only the tracked region to be on a planar surface and does not need a large, dominant physical plane to work.

Although the tracker succeeded in finding the target region in all frames of the presented sequences, failure is possible in some frames. Most common reasons include strong

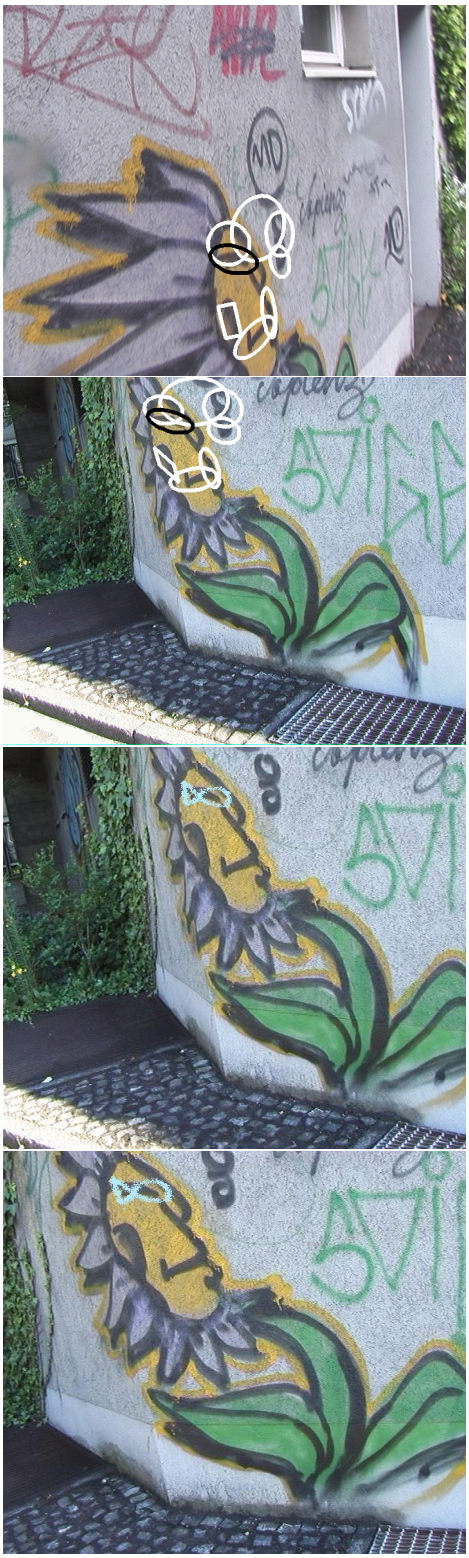

Figure 10. *Graffiti Scene. Top: Model view; second: first frame of sequence; third: Fish in first frame; bottom: Fish in frame 66.*

Figure 11. *Desk Scene. Top: region and BIWI logo in frame 1; other images: BIWI logo in frames 86 and 168.*

motion blur (disturbing the parallelogram-shaped regions sides tracker), failure of the anchor point extractor (especially corner extraction) and (partial) occlusion of the patch. In many such cases the mechanism described in section 4 allows to recover the region after a few frames.

The experiments confirmed that the system can accurately superimpose virtual textures to a user-selected planar part of a natural scene in real-time, under general motion conditions, without the need of markers or other artificial beacons. The sequences are available at `www.vision.ee.ethz.ch/ferrari/ISAR01`.

Figure 12. *Panel Scene. Top: region and signal in frame 1; second: signal in frame 24; third: lighting sign in frame 101; bottom: lightning in frame 202.*

7. Conclusion

In this contribution a novel method for augmenting planar patches in image sequences with (animated) textures was introduced. The most important feature of the system is the ability of working with natural patches of the scene instead of artificial landmarks. Moreover, it is able to automatically recognize the scene and to identify the patch to be augmented, in spite of changing viewpoint and/or illumination conditions. Once the patch has been identified, it is tracked and augmented in real-time. As shown in our experiments, the tracker is robust to large discontinuous motion, achieves real-time performance on a standard workstation, and it brings the image patches into complete correspondence, even in case of large out-of-plane rotations, as it recovers the affine deformations of the image patch as it evolves in the sequence. As a result, tracking a single patch suffices for rigidly connecting a planar virtual texture to the scene. Future developments include photometric changes in the virtual textures and extension to 3D virtual objects.

Acknowledgments

The authors gratefully acknowledge financial support from EC projects CIMWOS and VIBES.

References

[1] R. Azuma, A survey of augmented reality, Presence: Teleoperations and Virtual Environments, 6(4):355-385, 1997.

[2] Broll, W., Meier, E., Schardt, T. "The Virtual Round Table - A Collaborative Augmented Multi-User Environment", *3rd Int. Conf. on Collaborative Virtual Environments*, 2000.

[3] L-T Cheng, J A Robinson, "Dealing with Speed and Robustness Issues for Video-Based Registration on a Wearable Computing Platform", *Proc. 2nd Int. Symp. on Wearable Computers*, pp 84-91, 1998.

[4] J. Shi and C. Tomasi, "Good features to track", In *IEEE Conf. on Comp. Vis. and Pat. Rec.*, pp.593-600, 1994.

[5] K. Dorfmueller, "Robust tracking for augmented reality using retroreflective markers", *Computers & Graphics 23*, pp.795-800, 1999.

[6] W. Hoff, "Fusion of data from head-mounted and fixed sensors", *1st Int. Workshop on Augmented Reality*, 1998.

[7] M. Kanbara, H. Fuji, H. Takemura, and N. Yokoya, "A Stereo Vision-based Mixed Reality System with Natural Feature Point Tracking", *2nd Int. Symp. on Mixed Reality*, pp. 56-63, 2001.

[8] J. Kustaborder and R. Sharma, "Experimental evaluation of Augmented Reality for assembly training", *Int. Workshop on Augmented Reality*, 1999.

[9] T.Ohshima, K. Satoh, H. Yamamoto, and H. Tamura, "AR^{2}Hockey: A case study of collaborative augmented reality", *Proc. IEEE Virtual Reality Annual Int. Symp.*, pp.268-275, 1998.

[10] T. Okuma, K. Sakaue, H. Takemura, and N. Yokoya, "Real-time camera parameter estimation from images for a Mixed Reality system", *Proc. Int. Conf. Pat. Rec.*, 2000.

[11] Jun Rekimoto and Yuji Ayatsuka, "CyberCode: Designing Augmented Reality Environments with Visual Tags", *Designing Augmented Reality Environments*, 2000.

[12] M. Ribo, A. Pinz, and A. Fuhrmann, "A new optical tracking system for virtual and augmented reality applications", *IEEE Instrumentation and Measurement Technology Conf.*, 2001.

[13] G. Simon, A. Fitzgibbon, A. Zisserman, "Markerless tracking using planar structures in the scene", *Procs. of the Int. Symp. on Augmented Reality*, 2000.

[14] V. Sundareswaran, R. Behringer "Visual Servoing-based Augmented Reality", *Procs. Int. Workshop on Augmented Reality*, 1998.

[15] D. Tan, I. Poupyrev, M. Billinghurst, H. Kato, H. Regenbrecht, and N. Tetsuani, "The Best of Two Worlds: Merging Virtual and Real for Face-to-Face Collaboration", *Workshop on Shared Environments to Support Face-to-Face Collaboration*, 2000.

[16] M. Uenohara and T. Kanade "Vision-Based Object Registration For Real-Time Image Overlay" *Int. Journal of Comp. in Biology and Medicine*, 25(2):249-260, 1995.

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

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

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

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

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

[22] F. Mindru, T. Moons and L. Van Gool "Recognizing color patterns irrespective of viewpoint and illumination", *IEEE Conf. on Comp. Vis. and Pat. Rec.*, pp. 368-373, 1999.

[23] T. Tuytelaars and L. Van Gool "Content-Based Image Retrieval based on Local, Affinely invariant Regions" *Third Int. Conf. on Visual Information Systems*, pp. 493-500, 1999.

[24] T. Tuytelaars and L. Van Gool "Wide Baseline Stereo based on Local, Affinely invariant Regions" *British Machine Vision Conference*, pp. 412-422, 2000.

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

[26] U. Neumann and S. You "Natural Feature Tracking for Augmented Reality" *IEEE Transactions on Multimedia*, 1(1):53-64, 1999.