International Journal of Computer Vision

DOI: 10.1007/s11263-006-0017-9

Fast Non-Rigid Surface Detection, Registration and Realistic Augmentation

JULIEN PILET*, VINCENT LEPETIT AND PASCAL FUA
Computer Vision Laboratory, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland
Julien.Pilet@epfl.ch
Vincent.Lepetit@epfl.ch
Pascal.Fua@epfl.ch

Received March 21, 2006; Accepted September 5, 2006

Abstract. We present a real-time method for detecting deformable surfaces, with no need whatsoever for a priori pose knowledge.

Our method starts from a set of wide baseline point matches between an undeformed image of the object and the image in which it is to be detected. The matches are used not only to detect but also to compute a precise mapping from one to the other. The algorithm is robust to large deformations, lighting changes, motion blur, and occlusions. It runs at 10 frames per second on a 2.8 GHz PC. We demonstrate its applicability by using it to realistically modify the texture of a deforming surface and to handle complex illumination effects.

Combining deformable meshes with a well designed robust estimator is key to dealing with the large number of parameters involved in modeling deformable surfaces and rejecting erroneous matches for error rates of more than 90%, which is considerably more than what is required in practice.

Keywords: non-rigid detection, non-rigid augmented reality, real-time deformable registration

1. Introduction

Rigid object detection and tracking have been extensively studied and effective, robust, and real-time solutions proposed (Lepetit and Fua, 2005; Lepetit et al., 2005; Lowe, 2004; Rosten and Drummond, 2005). The two are of course complementary since trackers require initialization and, no matter how good they may be, will sometimes lose track, for example, because of severe occlusions. Non-rigid object tracking has also been convincingly demonstrated, for example in the case of animated faces (Baker et al., 2004; Cootes et al., 2001; DeCarlo and Metaxas, 1998) or even more generic and deformable objects (Bartoli and Zisserman, 2004; Sclaroff and Isidoro, 2003). However, the automated detection of such deformable objects still lags behind and existing methods (Belongie et al., 2002; Ferrari et al., 2004) are far less convincing for real-time applications. They tend to be computationally intensive and are usually geared more towards recognition or segmentation than providing the kind of fast initialization that a tracker needs to recover from potential failures.

In this paper, we propose a method that fits this requirement by allowing fast and robust detection and registration of an object that can be subjected to very large non-affine deformations such as those of Figs. 1 and 2. It relies on wide-baseline matching of 2-D feature points, which makes it resistant to partial occlusions and cluttered backgrounds: Even if some features are missing, the object can still be detected as long as enough are found and matched. Spurious matches are removed by enforcing smoothness constraints on the deformation, which is done very quickly in our approach. We then demonstrate its applicability by using it to realistically modify the texture of a deforming surface and to handle complex illumination effects.

More specifically, at the heart of our approach is a very fast wide-baseline point matching technique that allows us to establish correspondences between keypoints extracted from a training image of the undeformed

*Correspondence author.

Figure 1. In order to achieve surface detection, we use a model image (a). Then, our method computes a function mapping the model to an input image (b). To illustrate this mapping, we find the contours of the model using a simple gradient operator and we use them as a validation texture (c) which is overlaid on the input image using the recovered transformation (d). Additional results are obtained in different conditions (e to i). Note that in all cases, including the one where the T-shirt is replaced by a cup (j), the white outlines project almost exactly at the right place, thus indicating a correct registration and shape estimation. The registration process, including image acquisition, takes about 100 ms and does not require any initialization or *a priori* pose information.

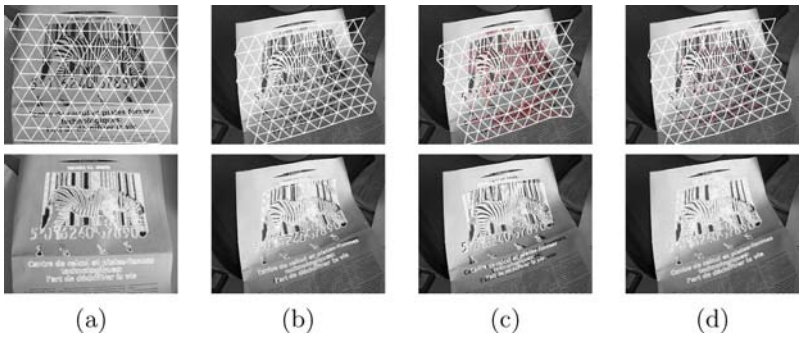


Figure 2. Comparing three different keypoint matching algorithms. (a) Model image and validation texture shown in white. Results using: (b) Real-time classification trees, (c) shape context descriptor reimplementation, and (d) SIFT.

object to those that can be found when the object deforms (Lepetit et al., 2005). Given such correspondences, if the target object were rigid, detecting it and estimating its pose could be implemented using a robust estimator such as RANSAC (Fischler and Bolles, 1981). However, for a deformable object, the problem becomes far more complex because not only pose but also a large number of deformation parameters must be estimated.

This paper's main contribution is a robust optimization scheme designed to work in high dimensional spaces with data very polluted by outliers, overcoming RANSAC limitation for non-rigid surface detection. A well designed robust estimator, along with a deformable 2-D mesh, is key to dealing with this large number of parameters. The keypoints positions are expressed as weighted sums of the mesh vertices in the model image and change as the mesh is deformed. Fitting then amounts to minimizing a criterion that is the sum of two terms. The first is a robust estimate of the squared distances of the keypoints in the model image to that of the corresponding ones in the input image. The second is a quadratic deformation energy (Fua, 1997). As was the case for the original snakes (Kass et al., 1988), this quadratic term allows the use of a semi-implicit minimization scheme that converges even when the initial estimate is very far from the solution, which, in our context, is what happens when the object is severely deformed. When combined with an appropriately defined robust estimator for the keypoint distances and optimization schedule, this approach to minimization allows detection in under 100 milliseconds on a 2.8 GHz desktop while being robust to large deformations, severe occlusions, and changes in lighting. In fact, we have verified that our method keeps on working with more than 90% of point matches being erroneous, which is key to robustness because no real-time matching technique can be expected to work perfectly well in the presence of clutter, orientation changes, shadows, or specularities. We do not know of any other technique able to produce similar results.

In the remainder of the paper, we first review briefly the existing literature and present an overview of our algorithm. We then discuss its most critical steps and our results.

2. Related Work

Many approaches to registering a model on an image have been proposed. Some feature-based algorithms first establish correspondences and then find the best transformation explaining them, while eliminating outliers. Others simultaneously solve for both correspondence and registration, without the need for correspondences and with or without using feature characterization. Finally, some techniques do not even rely on features. We review them briefly below and discuss why they have not yet been shown to be suitable for real-time detection of deformable objects.

2.1. Feature-Based Methods

These approaches rely on establishing correspondences between image-features of the target object in one or more images and those that can be found in an input image in which it is to be detected. These correspondences are then used to estimate the transformations.

Establishing Correspondences. Our method relies on establishing wide-baseline correspondences between a training image and an input one. To be useful, correspondences have to be insensitive to light and viewpoint changes, as well as to some amount of non-rigid deformation.

Among the many matching techniques that exist, we tested three: SIFT Lowe (2004), shape context descriptors (Belongie et al., 2002) and our own classification based method (Lepetit et al., 2005).

Even if these algorithms differ in speed, correspondences number and quality, the tests presented in Section 5 show that our algorithm's effectiveness is independent from the specific technique used to establish the point correspondences. However, only the classification based technique has proved fast enough for our purpose, real-time detection without loss of accuracy. The technique recently presented in Ling and Jacobs (2005) could fit our needs since it is deformation invariant.

From Correspondences to Detection. Whatever the matching technique used, the correspondences can then be used to detect the object in several different ways.

The simplest is to eliminate outliers and find a globally consistent interpretation using a robust estimator. Having each local match vote for a global transformation is the approach used by the Hough transform and its many variations. This is effective for rigid objects but impractical for deformable ones because it would require far too many degrees of freedom to represent all possible transformations into a vote accumulator. The same can be said of the popular RANSAC algorithm (Fischler and Bolles, 1981): With 25% of outliers and 100 degrees of freedom, 10^{12} samples are required to guarantee with 90% probability that at least one sample does not contain outliers (Hartley and Zisserman, 2000).

An alternative strategy is to proceed iteratively. TPS-RPM (thin plate spline - robust point matching, Chui and Rangarajan (2003)) and EM-ICP (expectation maximization—iterative closest point, Dewaele et al. (2004) and Granger and Pennec (2002)) are two well-known representatives of the family of algorithms that simultaneously solve for both correspondence and transformation using an iterative process. At each step, the current transformation estimate is first used to establish correspondences and assign weights to them, and, then, is refined using those correspondences. These methods use an entropy term—be it called temperature parameter, scale or blurring factor, or variance—that is progressively reduced. It controls the assignment of weights to the correspondences and has an important role in insuring convergence towards a desirable solution. As will be discussed in more detail in Section 3.2, our algorithm follows a similar strategy but makes use of local characterization to reduce the correspondence problem difficulty and to achieve real-time performance.

In Belongie et al. (2002), a method designed to compute a distance between shapes is presented. Shape context descriptors provide correspondences which are established one to one using bipartite graph matching. Although this method copes with some outliers and slightly different numbers of feature detected on both shapes, it is not designed to extract objects from a cluttered background or to handle scale changes.

Image exploration (Ferrari et al., 2004) is another strategy that hooks on a first set of correspondences and then gradually explores the surrounding area, trying to establish more matches. It can handle deformable objects but this complex process is slow and takes several minutes on a 1.4 GHz computer.

2.2. Direct Methods

For objects such as faces whose deformations are well understood and can be modeled in terms of a relatively small number of deformation parameters, fitting directly to the image data without using features is an attractive alternative to using correspondences because it allows the use of global constraints to guide the search. This has been successfully demonstrated in the context of non-rigid tracking (Baker et al., 2004; Cootes et al., 2001; DeCarlo and Metaxas, 1998; Sclaroff and Isidoro, 2003) but typically requires a good initialization because the criteria being minimized tend to have many local minima.

These methods are complementary to the one proposed here: They exploit more of the texture and therefore tend

to be more accurate. However, they require the initial estimate such as the one our algorithm can provide. There are in fact relatively few others that can do this for deformable objects. One of them has been proposed in Gumerov et al. (2004) but requires that the whole outline be detected, which severely limits its scope. Another is the tracking of Lin and Liu (2006) that exploits the repeating properties of a near regular texture to discover new texture tiles in new frames.

Finally, the recent work presented in White and Forsyth (2006) is related to ours in two ways. First, it registers a texture composed of a few colors, typically 3 or 4, by comparing color histograms. Then, it modifies the texture on the deformed surface, while handling illumination changes. This approach to retexturing differs from ours in that we avoid limiting the number of colors present on the surface by introducing some irradiance smoothing, which yields real-time performance on both color or gray level images.

3. Non-rigid Surface Detection

To detect a potentially deformable object, we rely on establishing correspondences between a model image in which the deformations are small and an input image in which they may be large. To this end, we use the fast wide-baseline matching algorithm (Lepetit et al., 2005) discussed in Section 2.1. Given a set C of correspondences between the two images, many of which might be erroneous, our problem can be formally stated as follows: We are looking for the transformation T_S mapping the undeformed model surface M into the deformed target one $T_S(M)$ and for the subset $G \subset C$ of correct matches such that the sum of the squared distances between corresponding points in G is minimized while the deformations remain as smooth as possible.

3.1. 2-D Surface Meshes

We represent our model M as a triangulated 2-D mesh of hexagonally connected vertices such as the one shown in Fig. 2. The position of a vertex v_j is specified by its image coordinates (x_j, y_j). The overall shape is therefore controlled by a state vector S that is the vector of all x and y coordinates. Given S and the barycentric coordinates $B_i(p)$ of image point p that belongs to a specific facet (v_1, v_2, v_3) of the undeformed mesh, we define the mapping

$$T_S(p) = \sum_{i=1}^{3} B_i(p) \begin{bmatrix} x_i \\ y_i \end{bmatrix}, \tag{1}$$

where x_i and y_i are vertex coordinates of the deformed mesh.

The mesh deforms to minimize the objective function

$$\varepsilon(S) = \lambda_D \varepsilon_D(S) + \varepsilon_C(S), \tag{2}$$

where ε_C is a data term that takes point correspondences into account, ε_D is a deformation energy that should be rotationally invariant and tend to preserve the regularity of the mesh, and λ_D is a constant.

We take $\varepsilon_D(S)$ to be an approximation of the sum over the surface of the square second derivatives of the x and y coordinates. More specifically, let E be the set of vertex index triplets (i, j, k) such that (v_i, v_j, v_k) form two connected and colinear edges, as illustrated by Fig. 3(a). Since the undeformed mesh M has equidistant vertices,

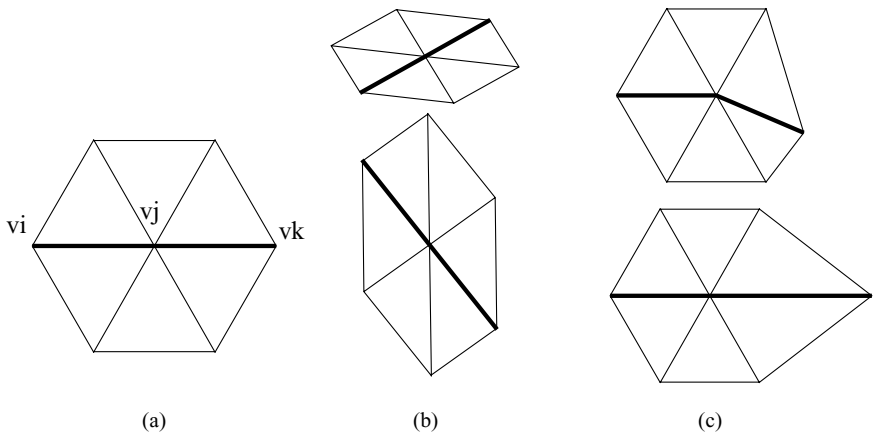


Figure 3. 2D mesh models. (a) Vertex neighborhood in an undeformed hexagonal mesh. (b) Two deformations that are not penalized. (c) Two penalized deformations. Deformations resulting from perspective projection resemble those in (b) and are therefore much less severely penalized than those resulting from erroneous matches.

we have

$$\forall (i, j, k) \in E : v_i - v_j = v_j - v_k, \tag{3}$$

and therefore write

$$\varepsilon_D(S) = \frac{1}{2} \sum_{(i,j,k)\in E} (-x_i + 2x_j - x_k)^2 + (-y_i + 2y_j - y_k)^2. \tag{4}$$

$\varepsilon_D(S)$ approximates the squared directional curvature of the surface as long as the vertices remain roughly equidistant and its value grows with the length difference of every two colinear connected edges.

This regularization term serves a dual purpose. First it *convexifies* the energy landscape and improves the convergence properties of the optimization procedure. Second, in the presence of erroneous correspondences, some amount of smoothing is required to prevent the mesh from overfitting the data, and wrinkling the surface excessively. As illustrated by Fig. 3(b) and (c), ε_D is appropriate for this purpose because it allows rigid motions but penalizes shape deformations. Of course, both those produced by perspective distortions and by the actual surface deformation tend to increase ε_D. However, this increase is insignificant when compared to those that spurious deformations resulting from erroneous matches could produce.

Equation (4) can be rewritten in matrix form as

$$\varepsilon_D(S) = \frac{1}{2}(X^T K'^T K' X + Y^T K'^T K' Y), \tag{5}$$

where K' is a matrix containing one row per triplet in E and one column per mesh vertex. The row corresponding to triplet (i, j, k) is filled with zeroes except for locations i, j and k that contain -1, 2, and -1, respectively. By replacing $K = K'^T K'$ in Eq. (5), we have:

$$\varepsilon_D(S) = 1/2(X^T K X + Y^T K Y). \tag{6}$$

To minimize $\varepsilon(S)$, we use the semi-implicit scheme so successfully introduced in the original snake paper (Kass et al., 1988): We are looking for a minimum of the energy and therefore for solutions of

$$\begin{aligned} 0 = \frac{\partial \varepsilon}{\partial X} = \frac{\partial \varepsilon_C}{\partial X} + KX, \\ 0 = \frac{\partial \varepsilon}{\partial Y} = \frac{\partial \varepsilon_C}{\partial Y} + KY. \end{aligned} \tag{7}$$

Since K is positive but not definite, given initial vectors X_0 and Y_0, this can be solved by introducing a viscosity parameter α and iteratively solving at each time step the two coupled equations

$$\begin{aligned} KX_t + \alpha(X_t - X_{t-1}) + \left.\frac{\partial \varepsilon_C}{\partial X}\right|_{X=X_{t-1}Y=Y_{t-1}} = 0, \\ KY_t + \alpha(Y_t - Y_{t-1}) + \left.\frac{\partial \varepsilon_C}{\partial Y}\right|_{X=X_{t-1},Y=Y_{t-1}} = 0, \end{aligned}$$

which implies

$$\begin{aligned} (K + \alpha I)X_t = \alpha X_{t-1} - \left.\frac{\partial \varepsilon_C}{\partial X}\right|_{X=X_{t-1},Y=Y_{t-1}}, \\ (K + \alpha I)Y_t = \alpha Y_{t-1} - \left.\frac{\partial \varepsilon_C}{\partial Y}\right|_{X=X_{t-1},Y=Y_{t-1}}. \end{aligned}$$

Because K is sparse and regular, solving these linear equations using LU decomposition is fast and upon convergence $X_t \approx X_{t-1}$ and $Y_t \approx Y_{t-1}$. This iterative scheme therefore quickly yields a solution of Eq. (8), even when starting with completely random guesses for X_0 and Y_0 as will be shown in Section 4.

3.2. Correspondence Energy

Minimizing ε_C, the data term of Eq. (2), tends to deform the mesh so that it matches the target object in the input image. This is achieved as follows.

Let C be a set of *correspondences* between the model and the input image. Its elements are of the form $c = \{c_0, c_1\} \in C$, where c_0 represents the 2-D coordinates of a feature point in the model image and c_1 the coordinates of its match in the input image. For the sake of generality, we allow potential matches between a point in the first image and multiple points in the second, so that the corresponding c_0 may appear in several elements of C. We write

$$\varepsilon_C = -\sum_{c\in C} w_c \rho\left(\left\|c_1 - T_{\mathbf{S}}(c_0)\right\|, r\right), \tag{8}$$

where ρ is a robust estimator whose *radius of confidence* is r and $w_c \in [0, 1]$ a weight associated to each correspondence. In our experience the choice of ρ is critical to ensure the elimination of outliers and convergence towards the desired minimum while the choice of the w_c has much less impact, as will be discussed in Section 4.1.

We take the robust estimator to be

$$\rho(\delta, r) = \begin{cases} \frac{3(r^2-\delta^2)}{4r^3} & \delta < r \\ 0 & \text{otherwise} \end{cases}. \tag{9}$$

As shown in Fig. 4, its shape is that of a quadratic ridge that gets narrower and taller when r decreases. In other words, r acts as a confidence measure. When it is large, most correspondences, potentially including poor

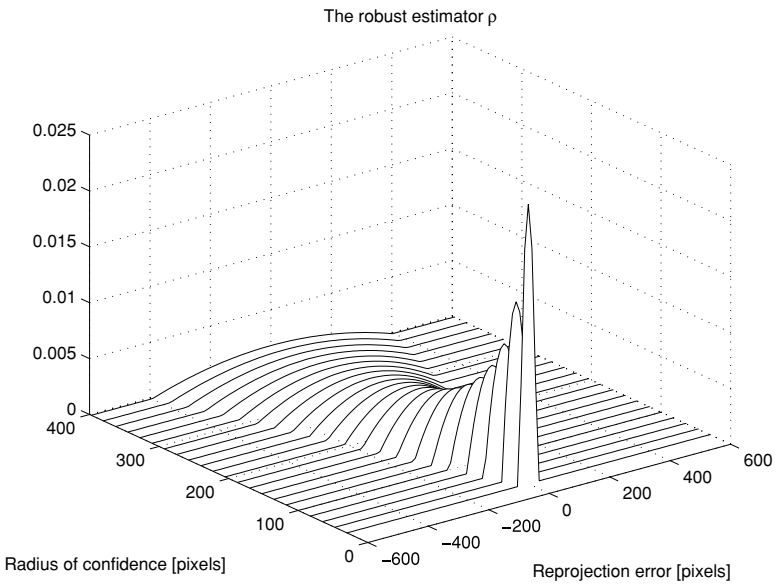


Figure 4. The ρ function of Eq. (9) is quadratic for distances smaller than the radius of confidence, elsewhere it is zero.

ones, fall within this broad *ridge of confidence* and are given some weight. As r diminishes, ρ becomes more peaked and selective. This formulation has the following advantages:

- The quadratic behavior of ρ within the ridge of confidence yields a relatively convex ε_C that is easy to minimize.
- ρ is normalized so that $\int_{-\infty}^{\infty} \rho(x, r)dx = 1 \quad \forall r > 0$, which means that the ε_C term computed with any r values remain commensurate to the $\lambda_D \varepsilon_D$ term of Eq. (2). Therefore, we do not need to adjust either the λ_D parameter or the w_c weights of Eq. (8). This is in contrast to methods such as SoftAssign (Chui and Rangarajan, 2003; Wills and Belongie, 2004) in which the surface rigidity must be progressively reduced according to a schedule that is not necessarily easy to synchronize with the annealing of r and may change from case to case.
- ρ has finite support so that correspondences that fall outside the radius of confidence are completely ignored and can be tagged as invalid.

These properties of the ρ estimator are what make the straightforward approach to optimization described below so effective.

3.3. Optimization Schedule

Minimizing ε therefore results in a mesh that moves towards the desired solution but whose progression can be blocked by outliers. To overcome this, we introduce a simple optimization schedule in which the initial *radius of confidence* $r_0 = 1000$ is progressively reduced at a constant rate $\eta = 0.5$: $r_t = \eta r_{t-1}$. For each value of r, we minimize ε and use the result as the initial state for the next minimization.

As discussed in Section 3.1, at each iteration of our semi-implicit optimization scheme, we evaluate the derivatives of ε_C. In this context, the fact that ρ has derivatives whose magnitude is inversely proportional to r is very beneficial: At the beginning when r is large, the gradients of ε_D are comparatively larger than those of ε_C, thus preventing erroneous matches from crumpling the surface while allowing correct and consistent ones to produce the right global deformation. As the optimization progresses and r decreases, the ρ derivatives and consequently the gradients of ε_C become larger. The triangulation starts bending as appropriate and the influence of the outliers progressively decreases.

The algorithm stops when r reaches a value close to the expected precision of the matches expressed in pixels, typically one or two. Such a deterministic algorithm is guaranteed to converge but the result might be wrong, for example because the target object is completely occluded. To decide whether or not to believe the result, we simply count the number of correspondences that fall within the ridge of confidence of our ρ estimator. As will be shown in Section 4, this criterion is surprisingly effective at distinguishing successes from failures.

4. Synthetic Experiments

In this section, we use synthetic data to illustrate the effectiveness of our implementation choices. More

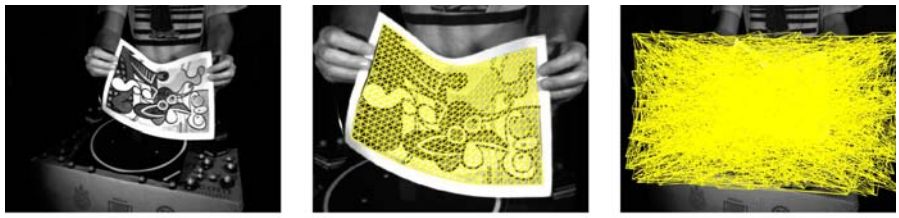


Figure 5. Image and meshes used for our synthetic experiments. (a) Original image. (b) Reference mesh computed using hand-picked correspondences. (c) A random initial configuration.

specifically we show that our algorithm is insensitive to parameter choices, insensitive to initial conditions, and effective at rejecting false matches.

Figure 5 depicts our approach to creating synthetic data for these experiments. We fed our algorithm with manually established correspondences between a model image in which the sheet of paper is flat, and the image of Fig. 5(a) until we obtained the 600-vertex deformed mesh of Fig. 5(b), which projects correctly over its whole surface. We treat this mesh as our reference, which can be viewed as the ideal result that can be expected from our algorithm. In the remainder of this section we will use different sets of correspondences, randomized initial conditions, and modified parameter settings. They produce different results that can then be compared to our reference. Proceeding in this manner ensures that the deviations we measure are strictly related to what we are trying to measure, as opposed to pose dependent problems.

4.1. Measuring Success

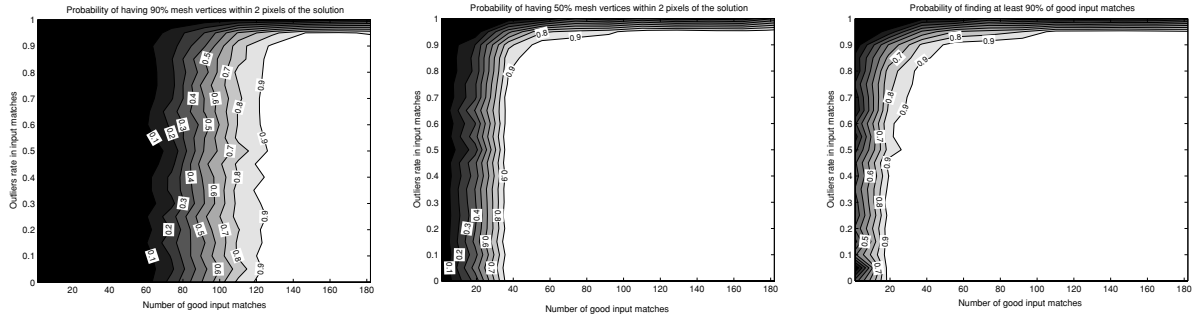
We define three objective success criteria:

- $\mathbf{C}_1$ 90% of the mesh vertices are within 2 pixels of those in the reference mesh.
- $\mathbf{C}_2$ 50% of the mesh vertices are within 2 pixels of those in the reference mesh.
- $\mathbf{C}_3$ At least 90% of the valid correspondences given as input are correctly labeled as such by the robust estimator, as discussed in Section 3.2.

Given that the test image is of dimension 1024×768, $\mathbf{C}_1$ and $\mathbf{C}_2$ rate the algorithm's accuracy and $\mathbf{C}_3$ its ability to discriminate valid correspondences from spurious ones. The 90% figure in $\mathbf{C}_1$ eliminates cases where a substantial part of the mesh is incorrectly reconstructed, even though the algorithm may have done a good job on the rest, a case that $\mathbf{C}_2$ labels as correct.

To test our algorithm, we ran it about one hundred thousand times with random initial conditions, such as the one of Fig. 5(c) that is very far from the solution, and using synthetic sets of correspondences containing varying numbers of valid matches and percentages of erroneous ones.

Accuracy and Robustness. Figure 6 depicts the success rates according to the $\mathbf{C}_1$, $\mathbf{C}_2$, and $\mathbf{C}_3$ criteria introduced above as a function of the number of valid correspondences and of the outlier rate. In each plot, the color depicts the percentage of results that meet the corresponding criterion. The black wiggly lines represent level lines in this probability landscape. Note that in all three plots, they are nearly vertical for outlier rates up to 90%, thus indicating that the performance does not significantly degrade before then. Given that $\mathbf{C}_1$ is much more stringent



$\mathbf{C}_1$ criterion $\mathbf{C}_2$ criterion $\mathbf{C}_3$ criterion

Figure 6. Probability of success according to the three criteria of Section 4.1 as a function of the number of valid input matches, on the horizontal axis, and the outlier rate, on the vertical axis. White indicates values close to one and black close to zero.

than $\mathbf{C}_2$, it is natural that it requires more valid matches to achieve the required level of precision. To recover 50% mesh vertices location, 40 matches are enough and 120 for 90%. This is very encouraging considering that the mesh has 600 vertices, which would imply 1200 degrees of freedom in the absence of regularization constraints. $\mathbf{C}_3$ is the less demanding of the three criteria and requires less than 15 to 20 valid correspondences. However, success in terms of $\mathbf{C}_3$ does not guarantee accuracy for large outlier rates because, even though the algorithm still finds most of the inliers, it starts mistakenly tagging outliers as valid matches, which degrades the precision.

Self Diagnostic. So far, we have compared our results against a manually obtained reference mesh. In practice, the algorithm has to self-diagnose its own successes and failures in the absence of any such reference. As a substitute, we use the absolute number of matches that are tagged as valid by the ρ estimator of Eq. (9) as a measure of success. In other words, our algorithm declares a successful detection when the number of valid matches is above a given threshold. In Fig. 7, we plot the corresponding ROC curves according to $\mathbf{C}_1$, $\mathbf{C}_2$, and $\mathbf{C}_3$. These curves indicate an excellent correlation between "objective" success, as measured by comparison to a reference result, and "subjective" success, as measured by the number of matches tagged as valid.

One limitation of the current approach, however, is that we only implemented a global success measure: The surface is either completely found or not at all. An interesting extension would be to measure partial success, for example in cases where the surface is partly occluded, by checking sub-areas as opposed to the whole surface.

Disambiguating Multiple Matches. Recall from Section 3.2 that a point from the model image can have several potential matches in the input image. One can simply rely on the progressively decreasing r radius of confidence of the ρ estimator to disambiguate those cases. Alternatively one could use a more sophisticated weighting scheme, an option we explore here by setting the w_c weights of Eq. (8) in one of the five following ways:

1. $w_c = 1$ for all correspondences,
2. $w_c = 1$ for the closest match, and zero to all others as in ICP,
3. as in EM-ICP (Granger and Pennec, 2002), with $\sigma = \frac{r}{3}$:

$$w_c = \frac{\exp(-\|c_1 - T_{\mathbf{S}}(c_0)\|^2/2\sigma^2)}{\sum_{d \in C, d_o = c_o} \exp(-\|d_1 - T_{\mathbf{S}}(d_0)\|^2/2\sigma^2)},$$

4. a variation of EM-ICP in which the Gaussian is replaced by ρ:

$$w_c = \frac{\rho(\|c_1 - T_{\mathbf{S}}(c_0)\|, r)}{\sum_{d \in C, d_0 = c_0} \rho(\|d_1 - T_{\mathbf{S}}(d_0)\|, r)},$$

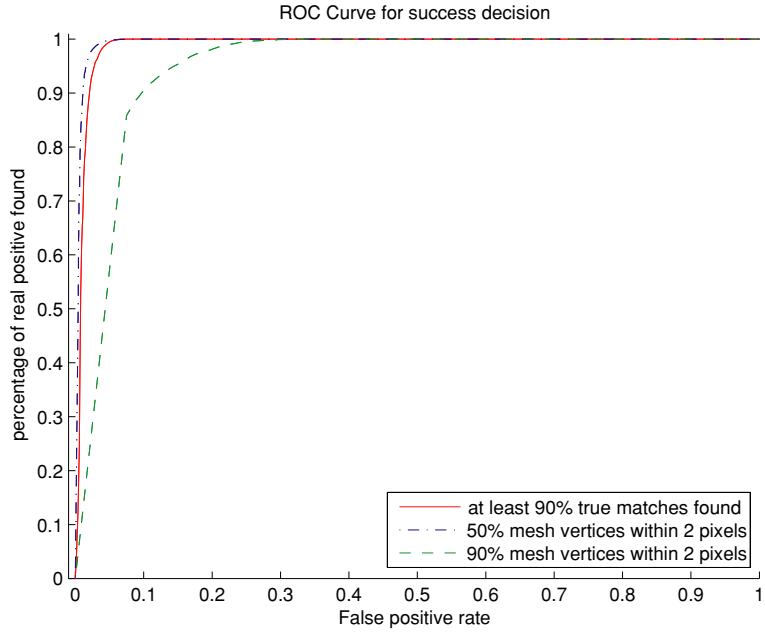


Figure 7. Self diagnostic ROC curves. We accept a detection result based on the number of matches tagged as valid by our robust estimator. We plot one curve for each one of the three criteria of Section 4.1. For each one, as the threshold for accepting a result is lowered, both the false-positive rate, on the x-axis, and the true-positive rate, on the y-axis, increase.

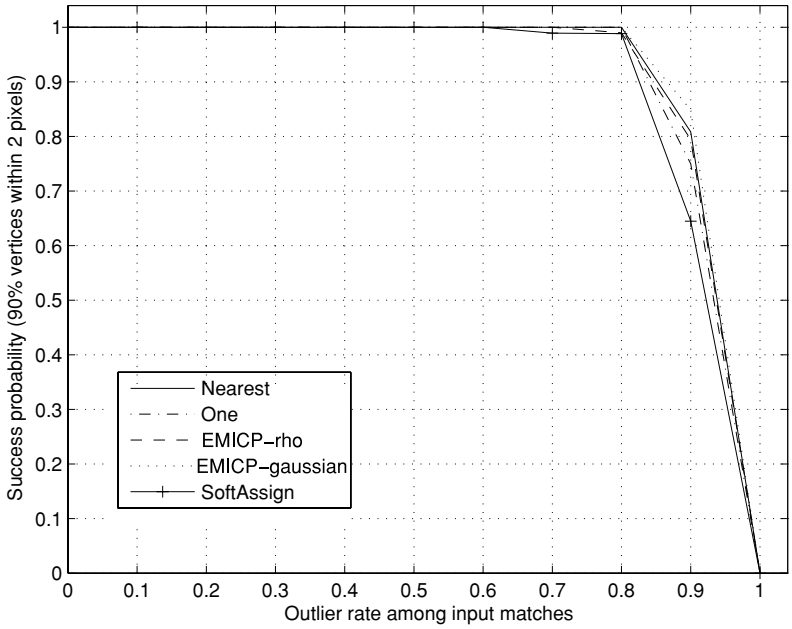


Figure 8. Comparing weighting schemes. Success rate as a function of erroneous correspondences percentage, for each one of the five schemes described in Section 4.1.

5. a weight computed by normalizing rows and columns of the correspondence matrix, as in SoftAssign (Chui and Rangarajan, 2003).

Figure 8 summarizes the result of this experiment. We used 150 valid matches and a variable number of spurious matches. We plot success rates according to the $\mathbf{C}_1$ criterion as a function of the percentage of outliers. Note that these curves correspond to a vertical slice of Fig. 6($\mathbf{C}_1$) and are very close to each other. For our specific purpose, but obviously not in a more general context, their respective performance are almost indistinguishable, but not their computational cost. In our real-time implementation, we therefore use the simplest one and set all w_c to one.

4.2. *Parameters and Initial Conditions*

We now turn to the influence of our parameter choices and of the initial conditions. We show that they influence the speed at which the algorithm converges much more than its final result.

Regularization Weight. In the $\varepsilon(S)$ total energy of Eq. (2), the relative influence of the regularization and observation terms is controlled by the λ_D parameter. It represents surface stiffness: The larger it is, the more deformations are penalized. If it is too large, legitimate bending might be prevented. If it is too small, the mesh may wrinkle excessively and treat some spurious correspondence as valid. In Fig. 9, we again use a fixed number of valid matches and plot success rates according to the $\mathbf{C}_1$ criterion as a function of the percentage of outliers and of the λ_D value used to perform the computation. For outlier rates below 60%, and even up to 80%, λ_D can be chosen in a very wide range without significantly affecting the results. As the outlier rate increases, larger λ_D values appear to give better results. It is to emphasize these large values of λ_D that we chose to plot $\frac{1}{\lambda_D}$ on the vertical axis of the graph.

Deterministic Radius Reduction. In our algorithm, the confidence radius r of Eq. (9) is decreased by a factor η after each minimization. Even though this deterministic approach might seem simplistic, we prefer it because, in practice, it is very hard to evaluate a new radius from currently valid matches. For example, a similar problem is solved in Rosten and Drummond (2005) using an expectation minimization (EM) approach and the authors have to "*give a kick downwards*" to their blurring factor when EM converges too early. Here we show that value of η has relatively little impact on the optimizer's behavior.

To this end, we randomly chose one particular set of correspondences and ran the optimizer several times using η values ranging 0.3 and 0.8. In Fig. 10(a), for each trial, we plot the number of individual Levenberg-Marquardt steps performed to minimize the total-energy of Eq. (2) for each successive value of r. Note how similar the curves are. The total number of Levenberg-Marquardt steps required to locate the surface does not change much. If the radius decreases slowly, the optimizer will use more

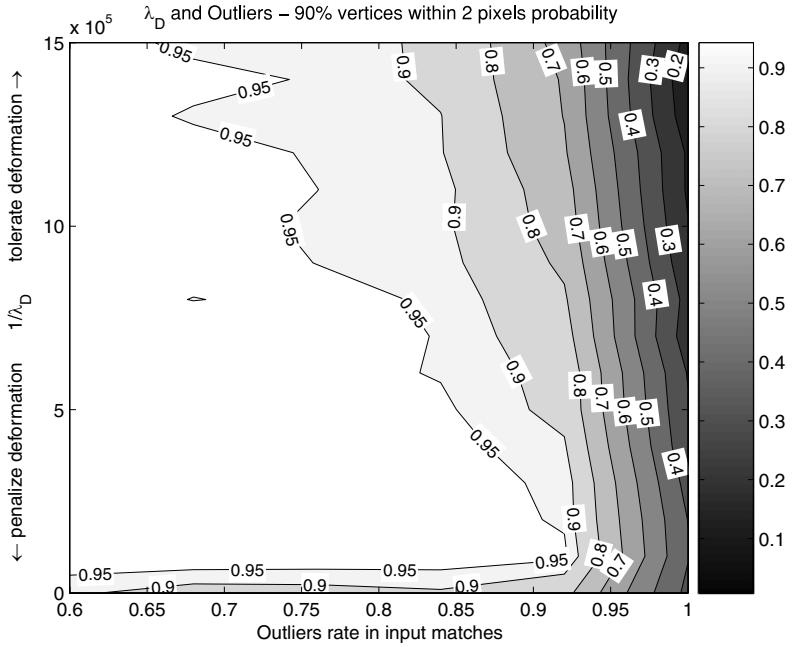
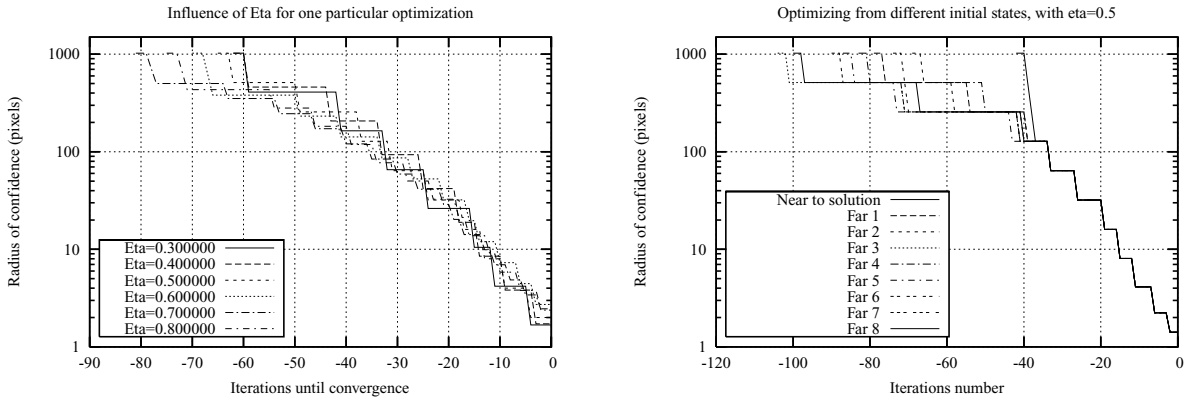


Figure 9. Probability of success given $\frac{1}{\lambda_D}$ and the input outlier rate. The choice of λ_D is only important for very high outlier rates.



(a) (b)

Figure 10. Number of individual Levenberg-Marquardt steps to achieve convergence. (a) For several values of η, number of steps required to minimize the total-energy of Eq. (2) for each successive value of r. (b) Similar plot for nine initial states, chosen to be increasingly far from the reference mesh.

radius values but will require fewer iterations at each. If the radius decreases faster, the situation is reversed but the global outcome is similar.

Sensitivity to Initial Conditions. Because our algorithm appears to be very effective at avoiding local minima, the choice of initial condition has little bearing on success or failure. It does however have an influence on the time required to achieve convergence.

To demonstrate this, we again randomly picked a set of correspondences and ran the optimizer several times using nine different initial conditions, chosen to be increasingly far from the reference mesh. The algorithm yielded the same result in all cases and Fig. 10(b) depicts the number of individual Levenberg-Marquardt steps performed for each successive value of r during each run. Starting close to the solution saves iterations for large values of r but not for small ones. Nevertheless, this behavior could obviously be exploited in a tracking context where a good initial estimate is usually available.

5. Results

We now turn to real images and video sequences. We first demonstrate that our approach leads to a real-time

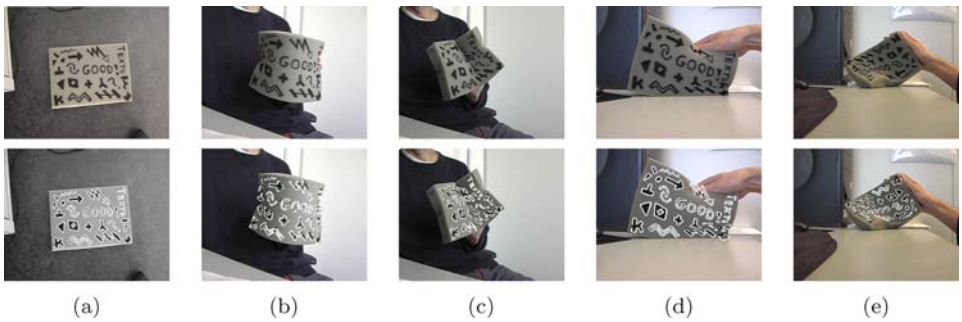


Figure 11. Deforming a piece of foam. (a) Model image and validation texture. (b) to (e) detection results.

robust implementation. We then show that it can be incorporated into an Augmented Reality application that accounts not only for geometry but also illumination, thus producing very convincing illusions, still in real-time.

5.1. Real Time Detection

The method has been tested in conjunction with three different feature point recognizers: The publicly available SIFT implementation (Lowe, 2004), a reimplementation of shape context characterization (Belongie et al., 2002), and a classification-based method (Lepetit et al., 2005). SIFT provide fewer but more accurate matches than shape contexts. The classification-based approach produces correspondences comparable to SIFT but does it faster. Because our technique is robust, the results are almost indistinguishable whatever the matching method used, as shown in Fig. 2. However, because the classification-based method is much faster than the others, it is only when using it that we obtain true real-time performance. In this example, the algorithm runs at 10 frames per second on a 2.3 GHz laptop. Furthermore, because the point matcher is relatively insensitive to light changes and motion blur, they do not hinder the registration process.

Since we work in each frame individually, we can find objects as soon as they become visible and our method is robust to both perspective distortion and severe deformations. In the example of Fig. 1, the ICCV logo on the shirt is detected very quickly and well before its deformation has become roughly planar. Similarly, the logo is equally well detected when worn by different people or seen on the ICCV mug. Figure 11 depicts similar speed and robustness to deformations when detecting a piece of foam. For well textured objects, we get no false positives and only false negatives when the deformations or occlusions are so severe that the target object is almost impossible to make out. Of course, the performance degrades in the absence of texture and this is one of the issues we will address in future work.

5.2. Realistic Augmented Reality

So far, we have shown that we could compute fast and accurately the 2-D deformation of a surface. In an Augmented Reality application such as the one depicted by Fig. 12, this is what is needed to modify in real-time the appearance of that surface. However, to achieve a convincing illusion, it is important not only to model geometric deformations but also lighting changes. To this end, we have developed a dynamic approach to estimating the amount of light that reaches individual image pixels by comparing their colors to those of the model image. This lets us either erase patterns from the original images and replace them by blank but correctly shaded areas, which we think of as *Diminished Reality*, or to replace them by virtual ones that convincingly blend-in because they are properly lighted.

The Lambertian Case. In practice, if we wish to build a versatile system that can be demonstrated in uncontrolled environments, we cannot make strong assumptions about light sources that are present when acquiring the input video. There can be many and their respective intensities and spectral properties are unknown, which can result in complex shading, shadowing, and color effects. To avoid the latter, we work independently on the red, green, and blue bands of color images.

However, it is easy to control the acquisition of the reference image. With no loss of generality, we can therefore assume that it has been acquired when the surface was both undeformed and lighted uniformly, which means that every surface point receives the same amount of light in the color band we are working with.

Under this assumption, let $\mathbf{p}_r$ and $\mathbf{p}_i$ be the projections of the same surface point p in the reference and input image respectively, and let A_p be the corresponding surface albedo. In the Lambertian case, the contributions of all the light sources seen at $\mathbf{p}_r$ and $\mathbf{p}_i$ add linearly. We can therefore write

$$I_{r,p} = L_r A_p, \tag{10}$$

$$I_{i,p} = L_{i,p} A_p, \tag{11}$$

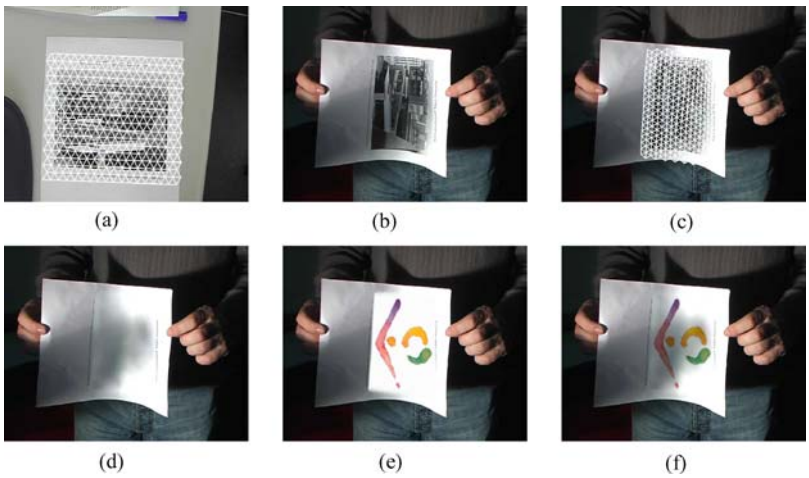


Figure 12. (a) The reference image of the target surface with the model mesh overlaid. (b) An input image. (c) The mesh is correctly deformed and registered to the input image. (d) The original pattern has been erased and replaced by a blank but correctly shaded image. (e) A virtual pattern replaces the original one. It is correctly deformed but not yet relighted. (f) The virtual pattern is deformed and relighted.

where $I_{r,p}$ and $I_{i,p}$ are the pixel intensities in the reference and input image respectively, $L_{i,p}$ the total irradiance in the input image at $\mathbf{p}_i$, and L_r the total irradiance in the reference image assumed to be the same at all surface points. In general, the values of $I_{r,p}$ and $I_{i,p}$ are different due to changes in both normal orientations and lighting conditions. However, the geometric registration we have established between the two images tells us that they correspond to the same physical point, which we exploit as follows.

Let us consider a white surface area with albedo A_w at location w on the surface. If the target surface has no white part, it is always possible to put a white object next to it while taking the reference image. We can measure on the reference image the pixel intensity $I_{r,w}$ where this white location w is projected and write

$$I_{r,w} = L_r A_w, \tag{12}$$

where L_r is the irradiance of Eq. (10).

Using this white normalization $I_{r,w}$, we can compute a new image, looking similar to the input one, except that the surface albedo is changed to A_w. In the input image, if there was no texture, the corresponding image intensity should be

$$I_{x,p} = L_{i,p} A_w = A_w L_r \frac{I_{i,p}}{I_{r,p}} = I_{r,w} \frac{I_{i,p}}{I_{r,p}}. \tag{13}$$

Note that $I_{x,p}$ is expressed exclusively in terms of image intensities, which are readily available, as opposed to albedoes or surface normals that are not.

Replacing the intensities $I_{i,p}$ of all the pixels on the object surface by $I_{x,p}$ yields images such as the one of Fig. 12(d) where the original texture has been replaced by a blank but correctly shaded surface. To draw a shaded new texture, as in Fig. 12(f), we simply multiply texture values with their corresponding white $I_{x,p}$.

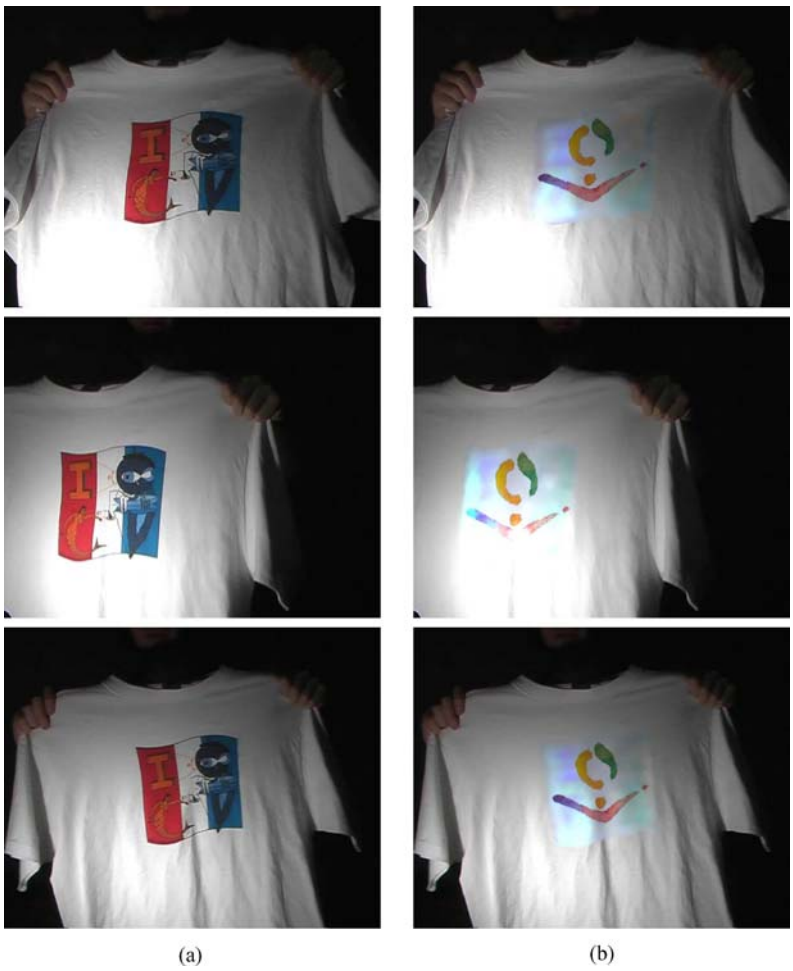
Note that, because we perform the computation locally, it remains valid no matter how many sources there are and what their specific characteristics may be. The only thing that has to be true is that the contribution of the individual light sources to the pixel intensity are all modulated by the same diffuse albedo and do not depend on the viewpoint.

In practice, we compute the lighting factor only at mesh vertices, averaging pixels values of both model and input images over an hexagonal area surrounding it. The resulting $I_{x,p}$ values are then interpolated over triangles by OpenGL.

In some cases, $I_{x,p}$ is difficult to estimate reliably on large single-colored areas. In the example of Fig. 13(a), recovering the $I_{x,p}$ blue component over the red area is hard because sensor inaccuracy on remaining blue light is amplified by a big factor. However, the visual impression given by Fig. 13(b) is still that the original painting has been erased and replaced.

Specularities and Saturation. The assumptions used to derive formula 13 are clearly violated for specular materials. However, as illustrated by Fig. 14, this does not have severe consequences even in the presence of strong specularities and the illusion remains convincing.

This is because, when there is a specularity, the image intensity increases and the $\frac{I_{i,p}}{I_{r,p}}$ ratio of Eq. (13) becomes large. As a result, the $I_{x,p}$ intensity that is used to draw the synthetic patterns also increases, which is perceptually correct since it yields intensity maxima at specularities' locations. In other words, the absolute value of $I_{x,p}$ may not be correct but its magnitude relative to its neighbors remains consistent with the presence of a specularity. And



(a) (b)

Figure 13. (a) original image. (b) The ISMAR logo replaces the shirt print. Recovering white is hard in this image since the model has large single-colored areas, making light evaluation difficult.

(a) (b) (c)

Figure 14. Handling specularities. (a) Input image with strong specularities. The main one is produced by a lamp, while the two smaller ones can be attributed to light coming through window. To produce this result, the paper has been covered by a transparent plastic sheet. (b) The picture has been erased from the surface but the specularities still appear to be at the right places. (c) The ISMAR logo has been inserted.

since the human eye is much more sensitive to relative values than to absolute ones, this suffices.

In practice, specular peaks often saturate the camera sensor, thus making the estimation of $I_{i,p}$ unreliable. We detect such cases by simple thresholding and we handle saturation by setting $I_{x,p}$ to its maximal possible value. Since color computation is applied independently on the red, green and blue channels, one channel can saturate while the other do not. As a result, not only specular peaks but also saturated areas in the input image are correctly transcribed into the synthetic ones.

6. Conclusion

We have demonstrated a very fast and robust approach to detecting deformable surfaces. It is robust to large deformations, changes in lighting, and motion blur and

runs at 8–10 frames per second on a 2.3 GHz laptop. It takes advantage of wide-baseline matching, deformable mesh and robust estimation techniques in such a way that the resulting algorithm has very few parameters that do not require any fine tuning. As a result, it was easy to incorporate it into a real-time Augmented Reality system that produces convincing illusions even when the illumination is complex.

The current computations are performed using 2-D meshes but the formalism presented in this paper naturally extend to 3-D, with only a very limited additional computational burden. This should be key to handling even more severe self-occlusions than the ones shown in this paper and, also, to incorporate physical knowledge about the deformation modes of the surface if they are known. This should help us handle less textured objects than the ones we have worked with so far, that is objects for which fewer interest points can be detected and matched. An alternative way to deal with relatively bland surfaces would be to broaden the definition of interest points to include those that can be found along contours, as opposed to corners, and could also be considered within our framework. We intend to pursue both avenues of research in future work.

Acknowledgments

This work was supported in part by the Swiss National Science Foundation.

References

Baker, S., Matthews, I., Xiao, J., Gross, R., Kanade, T., and Ishikawa, T. 2004. Real-time non-rigid driver head tracking for driver mental state estimation. In *World Congress on Intelligent Transportation Systems*.

Bartoli, A. and Zisserman, A. 2004. Direct estimation of non-rigid registration. In *British Machine Vision Conference*, Kingston, UK.

Belongie, S., Malik, J., and Puzicha, J. 2002. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(24):509–522.

Chui, H. and Rangarajan, A. 2003. A new point matching algorithm for non-rigid registration. *Computer Vision and Image Understanding*, 89(2–3):114–141.

Cootes, T.F., Edwards, G.J., and Taylor, C.J. 2001. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6).

DeCarlo, D. and Metaxas, D. 1998. Deformable model-based shape and motion analysis from images using motion residual error. In *International Conference on Computer Vision*, Bombay, India, pp. 113–119.

Dewaele, G., Devernay, F., and Horaud, R. 2004. Hand motion from 3d point trajectories and a smooth surface model. In *European Conference on Computer Vision*, pp 495–507.

Ferrari, V., Tuytelaars, T., and Van Gool, L. 2004. Simultaneous object recognition and segmentation by image exploration. In *European Conference on Computer Vision*.

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

Fua, P. 1997. RADIUS: Image Understanding for Intelligence Imagery. In *Chapter Model-Based Optimization: An Approach to Fast, Accurate, and Consistent Site Modeling from Imagery*, M. Kaufmann, O. Firschein, and T.M. Strat, (Eds.).

Granger, S. and Pennec, X. 2002. Multi-scale em-icp: A fast and robust approach for surface registration. In *European Conference on Computer Vision*, Copenhagen, Denmark, pp. 418–432

Gumerov, N.A., Zandifar, A., Duraiswami, R., and Davis, L.S. 2004. Structure of Applicable Surfaces from Single Views. In *European Conference on Computer Vision*, Prague.

Hartley, R. and Zisserman, A. 2000. *Multiple View Geometry in Computer Vision*. Cambridge University Press.

Kass, M., Witkin, A., and Terzopoulos, D. 1988. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331.

Lepetit, V. and Fua, P. 2005. Monocular model-based 3d tracking of rigid objects: A survey. *Foundations and Trends in Computer Graphics and Vision*, 1(1):1–89.

Lepetit, V., Lagger, P., and Fua, P. 2005. Randomized trees for real-time keypoint recognition. In *Conference on Computer Vision and Pattern Recognition*, San Diego, CA.

Lin, W.-C. and Liu, Y. 2006, May. Tracking dynamic near-regular textures under occlusion and rapid movements. In *European Conference on Computer Vision*.

Ling, H. and Jacobs, D.W. 2005. Deformation invariant image matching. In *International Conference on Computer Vision*, Beijing, China, pp. 1466–1473.

Lowe, D.G. 2004. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 20(2):91–110.

Rosten, E. and Drummond, T. 2005. Fusing points and lines for high performance tracking. In *International Conference on Computer Vision*, Beijing, China.

Sclaroff, S. and Isidoro, J. 2003. Active blobs: region-based, deformable appearance models. *Computer Vision and Image Understanding*, 89(2–3).

White, R. and Forsyth, D.A. 2006. Retexturing single views using texture and shading. In *European Conference on Computer Vision*, volume LNCS 3954, pp. 70–81.

Wills, J. and Belongie, S. 2004. A feature-based approach for determining long range optical flow. In *European Conference on Computer Vision*, Prague, Czech Republic.