Shape Recognition and Pose Estimation for Mobile Augmented Reality

Nate Hagbi*, Oriel Bergig*, Jihad El-Sana*, and Mark Billinghurst†

*The Visual Media Lab, Ben-Gurion University, Israel

†The HIT Lab NZ, University of Canterbury, New Zealand

ABSTRACT

In this paper we present *Nestor*, a system for real-time recognition and camera pose estimation from planar shapes. The system allows shapes that carry contextual meanings for humans to be used as Augmented Reality (AR) tracking fiducials. The user can teach the system new shapes at runtime by showing them to the camera. The learned shapes are then maintained by the system in a shape library.

Nestor performs shape recognition by analyzing contour structures and generating projective invariant signatures from their concavities. The concavities are further used to extract features for pose estimation and tracking. Pose refinement is carried out by minimizing the reprojection error between sample points on each image contour and its library counterpart. Sample points are matched by evolving an active contour in real time. Our experiments show that the system provides stable and accurate registration, and runs at interactive frame rates on a Nokia N95 mobile phone.

KEYWORDS: In-Place Augmented Reality, handheld AR, shape recognition, geometric projective invariance, 3D pose estimation, vision-based tracking, free-hand sketching, shape dual perception.

INDEX TERMS: H.5.1 [Multimedia Information Systems]: Artificial, augmented, and virtual realities; I.4.0 [Image Processing and Computer Vision]: Scene Analysis – Tracking

1 INTRODUCTION

Model based visual tracking has become increasingly attractive in recent years in many domains, such as robotics and Augmented Reality (AR). In many of these domains visual tracking is often combined with object recognition tasks. In AR applications, model based recognition and 3D pose estimation are often used for superposing computer-generated images over views of the real world in real-time.

Fiducial based computer vision registration is popular in AR applications due to the simplicity and robustness it offers. Fiducials are of predefined shape, and commonly include a unique pattern for identification. Fiducials are useful for various tasks, such as prototyping and producing tangible interaction techniques for better user interfaces [1, 2]. On the other hand, Natural Feature Tracking (NFT) methods are becoming more common, as they are less obtrusive and provide a more natural experience [3]. This is achieved at the cost of increased computational complexity and decreased accuracy, since little is assumed about the environment to be tracked.

Figure 1. A hand-sketched shape contour and a printed shape recognized, tracked, and augmented.

In this paper we describe a recognition and pose estimation approach that is unobtrusive for various applications, and still maintains the high levels of accuracy and robustness offered by fiducial markers. We recognize and track shape contours by analyzing their structure. We use contour concavities to generate projective invariant signatures, which allow shape recognition across different viewpoints. The concavities are further used to extract shape features for real time pose estimation and tracking. A nonlinear optimizer is finally used for refining the calculated pose to the desired level of accuracy.

Shapes offer various benefits for AR. They lend themselves to identification and pose estimation in cases of partial occlusion and moderate projective distortion. Furthermore, they are flexible and unobtrusive to use in many AR applications where natural shapes carry contextual meanings, such as augmented books, catalogs, and printed advertisements. The proposed approach is also suitable for authoring in In-Place Augmented Reality (IPAR) applications [4].

We have implemented the proposed approach in *Nestor*, a system that operates in real-time on a mobile phone. The system can read shape files, or perform a learning step in which the user shows a new shape to the camera. The shape is analyzed and inserted into a library, which is used to maintain the set of shapes to be tracked and their properties, such as the models assigned to them. When a learned shape is recognized at runtime, its pose is estimated in each frame and augmentation can take place, as depicted in Figure 1. Our experiments show the system performs robust recognition and registration, maintains accurate tracking, and operates in interactive frame rates on a mobile phone.

* {natios, bergig, el-sana}@cs.bgu.ac.il

† mark.billinghurst@hitlabnz.org

Since our feature extraction method is based on concavities, the system is limited to recognizing non-convex shapes. To achieve robust results, at least two concavities are often required. In addition, the shape extraction step is based on thresholding, which implies the shapes used must have high contrast relative to their background.

The rest of this paper is structured as follows. In the next section we describe background and related work. Section 4 provides a brief derivation of the algorithmic approaches used by our system. Section 4 gives the details of our approach and describes the operation of our system. Section 5 addresses the context-based automation of the shape learning process. Section 7 describes our experiments with the system and the results. Section 7 discusses possible applications for the approach, and Section 8 concludes and outlines future work.

2 Related Work

Object recognition and pose estimation are two central tasks in computer vision and Augmented Reality. Object recognition methods aim to identify objects in images according to their known description. Model based pose estimation methods aim to determine the six degrees of freedom of known objects in a coordinate frame related to the camera's coordinate frame.

The cores of AR applications are often based on recognition and pose estimation to allow the appropriate virtual content to be registered and augmented onto the real world. Fiducial based registration methods have been used from the early days of AR, mainly due to their robustness to different conditions of the environment and computational simplicity [3]. Fiducials are commonly of predefined shape and size, and are usually integrated with an identification mechanism for recognizing them.

The first fiducials were based on points in predefined geometric configurations [5, 6]. Planar fiducials then became popular, offering superior accuracy and robustness to changing lighting conditions. For example, ARToolKit [7] locates a square frame in the image and calculates its pose. The frame is first used for rectification of the pattern inside of it. Pattern matching is then performed on the rectified pattern against a pattern library, which determines the 3D model that should be rendered. The calculated pose is then used to render the 3D model augmented on the square frame. Fiala developed the ARTag library [8], which uses digital coding theory to minimize false detection and inter-marker confusion rates. ARTag requires a relatively small marker size and avoids explicitly storing patterns for identification. Studierstube Tracker [9] is a lightweight tracking library designed to run on mobile platforms with low processing power and little memory. It uses one of several known algorithms for pose estimation and digitally encoded ids for fiducial identification.

The specific geometric configurations used in each of these and other fiducial tracking libraries make them computationally cheap and robust. Moreover, fiducials are natural to use in a variety of AR applications that augment specific objects, rather than the environment around the user. For example, in various AR applications users make use of tangible objects to interact with virtual content [1, 2]. Nevertheless, the obtrusive and monotonous appearance of predefined shape fiducials often renders them unattractive for use in AR applications, as they require the application developer to 'engineer the scene'.

Recognition of general planar shapes has been addressed in the research literature from various directions. One of the most elegant approaches to this problem is based on the concept of geometric projective invariance, originally pioneered in computer vision by Mundy, Zisserman, Rothwell, Forsyth, and others [10, 11]. Geometric invariants are properties of geometric configurations that remain unchanged under certain classes of transformations. As such, invariants form a powerful basis in computer vision for object description and recognition. They allow ignoring the current pose of an object relative to the camera and calculating descriptors for it directly from world observations. In this paper we use geometric invariant constructions to calculate projective invariant signatures for shapes, which allow recognizing them across different viewpoints.

Planar shapes have been used for tracking in several domains. They can be reliably tracked amongst clutter and inherently offer useful redundancy. Drummond and Cipolla [12] developed a vision-based robot guidance system based on the Lie algebra of the affine group. The camera, in that case, was mounted to the end of a robot arm, which was guided to a target position by integrating the local affine transformations of a contour being imaged. The contour is first shown to the system, which then begins to track it on a frame to frame basis. To compensate for the inability of integrated affine transformations to account for general projective transformations, the two non-affine warp parameters are finally estimated according to the centroid of the shape.

While the system in [12] is closed in a two dimensional loop that integrates affine transformations, Ruiz et al. [13] proposed a projective approach for estimating the 3D pose of shape contours. Rather than first estimating the affine transformation parameters and then the remaining non-affine parameters, they use an invariant based frame construction similar to that in [10] for extracting projective invariant features on the contour. These are used for constructing a linear system of equations in homogeneous coordinates that gives the camera pose. Although theoretically more accurate than the construction originally proposed in [10], the construction proposed in [13] limits the scope of usable shapes by several assumptions on shape concavities. In addition, no optimization step is performed once the transformation is calculated and no running times are reported.

Iterative optimization is useful for performing registration as well, or for refining a given pose estimate. Fitzgibbon [14] proposed a registration method for point sets based on the Levenberg-Marquardt nonlinear optimizer. As pointed out therein, direct nonlinear optimization on point sets can be easily extended to incorporate a robust estimator, such as Huber kernel, which leads to more robust registration. It also has a wider basin of convergence compared with traditional point set registration methods, such as Iterative Closest Point. This method can also account for curves as sets of points, although it makes no use of the connectivity information offered by them. In our approach we use an iterative optimization process to refine an initial pose estimate. It differs from the method proposed in [14] by the way correspondences are determined. We use the connectivity information of the contour to match correspondences by evolving an active contour between the library contour and image contour.

3 Theoretic Background

Nestor is based on several theoretic and algorithmic concepts. We give here a brief discussion of each for the unfamiliar reader. Readers who are familiar with the concepts are invited to skim this section to synchronize the notation, and proceed to the next section.

3.1 Recognition by Invariants

Out of the impressive literature written on object recognition, our recognition approach is most related to the inspiring work on projective shape invariance by Zisserman, Rothwell, Mundy, and Forsyth.

A function $I(\mathrm{P})$ of a geometric configuration P is a *scalar invariant* to a linear transformation of coordinates $x' = \mathrm{T}x$ if it holds that $I(x') = I(x)$. In this paper we use invariants to planar projective transformations, i.e., in which T is a 3×3 non-singular square matrix acting on homogeneous coordinates.

Similarly, relations between features of geometric configurations that are not affected by projective transformations are referred as *invariant relations*. For example, collinear sets of points are transformed to collinear sets of points under projective transformations, and hence collinearity is an invariant relation. Tangency is also preserved under projective transformations, which implies the projection of a line tangent to a curve is a line tangent to the projected curve. Invariant relations are useful for locating *distinguished features* of shapes. As proposed in [15], we use curve bitangent lines as distinguished features to calculate the homography a contour undergoes. This is achieved by a variation of the DLT algorithm described in Section 3.2

Since projective invariants of algebraic curves can be measured directly from their perspective projection, it is natural to characterize and recognize such arrangements by their invariant values. Invariants of such arrangements have also been used to characterize non-algebraic curves by fitting to them algebraic curves. This approach has been taken in [16] for affine invariance. However, since fitting based methods tend to be global, they are usually susceptible to occlusion of curve parts. Fitting algebraic curves to partial sets of observations has also been pointed out to be highly sensitive to noise [17], which introduces further error into the invariant measurement process.

Instead of measuring invariants directly from image observations, we first transform the image shape to its canonical representation, where every measurement is theoretically invariant [10]. This is done by extracting a set of distinguished features, and deriving the transformation that aligns them to a canonical frame. Applying this canonization transformation to the shape or its features yields the canonical representation of the shape with respect to the chosen frame. Care has to be taken regarding the spacing of the selected distinguished features, as the quality of the canonization transformation drops the closer the set of distinguished features is to degenerate, e.g., the closer the set of distinguished points is to collinear. In this paper we transform shape concavities to their canonical frames in order to calculate invariant shape signatures.

3.2 Direct Linear Transformation (DLT)

In this section we describe the Direct Linear Transformation algorithm, which is useful for estimating transformations from sets of corresponding measurements. We give here a derivation of the classic DLT algorithm for estimating the 2D homography between a pair of projective planes from a set of corresponding point pairs in the planes. Note that an equivalent derivation can be done based on lines instead of points, since a homography H that operates on points by $x' = \mathrm{H}x$ operates similarly on lines by $l' = \mathrm{H}^T l$. For further details, we refer the interested reader to Hartley and Zisserman [18].

We assume that for two corresponding points x' and x, the homography H is given by $x' = \mathrm{H}x$, up to multiplication by a scale factor. We have a set of n corresponding homogeneous points x_i and x_i' in $\mathbb{P}^2$, such that $x_i \leftrightarrow x_i'$, and we want to solve for H. We mark $x_i = (x_i, y_i, z_i)$, $x_i' = (x_i', y_i', z_i')$, and

$$\mathrm{H}_{3\times 3} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}.$$

A simple cross-product form that defines the operation of H on x_i and x_i', taking into account the scale factor, is

$$x_i' \times \mathrm{H}x_i = 0.$$

If we denote by h^j the j-th row of H, Equation X translates to

$$\begin{pmatrix} y_i'\mathrm{h}^3 x_i - z_i'\mathrm{h}^2 x_i \\ z_i'\mathrm{h}^1 x_i - x_i'\mathrm{h}^3 x_i \\ x_i'\mathrm{h}^2 x_i - y_i'\mathrm{h}^1 x_i \end{pmatrix} = 0.$$

This gives a set of three equations in the entries of H in the form of $\mathrm{A_i h} = 0$,

$$\begin{bmatrix} 0^{\mathrm{T}} & -z_i' x_i^T & y_i' x_i^T \\ z_i' x_i^T & 0^{\mathrm{T}} & -x_i' x_i^T \\ -y_i' x_i^T & x_i' x_i^T & 0^{\mathrm{T}} \end{bmatrix} \begin{pmatrix} \mathrm{h}^{1T} \\ \mathrm{h}^{2T} \\ \mathrm{h}^{3T} \end{pmatrix} = 0.$$

Since the third line is linearly dependent on the first two, each pair of corresponding points contributes two equations. Denoting by n the number of corresponding point pairs available, stacking these n sets of equation pairs we get a $2n \times 9$ system of equations. We get a unique solution for H in the case $n = 4$. In case $n > 4$, due to noise and measurement errors there will generally be no single solution. However, in such case it makes sense to find the best solution by least squares as the solution that minimizes the homography error over all points.

As discussed in [18], normalization of the features should be carried out before applying the DLT algorithm. Normalization makes the DLT algorithm invariant to similarity transformations of features, i.e., invariant to the coordinate system scale and origin. Normalization also meaningfully improves the accuracy of the estimated homography by reducing the effect of numerical errors. In our case, this process is applied to each shape separately. In practice, to normalize a set of points, we first rigidly translate the points so their centroid translates the origin. We then apply isotropic scaling that transforms the average distance of the points from the origin to $\sqrt{2}$.

3.3 Gauss-Newton Iteration

Once an initial pose has been estimated from the calculated homography, a Gauss-Newton iteration refines the estimated pose of a shape contour by minimizing the error vector resulting from fitting its points to the corresponding points of the image contour.

We want to have at x_i

$$\mathrm{H}x_i = x_i'.$$

The algebraic error for x_i under H is then

$$\epsilon_0 = \mathrm{H}x_i - x_i'.$$

We approximate the operation of H as locally linear and get

$$(\mathrm{H} + \Delta)x_i = \mathrm{H}x_i + \mathrm{J}\Delta,$$

where J is the Jacobian of $\mathrm{H}x_i$ according to H,

$$\mathrm{J} = \frac{\partial \mathrm{H}x_i}{\partial \mathrm{H}}.$$

We seek Δ that minimizes

$$(\mathrm{H} + \Delta)x_i - x_i' = \mathrm{H}x_i + \mathrm{J}\Delta - x_i' = \epsilon_0 + \mathrm{J}\Delta.$$

This linear minimization problem can be solved using Least-Squares minimization, or simply by using the pseudo-inverse of J, leading to

$$\Delta = -\mathrm{J}^{+}\epsilon_0.$$

One of the main challenges we need to face is accurately determining the correspondence between contour points, which allows calculating the error ϵ_0.

3.4 Active Contours

To establish the correspondence between the reprojected points of the library contour and the points of the image contour, we use an active contour model. Since introduced by Kass et al. [19], active contours have become a widely used tool for various tasks in computer vision, such as segmentation [20][21]. An active contour is a parametric function

$$\mathrm{c(s)} = \big(\mathrm{x(s)}, \mathrm{y(s)}\big) \in \mathrm{R}^2, \mathrm{s} \in [0,1],$$

defined in an image $\mathrm{I(x,y)}$. An active contour is assigned an energy term, minimizing which balances between several objectives. The energy of an active contour is commonly divided to *internal energy*, which refers to the geometric properties of the contour, and *external energy*, which refers to the properties of the image where the contour resides. A simple internal energy term that takes into account the stiffness and elasticity of the contour is

$$\mathrm{E_{int}(c(s))} = \alpha \left|\frac{\partial}{\partial \mathrm{s}}\mathrm{c(s)}\right|^2 + \beta \left|\frac{\partial^2}{\partial \mathrm{s}^2}\mathrm{c(s)}\right|^2.$$

A traditional external energy term that refers to the amount of contrast along the contour is based on the gradient of the smoothed image

$$\mathrm{E_{ext}(c(s))} = -\gamma|\nabla \mathrm{I(c(s))}|^2.$$

Minimizing according to

$$\mathrm{E(c(s))} = \mathrm{E_{int}(c(s))} + \mathrm{E_{ext}(c(s))}$$

leads the active contour towards strong edges in $\mathrm{I(x,y)}$ while keeping it relatively smooth and elastic.

4 NESTOR

Nestor is a recognition and 3D pose tracking system for planar shapes. The main goal of Nestor is to serve as a 3D registration solution for AR applications, which allows augmenting shapes with virtual content. Nestor can be used to augment shapes that have visual meanings to humans with 3D models having contextual correspondence to them, as depicted in Figure 1. We first give an outline of the system, and then proceed to describe each of its steps in detail.

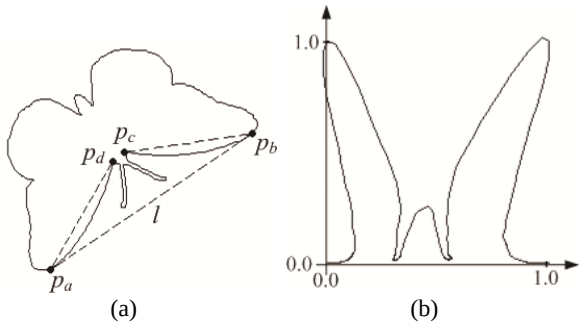


Figure 2. Canonical frame construction for a contour concavity. (a) Distinguished features on a contour concavity. (b) The canonical frame of the concavity.

Nestor operates by analyzing the live video feed provided by a handheld camera. Each frame goes through a series of filters that extract shapes which should be tracked. A projective invariant signature is then calculated for each of the contour concavities. Each such signature is used to generate a hypothesis for a single library shape. Features extracted from each concavity are then used to generate a first estimate for the homography between each hypothesized library shape and the image shape. The estimated homography is next used to reproject each hypothesized shape on the image contour to perform verification, which results in a single recognized shape. We then calculate an estimate of the homography between the image and library shape using features from all concavities. Finally, we refine the estimated transformation to the desired level of accuracy by nonlinear optimization on corresponding sample points along the reprojected library contour and the image contour. We determine the correspondence between these sample points by evolving an active contour from the unprojected image contour towards the library contour.

4.1 Shape Recognition

We begin shape recognition by applying adaptive thresholding to the image using integral images. The contour of each image shape is then extracted as a list of points $C^I = (p_1, p_2, \dots, p_n)$. We filter image contours in the beginning of the recognition process by removing contours with small area or length, as well as contours that are convex or close to convex.

We then proceed to construct frames that are preserved under projection for each extracted contour C^I. We use a construction similar to the one proposed in [10], which is based on the bitangent lines to the contour, illustrated in Figure 2(a). Each bitangent line l gives two tangency points, p_a and p_b, which segment a concavity from the rest of the curve, known as the *M-curve*. The interesting property of the bitangent line l and the points p_a and p_b is that their position relative to the curve remains unchanged under a change of viewpoint. We extract bitangent lines and their bitangency points by analyzing deviations from the convex hull of the contour.

Two additional points, p_c and p_d, for each concavity are extracted by casting from p_a and p_b lines tangent to the concavity. The extraction of these additional cast tangency points can be done while traversing the contour for bitangent lines. This process of extracting invariant feature points can be repeated recursively on nested concavities. These nested feature points can then be used for pose estimation in addition to the ones described above. However, the location accuracy of nested feature points on the contour drops with the size of the concavity.

(a) (b)

Figure 3. Contour point correspondence matching. (a) Library contour reprojected using the homography estimate. (b) Correspondence matching using the distance transform as external energy, and stiffness and elasticity internal energy terms. Corresponding sample points are marked by connecting lines.

The four extracted points p_a, p_b, p_c, and p_d, along with the bitangent line l and the cast tangent lines, are referred as the *distinguished features* of the concavity. The four distinguished points form a projective invariant frame for the concavity. The projective transformation that maps these points to the four corners of the unit square gives the canonical representation of the concavity. This transformation is calculated and then applied to all of the concavity points, yielding the canonical representation of the concavity, as depicted in Figure 2(b) for the lower right concavity of the contour in Figure 2(a). The selection of the unit square is arbitrary and different selections are possible as well.

We proceed to calculate a signature for each concavity of C^I from its canonical representation. We use a construction similar to shape *footprints*, originally proposed by Lamdan et al. [22]. Our signature is based on the areas bounded between the transformed concavity curve and a set of rays $\{r_i\}$ cast in constant polar intervals from a point p_{base} in the basis of the concavity canonical frame, midway between the transformed p_a and p_b. The signature coordinate values are set to be the bounded areas normalized by the total area bounded by the concavity in the canonical frame and the x-axis. Let us denote by r_i the i^{th} ray cast from p_{base}, and by a_i the area bounded by the concavity, r_i, and r_{i+1}. The signature is then

$$s = < s_1, s_2, \ldots, s_m >,$$

where $s_j = a_j / \sum_{i=1}^{m-1} a_i$ and m is the number of polar intervals. The nearest neighbor of s is then found in the shape library using a hash map, and a hypothesis is generated for it.

The final step in the recognition process is hypothesis verification, where each hypothesized library shape is reprojected and tested against the image shape C^I. The homography used for reprojection is calculated according to the same features used for constructing the canonical frames. The hypothesized library shape C^L with minimal reprojection error to C^I is selected, where we use the fraction of area common to the image shape C^I and the reprojected library shape as the error metric. Next, we turn to describe the calculation of the pose of the image shape using the recognized library shape C^L.

4.2 Pose Estimation

We begin pose estimation by calculating a first estimate of the homography H between the recognized library shape C^L and the corresponding image shape C^I. For this purpose, we use the Direct Linear Transformation (DLT) algorithm [18] on the distinguished features of C^I and C^L.

The DLT algorithm provides a good initial homography estimate. Figure 3(a) depicts the reprojection of a library shape on the image shape using the estimated homography. We next use a

Figure 4. The effect of partial occlusion on the extracted contour, marked in blue.

Gauss-Newton iteration to minimize the error between the reprojected contour and the image contour. The error we minimize is a function of the Euclidean distance between corresponding sample points on both contours. We use a calibrated camera and minimize the error of the camera external parameters, as described in [3].

To measure the error vector and perform the minimization, we first determine a point correspondence between C^I and C^L. In [12] it has been proposed to cast rays from sample points on the reprojected contour in the normal direction and find the intersections of the rays with the image contour. The intersection points can then be used as corresponding to the origins of the rays, and the error they exhibit can be calculated, for example, as their Euclidean distance.

Here we take on a different strategy, based on evolving an active contour from the reprojected library contour towards the image contour. To guide the active contour, we use an external energy term that depends on the distance transform of the image contour. Evolving solely according to this term has a similar effect to the ray casting operation, since each point evolves according to the gradient of the distance transform in that point, which is the normal to the reprojected curve. We add internal terms to preserve the stiffness and elasticity of the contour, and get the following energy functional

$$\mathrm{E}(\mathrm{c}(\mathrm{s})) = \alpha|\mathrm{DT}(\mathrm{I}(\mathrm{C_I}))|^2 + \beta\left|\frac{\partial}{\partial s}\mathrm{c}(\mathrm{s})\right|^2 + \gamma\left|\frac{\partial^2}{\partial s^2}\mathrm{c}(\mathrm{s})\right|^2,$$

where c(s) denotes the reprojected library contour, I(·) denotes the binary image of a contour, DT(·) denotes the distance transform of a binary image, and α, β, and γ are weighting coefficients. An evolved contour point ceases to move when the distance it has traveled in a single iteration is smaller than a predefined threshold, or when a predefined number of iterations has been reached. The former condition helps preventing the active contour points from moving along the contour. The resulting correspondence is depicted in Figure 3(b), where corresponding points are connected by line segments.

Our experiments show that the internal energy terms make the active contour less susceptible to image noise, which can corrupt the normals at contour points, and maintain the structure of the contour. The proposed method can also take into account additional useful constraints. For example, it can be extended to integrate the known prior model of the contour, as proposed in [23], as well as other structural constraints.

The nice properties of this correspondence matching strategy do not come for free. Calculating the distance transform of an image is a relatively expensive task, even when using only fixed point operations. However, it is only necessary to calculate the distance transform in a narrow band around the reprojected contour C^L.

Masking the distance transform calculation to a narrow band meaningfully reduces the per-frame processing time.

An even more effective scheme can be used to completely avoid the distance transform calculation in each frame. Correspondence information can be calculated by unprojecting C^I using the inverse homography H^{-1}, and then evolving it towards C^L, rather than the other way around. In this way, the distance transform image of C^L is calculated and stored once when it is learned by the system. Any subsequent image contours identified as matching C^L will then use this distance transform image for correspondence matching. Using this scheme the calculation of the distance transform image in each frame is avoided.

4.3 Recursive Tracking

A simple strategy can be used for tracking shapes on a frame to frame basis, which allows skipping the recognition step for most shapes in most frames. This is achieved by maintaining for each shape a set of simple properties that can be efficiently and robustly tested to see if the shape corresponds to any shape in the previous frame. For each image shape, we calculate its centroid, length, and area. In each frame we attempt to find the corresponding shape from the previous constant number of frames according to these properties.

In fact, we can skip the DLT step as well by using a motion model to get a first estimate of the contour in this frame, or simply by using the transformation calculated for C^L in the last frame as a first estimate. The minimization process can then carry on as described above. In the case of jerky movement, the active contour can lose track of the image contour. In this case we reapply the DLT algorithm from scratch.

To filter noise in the final estimated homography, we use Double Exponential Smoothing (DESP) [24]. This is useful in cases of severe noise or bad lighting. Using DESP gives results that are comparable to more powerful methods, such as Kalman filtering, in a small fraction of the processing time [25].

4.4 Partial Occlusion

Partially occluding C^I effectively changes the contour extracted from the image. Depending on the brightness of the occluding object, C^I may be extended or shrunk, with only part of it faithful to the corresponding library shape C^L. Figure 4 depicts a shape partially occluded by a hand and the extracted contour in blue. The occluded part may contain concavities that point to different library shapes. However, the recognition method described above deals with partially occluded shapes in a straightforward manner as long as enough faithful concavities remain visible. It thus remains to deal with partial occlusion through the pose estimation process.

Since we assume C^I has been recognized, its visible features still provide us with a first estimate for the pose of C^I. Even if there are not enough visible features for pose estimation from scratch, the pose of the shape can be assumed to be the one from the last frame, or extrapolated according to a motion model.

The main challenge is thus to reduce the effect of unfaithful parts of C^I in the error minimization process. We can achieve this by treating the points of these contour parts as outliers that should be identified and ignored through the minimization process. This can be achieved using RANSAC [26], which detects and filters outliers by randomizing a basis for estimating the homography between the contours and then checks the fraction of remaining points that conform to this estimate. Effectively, it is sufficient to check this conformance only for a small portion of the contour points sampled uniformly.

Figure 5. Shape class representatives used to automatically assign virtual content to new shapes.

Finally, since we are tracking recursively, a shape can be tracked from previous frames even when only a small and unrecognizable portion of it remains visible. This is achieved by continuing to apply the minimization process in Section 4.2 to the visible part of the contour.

4.5 Shape Library

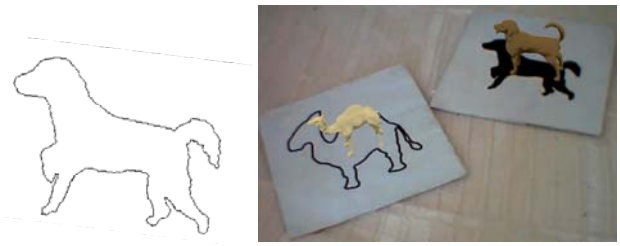
The system maintains a shape library that contains the shapes learned so far. The system can load a directory of shape files and learn them. The user can also teach the system new shapes at runtime, for example by sketching. To teach the system a new shape, the user presents it to the camera in a frontal view. The system then analyzes the shape contour and adds it to the shape library. The user can now attach a virtual model to be augmented to the new shape and modify its different properties, such as scale and rotation.

When teaching the system a new shape, the image goes through the same recognition step described in the Shape Recognition Section, and its signatures are hashed. The curve, its signatures, and additional required information are stored in the shape library. These are later used in the recognition step in real-time. Since we use shape contours, the system works for curves and solid blobs in the same manner. For curves, double edges can be detected and inner edges ignored in order to perform recognition and tracking according to outer edges.

Our shape library is based on three lists of shapes, which provide two caching levels to avoid searching through the entire shape library in every frame. The *shape list* contains all the shapes known to the system. The *visible shape list* points to shapes that are visible in the current image. The shapes in this list are tracked and updated in each frame. In addition, we maintain an *execution shape list*, which contains the shapes that have been recognized in the current execution. When a shape becomes invisible, it is moved from the visible shape list to the execution shape list. When a shape becomes visible, it is first searched for in the execution shape list. In case it is not found, it is searched for in the shape list. Once the shape is found, it is moved into the visible shape list. This way, the shape library is searched for each shape once per execution, when the shape first appears. This strategy can be useful when only a few shapes are visible in a single frame, and when only a small number of shapes are used through a single execution.

5 Contextual Shape Learning

So far, to teach the system a new shape, the user has to explicitly 1) show it frontally to the camera and 2) assign a model to it. In this section we address the automation of this shape learning process. The former step can be integrated into the application usage by automatically rectifying a new shape according to the plane it lies in, which is inferred from a previously learned shape that is coplanar with the new shape. The latter step can be performed automatically by classifying the new shape to one of the shape classes and suggesting contextual model assignments to the user.



(a) (b)

Figure 6. Automatic rectification and model assignment (a) A new sketched dog shape automatically rectified according to a previously learned camel shape (b) The new shape is automatically assigned a 3D model by classification.

5.1 Automatic Shape Rectification

To learn an unknown shape appearing in the image, for example upon user request, we automatically perform rectification according to the rectifying transformation recovered from a tracked known shape that lies in the same plane. This allows the development of sketching applications where the user draws different shapes on the same page according to the applicative visual language rules and context.

Upon user request to learn a new shape in the image, e.g., by clicking it with a mouse, the nearest tracked shape NC to the new shape C is found according to the shapes' centroids. To project C to the image plane, we apply to C the rectifying transformation of NC, which is the inverse of the projection homography of NC, H_{NC}^{-1}. This projects C to the image plane outside of the image bounds and to a scale that depends on its location relative to C in the world. We apply a similarity transformation to centralize the rectified contour of C and normalize its scale.

Figure 6(a) shows the result of this automatic rectification process. This step allows Nestor to learn the new dog shape in Figure 6(b) according to the previously learned camel shape. The dog model that is augmented on top was automatically assigned to the shape by the system according to a shape class library that links between the general shape class of dogs and its 3D model. This automatic assignment is explained next.

5.2 Contextual Model Assignment

According to the applicative context, the class of a newly learned shape can be determined, giving rise to the assignment of the appropriate virtual content to it. For example, when the user sketches a flower, Nestor consults the shape class library and finds the class the flower sketch belongs to. Nestor then automatically assigns a model to the newly learned shape according to a shape class library that correlates between shape classes and their meaning, i.e., their virtual content. This is although the system has not yet seen the specific shape being learned.

To classify a new shape to one of the shape classes, we measure the similarity of the new shape to the representative shapes of the shape classes. Figure 5 depicts some of the class representative shapes used by Nestor. We calculate a Shape Context descriptor [27] for each of the class representative shapes in advance. When a new is learned, we calculate its descriptor and find the class library shape that is most similar. We assume the deformation between the shape the user sketches and its corresponding class representative is an articulated deformation that maintains local similarity and topology. This motivates the use of the Inner-Distance as a metric for calculating the Shape Context descriptors, as proposed in [28].

Depending on the application, the meaning given to new shapes can depend on nearby shapes. The scope of shapes to recognize from is then meaningfully reduced, increasing recognition success rate. This allows dealing with much larger shape class libraries, since the scope of possible shapes in each step is narrower and context-dependant.

6 Experimental Results

We benchmarked and tested Nestor on a Nokia N95 mobile phone and a Dell Latitude D630 notebook computer. The Nokia N95 is equipped with a 330MHz processor and a camera that captures 320×240 pixel images. The Dell notebook is equipped with a 2.19GHz processor and a Logitech QuickCam webcam that provides 640×480 pixel images.

We measured the relation between the number of tracked shapes in each frame and tracking time. Figure 8 shows the average tracking time in the PC and Nokia N95 configurations for different numbers of shapes being tracked in each frame. For testing the system we used solid blobs, each containing 2 to 8 concavities. As can be seen, as the number of shapes that are tracked increases, so does the tracking time (from 28.3ms on an N95 phone for one shape to 40.4ms for six shapes). The library consisted of 100 shapes taken from the MPEG-7 shape dataset.

Recognition performance is related to the number of shapes in the shape library and the slant of the viewed shape. To assess this relation we measured the recognition rate of the system with different shape library sizes and slants. Figure 9 shows the recognition rate as a function of the number of shapes in the library for different slants. The experiment was performed using the notebook configuration described above, with the camera fixed approximately 40cm from the shapes. The system was trained with solid blob shapes from the MPEG-7 shape dataset, each containing 2 to 8 concavities. For each library size, the recognition rate was tested on all of the shapes in the library. As the slant angle increases, the amount of information contained in the image of each concavity drops, which reduces the recognition rate.

(a) (b) (c)

Figure 7. Shape augmentation in different contexts.

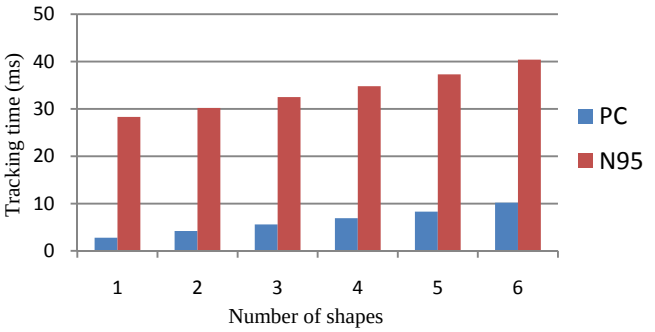


Figure 8. Average tracking time per frame in milliseconds on a Dell Latitude notebook and a Nokia N95 as a function of the number of shapes tracked in each frame.

We also measured the reprojection error for different distances of the camera from imaged shapes. Figure 10 shows the average reprojection error over ten different shapes in a distance ranging between 20cm and 100cm. For most shapes the average reprojection error was smaller than one pixel when positioned 1m away from the camera. As can be seen, Nestor has about half the error of ARToolKit.

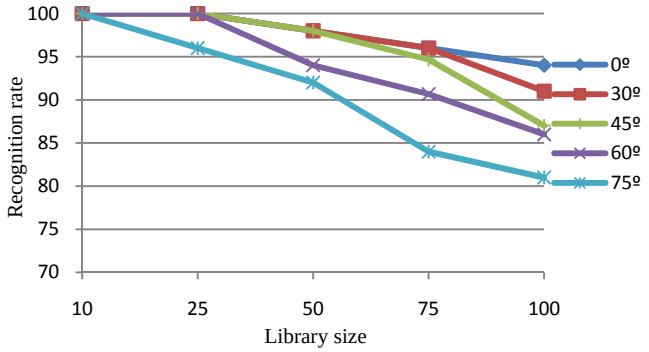


Figure 9. Recognition rate as a function of the number of shapes in the shape library. Different slant degrees are depicted by lines of different colors.

Since our recognition approach is based on concavity features and pose is refined iteratively using an active contour, it was interesting to test the system on shapes which are close to planar. We found that in most cases such shapes are successfully recognized, and that the estimated pose in such cases is subject to small and stable offset error. Figure 7(b) shows a logo of The HIT Lab NZ printed on the side of an oval shaped mug. Although the mug side on which the logo is printed is not entirely flat, the shape is recognized and stable augmentation is performed.

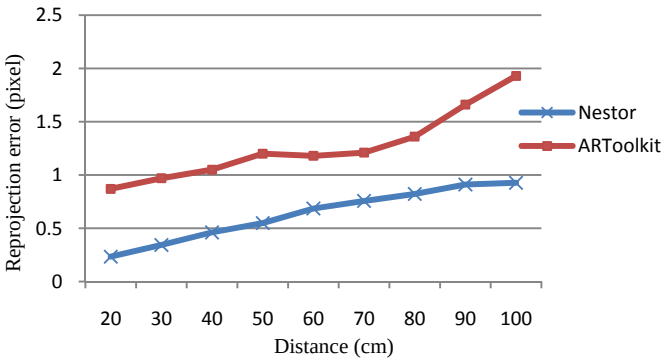


Figure 10. Average reprojection error in pixels as a function of the distance of the camera from an imaged shape in centimeters.

7 APPLICATIONS

Augmented Reality applications that make use of fiducial based registration can be found nowadays in various domains. In most cases the fiducials are of predefined shape, such as square frame. For most of these applications, augmentation of planar shapes of non-predefined and non-uniform structure can be useful. Dual perception is another benefit of the method, where the shape conveys an idea in a given context even without using a device, yet the experience can be further enhanced by virtual content when using the AR application. Here we give examples for existing applications that can use natural shape registration, as well as new applications.

AR advertising is a fast growing domain, in which the most difficult part is sparking interest in people before the AR experience actually begins. From our experience, square fiducials are often too monotonous for users to make any effort and view them using an AR device. In this context, shapes can be designed to be attractive from the first glance.

Augmenting books has been proposed in different domains as well, such as education and guidance. Books often contain object figures contrasted from their background, which can be augmented using the proposed approach, e.g., toddler animal books and flower handbooks. The shape library, in this case, can be downloaded when starting to browse the book. Figure 7(c) depicts an example of a motorbike catalog page. The outline of the motorbike photo is used for augmenting a 3D model of a motorbike on top of the book page.

Figure 11. Augmentation of a teddy bear sketch on a hand palm.

The proposed approach is also useful for developing AR applications based on logos, which can be found on different kinds of media. A business card could point to a 3D model that downloads in real-time and appears on top. Figure 7(b) demonstrates augmentation of The HIT Lab NZ logo printed on the side of a mug.

Another interesting direction is enabling the development of applications that combine sketching and AR. Sketching can be used for authoring AR content, for interacting with existing virtual content, or simply for creating AR fiducials on-the-fly. Nestor allows tracking from hand sketches, to which 3D virtual content can be assigned according to the application. Figure 11 shows the augmentation of a teddy bear sketched on a hand palm. A virtual teddy bear model that was assigned to the sketch appears on top.

8 Conclusion and Future Work

We have described Nestor, a recognition and pose estimation system for planar shapes. The system operates in interactive frame rates on a Nokia N95 mobile phone. It performs robust recognition of shapes and maintains accurate and stable 3D registration.

Due to the redundancy of shape contours, it is natural to extend the system to deal with partial occlusion. One of the points that require attention in this case is the optimization process, in which outlying correspondences resulting from occlusion must be identified and ignored. This can be achieved, for example, by introducing a robust estimator in the optimization process.

To support large shape libraries, we intend to introduce a linear classifier on our raw signatures. A training step will be used to teach the classifier the different shapes it should recognize. The signature space will then be transformed to a new basis that allows distinguishing between signatures more robustly.

Nestor allows planar shapes to be used for registration as flexible fiducials for AR. This allows the development of sketch-based applications that do not require any additional means for registration, such as predefined shape fiducials.

Acknowledgments

This work was supported by the Lynn and William Frankel Center for Computer Sciences.

References

[1] Kato, H., Billinghurst, M., Pouipyrev, I., Imamoto, K., and Tachibana, K., Virtual Object Manipulation on a Table-Top AR Environment, *International Symposium on Augmented Reality*, pp. 111-119, 2000.

[2] Lee, G.A., Nelles, C., Billinghurst, M., and Kim, G.J., Immersive Authoring of Tangible Augmented Reality Applications. *Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality*, pp. 172-181, 2004.

[3] Lepetit, V. and Fua, P., Monocular Model-Based 3D Tracking of Rigid Objects: A Survey. *Foundations and Trends in Computer Graphics and Vision*, pp. 1-89, 2005.

[4] Hagbi, N., Bergig, O., El-Sana, J., Kedem, K., and Billinghurst, M., In-Place Augmented Reality, *International Symposium on Mixed and Augmented Reality 2008*, pp. 135-138,

[5] Hoff, W.A., Nguyen, K., and Lyon, T., Computer vision-based registration techniques for augmented reality, *Proceedings of Intelligent Robots and Control Systems XV, Intelligent Control Systems and Advanced Manufacturing*, pp. 538–548, 1996.

[6] State, A., Hirota, G., Chen, D., Garett, W., and Livingston, M., Superior augmented reality registration by integrating landmark tracking and magnetic tracking, *Computer Graphics, SIGGRAPH Proceedings*, pp. 429–438, 1996.

[7] Kato, H. and Billinghurst, M., Marker Tracking and HMD Calibration for a video-based Augmented Reality Conferencing System, *2nd International Workshop on Augmented Reality*, 1999.

[8] Fiala, M., ARTag, An Improved Marker System Based on ARToolkit, NRC Institute for Information Technology, NRC 47166/ERB-1111, 2004.

[9] Schmalstieg, D. and Wagner, D., Experiences with Handheld Augmented Reality, *The Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2007.

[10] Rothwell, C.A., Zisserman, A., Forsyth, D.A., and Mundy, J.L., Canonical Frames for Planar Object Recognition, *Proceedings of the Second European Conference on Computer Vision*, pp. 757-772, 1992.

[11] Zisserman, A., Forsyth, D., Mundy, J., Rothwell, C., Liu, J., and Pillow, N., 3D object recognition using invariance. *Artificial Intelligence*, pp. 239-288, 1995.

[12] Drummond, T. and Cipolla, R., Visual tracking and control using Lie algebras, *Computer Vision and Pattern Recognition*, pp. 652-657, 1999.

[13] Alberto Ruiz, Pedro E. López de Teruel and Lorenzo Fernández., Robust Homography Estimation from Planar Contours Based on Convexity, *European Conference on Computer Vision*, pp. 107-120, 2006.

[14] Andrew W. Fitzgibbon., Robust registration of 2D and 3D point sets, *In Proc. British Machine Vision Conference, volume II*, pp. 411-420, 2001.

[15] Rothwell, C.A., Zisserman, A., Forsyth, D., and Mundy, J., Planar Object Recognition using Projective Shape Representation. *International Journal of Computer Vision*, pp. 57-99, 1995.

[16] Carlsson, S., Projectively Invariant Decomposition and Recognition of Planar Shapes. *International Journal of Computer Vision*, pp. 193 - 209,

[17] Fitzgibbon, A.W., Pilu, M., and Fisher, R.B., Direct least-squares fitting of ellipses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 476-480, 1999.

[18] Hartley, R. and Zisserman, A., Multiple view geometry in computer vision. 2003.

[19] Kass, M., Witkin, A., and Terzopoulos, D., Snakes: Active contour models. *International Journal of Computer Vision*, pp. 321-331, 1988.

[20] Chan, T.F. and Vese, L.A., Active Contours without Edges. *IEEE Transactions on Image Processing*, pp. 266-277, 2001.

[21] Caselles, V., Kimmel, R., and Sapiro, G., Geodesic Active Contours. *International Journal of Computer Vision*, pp. 61-79, 1997.

[22] Lamdan, Y., Schwartz, J.T., and Wolfson, H.J., Object Recognition by Affine Invariant Matching, *Computer Vision and Pattern Recognition.*, pp. 335-344, 1988.

[23] Riklin, T.R., Sochen, N., and Kiryati, N., Mutual segmentation with level-sets. *Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop*, 2006.

[24] LaViola, J.J., Jr., Double exponential smoothing: an alternative to Kalman filter-based predictive tracking, *Proceedings of the workshop on Virtual environments*, pp. 199-206, 2003.

[25] LaViola, J.J., Jr., An experiment comparing double exponential smoothing and Kalman filter-based predictive tracking algorithms, *Proc. IEEE Virtual Reality*, pp. 283-284, 2003.

[26] Martin, A.F. and Robert, C.B., Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, pp. 381–395, 1981.

[27] Belongie, S. and Malik, J., Matching with Shape Contexts, *IEEE Workshop on Contentbased Access of Image and Video Libraries*, pp. 20, 2000.

[28] Ling, H. and Jacobs, D.W., Shape Classification Using the Inner-Distance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, pp. 286-299, 2007.