Real time non-rigid 3D surface tracking using particle filter

Ibai Leizea a, Hugo Álvarez 1, Diego Borro 1

a CEIT and Tecnun, University of Navarra, Paseo Manuel Lardizábal, 15, 20018 Donostia-San Sebastián, Spain

Article info

Article history:
Available online 17 December 2014

Keywords:
Real time tracking
Deformable tracking
Particle filter
Shape recovery

ABSTRACT

Recovering a deformable 3D surface from a single image is an ill-posed problem because of the depth ambiguities. The resolution to this ambiguity normally requires prior knowledge about the most probable deformations that the surface can support. Many methods that address this problem have been proposed in the literature. Some of them rely on physical properties, while others learn the principal deformations of the object or are based on a reference textured image. However, they present some limitations such as high computational cost or the lack of the possibility of recovering the 3D shape. As an alternative to existing solutions, this paper provides a novel approach that simultaneously recovers the non-rigid 3D shape and the camera pose in real time from a single image. This proposal relies on an efficient particle filter that performs an intelligent search of a database of deformations. We present an exhaustive Design of Experiments to obtain the optimal parametrization of the particle filter, as well as a set of results to demonstrate the visual quality and the performance of our approach.

1. Introduction

Recovering the 3D shape of a non-rigid surface from a monocular video sequence is a highly ambiguous problem because many 3D surfaces could have the same projection (see Fig. 1). Even when we have the intrinsic camera parameters and a well-textured surface, it is difficult to select the best mesh between all possible configurations of a deformable surface in order to solve the depth ambiguities. Along with the standard approaches that try to solve the shape recovery problem, we find approaches that establish prior knowledge [60], have a reference image [40], or are even based on a set of images of the target object [6].

Some methods rely on modelling the physical properties of a surface to achieve an approximation of the physical behaviour [14]. Although they obtain accurate results, the challenge is to determine the physical properties that govern the surface behaviour. Others adjust the behaviour of a surface to the movements registered in a database [43], which contains the most representative deformations of the surface. There are other approaches [29] that reconstruct the deformation of a non-rigid surface by using 2D-3D correspondences between an input image and a reference image in which the 3D shape of the surface and the intrinsic parameters of the camera are known. Likewise, some methods [4] that do not require prior knowledge have been proposed. They extract 2D-3D correspondences for each frame by tracking points over the video sequence.

The central idea of our method is to recover simultaneously and in real time the camera pose and the non-rigid 3D surface through the use of an efficient particle filter [19,37]. The particle filter performs an intelligent search in a database where a range of deformation templates and appearance descriptors are stored. Furthermore, two tracking methods are combined to offer robust tracking. The first one is based on appearance, which serves to do both the initialization and reset from a failure, while the second one is based on temporal coherence.

The rest of the paper is organized in 5 sections as follows. First, we provide an overview of some works related to recovering a deformable 3D surface and the justification for choosing our method is presented. In Section 3, we address all the steps that have been carried out to solve the problem of recovering a non-rigid 3D surface. Afterwards, in Section 4 a Design of Experiments is presented to determine the optimal values of the parameters of the Particle Filter, as well as, a test suite that demonstrates the good performance of our tracking method. Finally, conclusions and future works are enumerated in Section 5.

2. Related work

The non-rigid surface recovery problem has been addressed in several ways. Here we present a categorization of four main
groups: physics-based models, learning-based models, template-based models and non-rigid structure from motion.

The physics-based deformation models are the earliest approaches to modelling the behaviour of an object according to the physical laws that govern it. The key idea is to capture, in a generic way, prior knowledge about the physical properties of the object in order to achieve an approximation of the physical reality. They have been oriented to 2D shape recovery [20] as well as for 2D surface registration [35, 60, 54]. The 3D shape recover models, in turn, at first started with relatively simple reconstructions [51] and then, the complexity increased [14, 48]. Two main approaches appeared to solve the problem. One is the inclusion of regularization terms that minimize the global energy, which is composed of internal and external energies [14]. And an alternative approach is the finite element method (FEM) formulation that are used for accurate surgery simulations [34] and for large deformations in the animation area [18]. Subsequently, in conjunction with the previous formulation, modal analysis [11] was introduced to reduce the high dimensionality of the FEM formulation and to simplify the problem [32, 31]. Although the physics-based methods have led to great success and high levels of accuracy, the properties of the surface material are usually unknown. At the same time, this methodology is prone to instabilities in convergence, which are often difficult to optimize.

Learning-based models (also referred to as statistical-based models, machine learning models or example-based models) are a great alternative to the physics-based models, since they infer the surface behaviour from a set of available examples (hand labelled or generated through 3D laser scans). More precisely, these techniques apply a statistical dimension reduction technique to the training data in order to create a database of representative shapes, i.e., to learn a low-dimensional model. Active Appearance models are the first works to deal with these techniques, using both 2D [8, 59] and 3D [26] models. Following these models, Morphable Models [5] and recent approaches that promote the same philosophy of reducing the dimensionality have been proposed [43]. The main drawback of this formulation is that it imposes more restrictions on mesh deformations (linearity), so the representation cannot be as close to reality as one would like. Other approaches like [45] use non-linear statistical techniques, but based on a hierarchical strategy. They combine local deformation surfaces to generate a global shape that represents the shape. However, the biggest issue that the learning methods face is the generation of the training database. Additionally, training data is created specifically for each model, which implies that each object must be processed individually, even if two objects are composed of the same material.

The template-based models are well known methods that use a reference image (template) to solve the 3D shape reconstruction of non-rigid surfaces [29, 41, 33]. More specifically, the 3D shape of a surface in an input image is recovered from a set of visual correspondences between the input image and the reference image in which the 3D shape of the surface is known. These methods require the execution of two steps: image registration and shape inference. The first one is responsible for detecting 2D–3D correspondences between the input and reference images. Pixel-based methods, also known as direct methods, use intensity differences between two images to calculate the 2D–3D correspondences [9, 15, 16] and feature-based methods [52] attempt to detect the presence of the same feature in two images. Conversely, the shape inference phase tries to adjust the deformation model to correspondences according to a set of constraints [2, 41, 33]. However, few approaches [47, 28] deal the limitation of simultaneously obtaining the deformation and the camera pose. Furthermore, the main problems for the template-based methods are dealing with bad correspondences (which cause erroneous reconstructions) and running in real time obtaining the 3D structure.

In contrast to the template-based models, which require a reference image with a known 3D shape of the surface (which is not always easy to obtain), non-rigid structure from motion (NSFM) methods [57, 6, 4] are recent techniques that do not need prior knowledge to simultaneously track and recover the 3D shape of non-rigid 3D surfaces. These algorithms extract frame-to-frame 2D correspondences by tracking points over a video sequence, and try to recover the 3D locations in each input image. Since modelling a deformable surface as a mesh of vertex typically yields many degrees of freedom [1, 53, 23] represent the deformations as a linear combination of deformation modes. Other techniques, however, subdivide the problem through the motion of several rigid deformations [55, 50, 12]. These NSFM methods have certain weaknesses that are worth citing. First, these algorithms rely on tracking feature points throughout a video sequence, which makes a good textured surface necessary. Second, these methods have only demonstrated applicability when tracking smooth deformations of 3D surfaces [44], since more complex deformations require a greater number of modes which makes the system ambiguous.

Fig. 1. Ambiguities. There are ambiguities when recovering a non-rigid 3D surface. The same original image can recover different meshes.
learning-based methods is that an appropriate database that includes the most significant deformation modes of the surface is required, which generally involves a laborious manual process. In addition, despite the fact that many deformable techniques solve the problem with accurate results, in most cases they are costly to implement and have a high computational cost.

For all these reasons, we propose an approximation which is able to simultaneously reconstruct a non-rigid 3D shape and calculate the pose in real time. Furthermore, our proposal is fully automatic and does not require manual intervention. Additionally, our method is highly parallelizable, i.e., it supports efficient implementation, and although the surface variation is only for a limited degree of freedom, it works in an easier regime because it does not need to reach the same level of complexity as other approaches [42].

### 3. Simultaneous pose and non-rigid surface recovery

The central idea of our method is to recover in each frame both the pose and the non-rigid 3D shape of the target surface. In order to accomplish this, a 3D particle filter has been designed to continuously update the camera pose and recover the 3D surface. Our experimental studies in Section 4 have demonstrated that the proposed method can reliably recover 3D structures of surfaces plus the camera pose with low computational cost.

The framework of our system is divided in two phases: offline and online.

#### 3.1. Offline phase

The main goal of this phase is to build the two principal databases that will be used in the online phase. The first database, referred to as the appearance database, is used to perform the initialization step, i.e., to solve the first camera pose problem. This method is based on appearance, so it requires a database of feature descriptors that belongs to the training texture.

The second database, called the deformation database, is related to the frame-to-frame tracking method that is based on temporal coherence and provides a fast and robust tracking against non-strong camera movements. At this stage, a set of deformations that represents the behaviour of the surface is stored. This can be done by applying different types of deformations to the original state of the surface.

It is worth noting that this training phase is executed only once for each texture.

##### 3.1.1. Appearance database

The aim of this database is to store a set of reference features (with known 3D coordinates) together with their corresponding image descriptors (see Fig. 2), which is a well known procedure...
Reduces the computational cost in the online phase. Methodology) in order to build a parallelization method that divides the problem into smaller ones (following the divide and conquer approach) in order to subdivide the problem into smaller ones (following the divide and conquer approach). It is based on the behavior of a sheet of paper, so our surface model preserves the look and feel of the physical properties. In addition to applying deformations to the feature points, we also use a surface $M$ represented as a rectangular 3D regular mesh divided in several planar patches ($4 \times 4$) in order to subdivide the problem into smaller ones (following the divide and conquer methodology) in order to build a parallelization method that reduces the computational cost in the online phase.

In short, in order to recognize the simulated deformation that is closest to the current mesh in the online phase, a set of templates is stored in this database where each deformed mesh is expressed as the 3D position of the vertices. Below is a detailed summary of the steps that have to be followed to define it (Fig. 3).

3.1.2. Deformation database

The deformation database focuses on having a basic knowledge of the representative deformations that the 3D model can support. Accordingly, a range of deformations are applied to the reference points that lie on the target surface (see the previous section). More specifically, we rely on the proposal by [17], which presents a surface that produces a realistic page-turning effect by curling it. It is based on the behavior of a sheet of paper, so our surface model preserves the look and feel of the physical properties. In addition to applying deformations to the feature points, we also use a surface $M$ represented as a rectangular 3D regular mesh divided in several planar patches ($4 \times 4$) in order to subdivide the problem into smaller ones (following the divide and conquer methodology) in order to build a parallelization method that reduces the computational cost in the online phase.

In short, in order to recognize the simulated deformation that is closest to the current mesh in the online phase, a set of templates is stored in this database where each deformed mesh is expressed as the 3D position of the vertices. Below is a detailed summary of the steps that have to be followed to define it (Fig. 3).

Curling training. In this step some representative deformations of the original mesh are applied, i.e., deformation ranges are stored. Each deformation is classified as one of the following three types:

- **1-corner.** One corner of the sheet is folded. The surface is deformed by wrapping it around an imaginary cone model. This group includes those movements where a curl influences the whole side of the surface (Fig. 3 top left), like turning a page of a book. In particular, it is a process used in [22] to make a deformation and a rotation around an imaginary cone model. It also includes movements where the curl is a vertical movement that affects a small local area (Fig. 3 top left), i.e., a displacement is made from the corresponding corner to the center of the surface. This method is called peeling and it consists of a process that is similar to the cone model but in this case, the surface-turning point cannot be moved freely, i.e., the creased polygon is defined by a triangle.

- **2-corner.** Two adjacent corners are folded at the same time (Fig. 3 top middle). The surface is deformed by wrapping it around an imaginary cylindrical model.

- **4-corner.** The four corners of the sheet are folded at the same time (Fig. 3 top right). It also includes a cylindrical model to make the curling step and the wrapping generated on one side of the surface is applied equally on the opposite side.

Considering that a deformation degree is estimated by the maximum distance between the current position of the mesh (after applying a deformation) and its canonical location (at-rest position), the minimum and the maximum deformation degree for all types of deformation has been set to 0 and 60 degrees, respectively. These values rely on the fact that higher deformations are not
where to the remaining clusters; and \( \frac{1}{m} \) is the collection of distances from the template divided into several clusters according to the type of curling move-

tion (1-corner) with up and down deformation, 8 clusters with up and down deformation for each one of the second type (2-corners) and the last 4 clusters with up and down deformation for the last type (4-corners).

**Deformation profiles.** Since the behaviour of the implementa-
tion of the curling deformation that has been used is non-linear, an efficient mechanism has been created to simulate a linear response. It consists of generating deformation profiles for each cluster (Fig. 3 down), so that the deformation degree (the maximum distance between current and canonical position of the mesh – \( \max_{i \in \text{mesh}}|v_i - w_i| \) where \( v \) corresponds to the deformed mesh and \( w \) the canonical mesh-) is expressed as a percentage (see Fig. 4). This way, by knowing the deformation degree of the current mesh, in the online phase (see Section 3.2) we can infer the deformations that are in a range of proximity (expressed as a percentage) in order to always perform a search for the same range of deformation, i.e., avoiding the non-linear behaviours.

Furthermore, in an offline process we pre-calculate all the similarities between all the templates of one cluster against all the templates of the rest of clusters in order to obtain fast indexation in the online phase. Given a template from one cluster, this would help us to determine which template in another cluster is (see Eq. (1)) more similar. The main advantage of this technique is the cost reduction, which is obtained by means of queries to the database, since it is known where the current mesh can be situated. In this sense, the similarity factor between templates A and B is calculated by the distance of the vertices of each template.

\[
T_i^{c_j} = \left\{ d_i^{c_{1c_j}}, d_i^{c_{2c_j}}, \ldots, d_i^{c_{nc_j}} \right\}
\]

\[
d_i^{c_{1c_j}} = \sum_{k=1}^{n} \| v_i^{k} - v_i^{h_k} \|^2
\]

(1)

where \( T_i^{c_j} \) is the collection of distances from the template \( i \) of cluster \( c_j \) to the remaining clusters; and \( d_i^{c_{1c_j}} \) is the Euclidean distance of the \( n \) number of vertices (\( v \)) of the template (\( t \)) \( i \)-cluster \( l \) and template \( j \)-cluster \( m \), respectively.

**3.2. Online phase**

As mentioned in the literature, the main challenge of the non-rigid problems is not only calculating the pose that minimizes the reprojection error of the 2D–3D matches, but also the need to determine the best 3D point configuration, because many shapes can give the same projection result, i.e., the ambiguity problem must be solved. Therefore, this second phase detects the new shape of the mesh (the new 3D positions of the vertices) and calculates the correct pose that projects the corresponding 3D coordinates to the 2D cues detected in the input image.

As Fig. 5 depicts, this phase combines two different phases: initialization and frame-to-frame tracking. Both methods are combined to obtain a robust tracking. The frame-to-frame tracking is used as long as possible, and the initialization step is used in the first frame or to reset from the frame-to-frame tracking. It requires the surface at rest position, solving the problem as a rigid alignment and facing the visual tracking problem like other authors [36].

**3.2.1. Initialization**

The main goal of the initialization approach is the initialization of the data that will be used by the frame-to-frame tracking. Let us say that its purpose is to find a planar texture in the image, i.e., a rigid problem [24]. The computation of the camera pose well at the beginning and when an error occurs the surface is treated as if it were a rigid planar surface. For that purpose, an incremental tracking similar to the one presented in [56] is used. The input image goes through a detection process that consist of detecting a set of feature points (the FAST detector) and their corresponding descriptors (the simplified SIFT method, as is done in the offline phase (see Section 3.1)). Taking into account the precision and the computational cost, we propose the 4-4-31 SIFT parametrization for 640 \( \times \) 480 image resolution. The i-j-k parametrization corresponds to the number of regions (\( i \)), the number of histogram bins (\( j \)) and the patch size (\( k \)). Once the current image features and SIFT descriptors are computed, the matching process against the appearance database is based on the K-nearest approach. Accordingly, we compute the camera pose by using a robust hypothesize-and-verify PROSAC method [7], which returns the best homography. We consider this resulting matrix to be correct
if at least 25 percent of the matches are inliers. Moreover, to initialize frame-to-frame tracking (FTF) with the highest number of features, all the 3D points of the training database are projected with the new camera pose and associated to the closest detected feature point if the distance is below a predefined threshold.

3.2.2. Frame-to-frame tracking

The frame-to-frame tracking technique is used to recover the camera pose and the 3D surface through the motion property. The frame-to-frame tracking is divided into several steps: feature motion, deformation inference and feature recovery.

**Feature motion.** We use the Lucas–Kanade optical-flow method [25] to estimate the movement of the points that lie on the texture between two successive frames. More specifically, to reduce the computational cost, for each patch of the texture only the optical flow of the most significant points (those that have the strongest FAST response) are extracted. We infer the movement of the remaining points by applying the movement of the corresponding representative point. We also associate the new predicted locations to those points detected in the current input image (FAST detector) to avoid drift.

In order to avoid outlier points, we set affine constraints movements to the points of the surface. The study is made by using the patches fixed in the offline phase. Thus, the pose of each planar patch is formulated as a rigid problem. This method obtains an affine transformation that maps points on one plane (the planar patch of the previous frames points) to points on another plane (the plane formed by the new image features), and if the number of points is below a predefined threshold (15 points in our experiments), the patch is discarded. Then, RANSAC [13] is used to find the transformation that solves each local rigid problem, accepting only those patches that have 60 percent of inliers. Furthermore, the vertices shared between neighbouring patches should not exceed a threshold (in our case 10 pixels). As a consequence, we delete those points that do not fit well with respect to each of the neighbouring point. Finally, we say that the filtering step is good as long as at least 25 percent of the patches have a good pose.

**Surface deformation inference.** Once robust points have been calculated, a particle filter is used to recover the new pose and the new 3D mesh. Given the current 2D–3D correspondences and the previous pose, the particle filter updates the camera pose and the shape of the surface. It is noticeable that the particle filter selects fewer correspondences from the centre of the mesh (inner patches), as it is a correct way to approximate the real behaviour of the model with a low computational cost. This idea is supported by the results that are shown in the Experiments in Section 4.

The key idea of the particle filter is to detect a mesh that is stored in the database together with a pose that best resembles the current image. The functionality offered by the particle filter system as well as the way in which we apply for our particular case is detailed below.

**Particle filter.** Particle filters belong to the family of Sequential Monte Carlo (SMC) methods, and are robust against non-static scenes in which multi-modality is likely [37]. The main characteristic is that it represents the movements through a set of weighted samples known as particles. For the more detailed information about particle filters readers can refer to [3,39]. Furthermore, the use of particle filter covers many areas in the field of Computer Vision. One of the most noteworthy, is the head pose estimation and tracking. A more extended survey of the use of this type of techniques can be found in [30].
Particle filter is divided into two stages: particle generation and particle evaluation. In the particle generation step, the previous camera pose is perturbed to generate multiple camera pose candidate points of the current frame (particles). Thus, the posterior density of the motion parameters is approximated by a set of particles. Then, the particle evaluation step is responsible for assigning a weight to each particle and selecting the correct one.

In our particular case, just perturbing the pose is not enough. This practice is well suited for rigid bodies, but a deformable object requires that the vertices of the mesh as well as the pose be perturbed at the same time. Therefore, our particle generation step consists of perturbing the mesh (perturb shape) and the pose (perturb pose):

1. Perturb shape. To do this perturbation step (see Fig. 6), we set a filter for the deformation database in order to predict what the suitable deformations are. Instead of searching through the entire database (which would involve a very high computational cost), we make an intelligent search. Given the degree of deformation of the current template and pointing it in the right direction (we use the similarity factor (see Eq. (1)) to know which is the most similar template for each remaining clusters). It is noteworthy that a different deformation step (DefAll) is used here. Given the total number of particles to process (NumParticles) as input, the number of particles that are assigned to the same cluster is proportional to the degree of deformation. Thus, the greater the degree of deformation, the higher the number of particles for the same cluster, since in cases where the deformation is strong it is difficult to change to another type of deformation (another cluster).

2. Perturb pose. The pose perturbation is generated using the 3D meshes obtained in the shape perturbation step. However, we do not process all the particles from the former step, as the computational cost would be very high. We have grouped particles with a similar degree of deformation (groupDef) in each cluster (see Fig. 7). Thus, only the new pose for those particles that is representative for each group is calculated through an efficient PnP algorithm [21]. The poses of the remaining particles are copied from the closest representative particle.

After finishing the particle generation process, the particle evaluation phase weights each particle to subsequently select the best one. Assume that each particle is formed by the set of vertices that compose the 3D mesh, the camera pose and its associated weight. The likelihood \(w\) of each particle \(P_i\) is proportional to the percentage of inliers (those correspondences with a reprojection error lower than a predefined threshold: \(~9\) pixels for our experiments) (see Eq. (2)).

\[
w = \sum_{p \in \text{patches}} \frac{f_p}{N_p} \quad \text{and} \quad f_p = \sum_j \left( \frac{u_j}{v_j} - K R \hat{v} \right) < \text{thr}
\]

where \(f_p\) is the number of inliers related to patch \(p\) and \(N_p\) is the number of points in patch \(p\). An inlier is defined by \(f\) where \((u_j, v_j)\) are the 2D image position of the visual cue \(j\) and \(R\) and \(K\) are the corresponding 3D coordinates, \(R\) and \(K\) represent the intrinsic parameters that describe the characteristics of the camera (focal length, skew, etc.), which can be extracted through a camera calibration process.

The results achieved by our approach are satisfactory enough (Section 4), so the idea of introducing a new term to preserve the length of the edges has been rejected because it would increase the computational cost. Moreover, the templates generated in the offline phase offers a close approximation to the real physical properties of the model, so they impose restrictions inherently.

Finally, to set the inliers and refuse the outliers after the particle filter evaluation, the new 3D points are projected with the new pose and checked against the FAST points from the detection phase. If the distance to the closest FAST point is higher than \(~7\) pixels, the current point is set as outlier and the list of points for the next frame is updated.

**Feature recovery.** This last phase is called in the case where the number of current tracked points is below a threshold. Two major restore stages can be distinguished. The first one recovers points along the whole mesh (a high number of points are recovered), while the other one is more specific to each patch that compose the mesh (in this case only the points of a specific patch are recovered). The procedure is the same for both stages. It goes over the list of points detected in the initial state (tracking-by-detection step), and if a point has not been tracked yet, it
is projected with the new pose and it is verified that there is a FAST point close to it. If these conditions are met, then this point is added to the list of tracked points for the next frame.

4. Experiments and results

In this section we present the results of our proposal. Additionally, we show the Design of Experiments process that was used to obtain the optimal parametrization of our particle filter.

To perform the set of experiments we have used a sheet of paper with different types of textures, as Table 1 depicts.

4.1. Parametrization

We applied several points of the methodology presented in [49] to deduce what the optimal parametrization for our particle filter is. The main goal is to find a robust, efficient and real time particle filter.

First of all, in the design phase, we raised the problem that consists of knowing which variables influence both the error and the execution time of the particle filter. Thus, the error (inversely proportional to the weight of each particle) and the execution time were chosen as responses, i.e., variables that we wish to measure as the result of the evaluated process. The target was global param-
etritization that minimizes both responses. Furthermore, we used two textures (Table 1 top and middle) with a different number of feature points in order to attain general parametrization.

The second phase, measurement, consisted of identifying and classifying the variables that may influence the response. In this case we have identified the primary factors observed in Table 2 (granularity corresponds to the number of templates stored in each type of deformation or cluster deformation). Although these are factors whose cause-and-effect in the response is unknown, they are controllable factors, i.e., we can set ranges and levels for each one.

We defined 3 types of regions (Table 2) of levels in the same way as described in [10]:

1. Region of operation. The range in which theoretically the experiments can be done.
2. Region of interest. The range that we want to analyse.
3. Region of experimentation. Here the levels are fixed according to what we want to experiment. Those values that are so high and whose results are not significant at first glance are detected.

In the third step, pre-analysis, we selected the type of design to be set up. We decided to use a full factorial design, and therefore, the number of experiments is the product of the number of levels of each factor. Afterwards the experiments were executed (experiments stage) to determine which factors are statistically significant.

For the analysis, in a first step, the effect of each factor as well as the interactions between them were calculated. The former express the variation of the response caused by changes of levels of the input factors. While, the latter, corresponds to the impact whereby the apparent influence of a factor on the response depends on one or more factors. With all this information suggests which are statistically significant factors. Significant is called to that factor with a low probability of error does influence the response. To carry out this task, an Analysis of Variance (ANOVA) was used to see, at a given level of confidence, which effects influence the response. Thus, it is reported that an effect is considered as significant if its p-value is lower than 0.05.

The former two steps (experiments and analysis) are executed sequentially and in an iterative manner. Based on the range of interest previously assigned to each factor, local ranges of experiments are selected. Then, in view of the effects achieved of the factors through the p-values and related interactions between them, a new range is selected (either higher or lower depending on the response) for the significant factor. In addition, all the non-significant factors are excluded from the following analysis, setting for them the last assigned value.

Table 3 depicts the p-values extracted from the experiments for error and time. The ANOVA emphasizes that the most dominant factors are GroupDef and Granularity. NumParticles is also important, although the p-value associated with the error of the first texture is not fully reliable.

Finally, we introduced an optimization to get the parametrization for multiple responses, in this case for execution time and the error. It searches for a combination of the factors that jointly optimize the two responses by satisfying the requirements of each one.

We have determine the target value for execution time in 20 (close to real time if we consider additional cost associated to image processing) and 0 for error. With regard to the upper bound we have fixed with a high value for both responses. We have

Fig. 10. Guernica factors behaviour. Fixing the factors which are not considered as significant, the behaviour of the factor Granularity and NumParticles is studied.
selected 20 for the execution time case because this way we prioritize the error until this value.

After the calculation of each individual desirability (d), we combined them to get a measure of the composite. This measure of composite desirability (D) depends on the weight set for each response. In this case, both response will have the same weight.

The plot of the optimization (see Fig. 8) shows the effect of each factor on the two responses. The red vertical line determine the current desirable optimization settings according with the top displayed numbers. The horizontal blue lines and the corresponding numbers in the first column represent the value of the response for the current factor level. Finally, in each response the individual desirability (d) is shown, while optimal desirability (D) is shown in the top. If these values (d-error, d-execution, D-composite) are closer to 1 then the results that we want to achieve are better. Analysing the results, we can deduce that the best parametrization is 3500 for NumParticles, 5 for DefSame, 5 for DefAll, 1 for DefGroup and 14000 for Granularity.

Furthermore, the following graphs (see Figs. 9 and 10) shown for each of the textures and setting those values that are not currently consider to be significant, with the exception of the Group-Def that, in spite of being a significant factor, it was considered not to perform a study of it, the behaviour of the NumParticles and Granularity factors.

In the error case, looking at both textures graphs (Figs. 10 and 9 left), as the value of number of particles and the granularity increases the error values decrease. Sometimes it is perceived that error has ups and downs. This is due to the fact that the parametric configuration let introducing some particles in the particle filter while others do not. Therefore, it is possible to occur that the best particle that fits to the feature points is not being used. That is also why the interactions between different factors are given strategic importance, even the possible appearance of some noisy factors that cannot be controlled. Nevertheless, the overall performance is always descendent. It should be also be mentioned that the behaviour from 6000 number of particles and 8000 of granularity the error values are no longer decreasing because it achieves a saturation state where the number of particles and the granularity do not affect to the response. Similarly but in the reverse is the case at the time (Figs. 9 and 10 right). In the number particles the behav-
4.2. Experiments

This section presents the results obtained with the proposed method and its optimal parametrization. A hardware set-up that consisted of an Intel Core 2-Quad Q8550 at 2.83 GHz and 4 GB of RAM with a Logitech QuickCam Connect webcam was used to obtain the results.

4.2.1. Robustness

The first type of experiments consisted of applying a Gaussian noise with a normal distribution to the acquired matches. These matches are the ones that are provided to the particle filter, so this experiment lets us evaluate the robustness of the particle filter results. Therefore, three main tests were developed.

The first test consisted of introducing noise into the input data. More specifically, a Gaussian noise from 5 to 20 pixels of standard deviation was set to a percentage number of random correspondences that varies from 1 to 20. In analysing the results plotted in Fig. 11, we can say that our particle filter kept the reprojection error (in pixels) at reasonable levels despite the presence of some outliers. Thus, it can clearly be seen that once the noise exceeded 10 pixels the results start deteriorating. For example, with 15 and 20 of noise for the 15 percent of matches, the reprojection error increases.

Table 4

<table>
<thead>
<tr>
<th></th>
<th>Frame-to-frame tracking</th>
<th>Particle Filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core 2-Duo</td>
<td>48.81</td>
<td>33.62</td>
</tr>
<tr>
<td>Core 2-Quad</td>
<td>34.61</td>
<td>19.72</td>
</tr>
<tr>
<td>Core i7-930</td>
<td>30.94</td>
<td>17.23</td>
</tr>
</tbody>
</table>

Fig. 13. Results of 3D recovering surfaces. These are the three experiments that we have performed: Stones (top), Guernica (middle) and Salzmann (bottom). For each of the three examples, the following information is presented: the original image (first row), the projection of the recovered 3D mesh in that image (second row) and the recovered 3D mesh (third row).
errors are 5.11 and 7.72, receptively. Moreover, there are no more results presented when the noise is higher than 20 pixels because from there, the deformations are degraded as can be seen in the last example of Fig. 11. Nevertheless, although the results deteriorated when the number of noisy pixels exceeded 10, the projection was not bad at all and the particle filter correctly deduced the type of deformation is being applied.

The second test in turn, introduced noise to all correspondences but within reasonable limits. Thus, the Gaussian noise varies from 1 to 5 pixels of standard deviation. The graph Fig. 12a shows that the reprojection error (measured in pixels) increases provided the noises increases too. Even so, it should be noted that the results are sufficiently valid to obtain a correct visualization.

And finally, the last type of evaluation of the performance consisted of introducing small noise to the inliers while for the considered outlier correspondences (randomly selected as the first test) was added a very large noise, in this case 20 pixels of standard deviation. Thus, the experiment was to study the error for a set of correspondences randomly selected from the input data with a 20 pixels of noise for all of them. Fig. 12b indicates the results for each of the percentages of matches. Unlike the first test where the noise was only added to various matches, in this case the error is larger but did not differ a great deal from the first one. For instance, for the case where the number of selected matches was 15 and the noise 20, the error was about 7.72 while in the latter case reached 10.08 pixels of error.

Furthermore, it is equally important to highlight that the most affected points by the noise are those that suffer a deformation. In other words, the majority of the points that are located in the center of the texture, practically are not altered since can be considered as rigid problem due to little or no deformation.

4.2.2. Visual quality

Fig. 13 shows the correct visual output obtained by our approach for each of the three texture examples (Stones, Guernica and Salzmann). The Stones example represents the reconstruction of a deforming sheet of paper after a video sequence of 89 frames, while the Guernica and Salzmann examples performed the reconstruction after a video sequence of 51 and 71 frames, respectively. We selected these texture examples based on the uniformity of the features detected in each one. Stones is composed of a large number of features distributed in a uniformly way, while Guernica, apart from having fewer points, has a more irregular distribution. Salzmann, in turn, looks like Stones (the video sequence was provided by courtesy of [42] and it was used to compare the quality of our proposal against existing solutions) and just let us to check the visual behaviour of our method in an environment that is equal to that used by other works. It should be noted that the results of our proposal are as good as those results provided by other approaches.

4.2.3. Performance

The Particle Filter was also tested with different hardware configurations. Three different PCs were used: an Intel Core 2-Quad Q9550 at 2.83 GHz and with 4 GB of RAM, an Intel Core 2-Duo E8500 at 3.16 GHz and with 3 GB of RAM and an Intel Core i7-930 2.8 GHz and with 3 GB of RAM. Table 4 presents the times for the Particle Filter method and the frame-to-frame tracking step. The execution time of the frame-to-frame tracking includes both the visual detection and the particle filter method.

In analysing these values, we can infer that our method executes in real time and is highly parallelizable, as the better results obtained with better configurations demonstrate. Indeed, a future GPU implementation of our proposal will let us obtain lower computational times, and consequently improve other aspects such as more detailed databases or better image processing.

Finally, regarding initialization step, it is also worth mentioning that the computational cost for the described stage runs in average time from ~75 ms to ~150 ms depending on the selected hard

---

ware configuration. Since it is a single computation of the camera pose at any given time, it does not affect the performance of the whole process, so it guarantees the execution in real time.

4.2.4. Discussion

Since the range of deformations is limited, complex deformations may cause false positives (Fig. 14). If this happens, a simple shake will be enough to restart the recognition. As the majority of the non-rigid tracking solutions that are based on storing a set of deformations in a database, the main problem is to include a feasible set of deformations that best fit the physical behaviour of the model. Notice that the left example in Fig. 14 fits only to one of the two deformations that are being applied, so creating a more complex database, as the composition of various deformations, could solve this issue.

On the other hand, composition of deformations are not addressed. By the term composition we mean the combination of various types of parametrization of deformations at the same time in the same 3D structure. The main goal of our solution is oriented to handle simple deformations that do not achieve high degree of accuracy but instead correct visual feedback, i.e., achieve real time performance over accuracy sufficiently acceptable for the human eye. Incorporating composition of deformations will involve increasing the size of the database and, consequently, the computational cost would be higher as well as the reserved memory. In this sense, we shall seek an appropriate balance between the performance of the system and the size of the database. Moreover, the presented category of clusters is considered complete enough to cover a set of simple deformations to meet the requirements above raised.

Additionally, the selected design to create the database based on clusters is as a supervised manner. In other words, in such cases when the deformation is at very low level of deformation, the search is almost always performed along all the clusters giving similar probabilities. However, as the deformation achieves higher levels, is given higher priority to the same cluster over the rest ones. This assumption is based on the greater is the deformation degree the probability of being close to the same cluster makes

Fig. 16. 3D reconstruction from different points of view. Different side views of the camera for two different type of textures (Stones at the top and Guernica at the bottom). Medium (left) and extreme (right) side views are shown.
more sense. By this, when the deformation pattern (1-corner, 2- corners and 4-corners) begins to appear, the particles of the same cluster are more likely to be than the rest of the remaining clusters. This ensures that no abrupt changes appear with regard to the different parameterizations of deformations. Furthermore, the selected cluster structure together with the current function of similarity gives as a result in a correct visual feedback and a real time performance that is what we want to achieve.

Consequently, our framework is a complete solution that detects and tracks a texture in real time with a set of deformations that best fits to the logical behaviour of a sheet of paper.

 Likewise, it should point that extrapolating our methodology to other type of planar objects could be complex. Given that, we have performed some experiments with materials such as clothes. Even though false positives may appear (middle Fig. 14), sometimes the results are visually acceptable (on the right of Fig. 14, although a forward movement is being applied, the model fits backward). In other cases, though there are many deformation possibilities, the system returns satisfactory results (Fig. 15).

Finally, Fig. 16 shows the visual results from different viewpoints. Not only do we check the output 3D reconstruction of the mesh from a frontoparallel view of the camera, but also from more side-views. In this sense, as can be appreciated in Fig. 16, the side view varies from medium (left) to extreme (right). Moreover, the error in pixels varies from 4.68 to 12 for the Stones example and from 4.7 to 11.8 for the Guernica one in the case of medium and extreme side views, respectively. In conclusion, it is possible to check that when a deformation is applied from an extreme side view of the camera a degree of occlusion of the texture is caused and consequently the visual feedback is not as good as being in the frontoparallel case. Even so, it can be seen that the visual feedback is correct enough for the human eye.

5. Conclusions and future work

The problem of recovering the 3D shape of a deformable surface has become an important issue in different areas such as Computer Graphics and Computer Vision. In this paper we have presented a study of the different methods that we can find in the literature in order to solve the problem of the ambiguities that arise when reconstructing these types of surfaces. Furthermore, we have presented an alternative approach that is able to simultaneously calculate the 3D shape and the pose of a surface in an input image. The solution consists of creating one database for appearance descriptors and another one for deformation templates without any prior knowledge about the physical properties of the object. The novelty of our system relies on a particle filter that obtains real time execution by an efficient search through the deformation database to solve both problems simultaneously. Additionally, using a parallelizable technique like the particle filter, facilitates future improvements such as a GPU implementation.

In addition to this, a refined Design of Experiments has allowed us to deduce the optimal parametrization of a particle filter that achieves correct visual results in line with low computational cost. The robustness of our system has also been tested by adding noisy matches and comparing our system with other approaches.

The above contributions make us realize that new innovations and improvements are still possible in this field, and we will address them in the future. In terms of the database generation, which is a common problem in this type of techniques, it would be convenient to use this type of formulation in conjunction with a dimensional reduction technique such as PCA to retain the most significant modes of deformation. It would also be interesting to experiment not only with smooth movements but with strong ones where problems like the loss of information (that involve outliers) are dealt with. Following this idea, a possible improvement could be to combine our method with existing ones that offer more robustness but the computational cost is higher. Hence, considering the result of our system as an initial approximation of the deformed structure and its corresponding pose, a more robust system can feed with this information in order to achieve high levels of accuracy in real time performance.

On the other hand, although our frame-to-frame tracking stage is able to self-regulate to small errors (selecting the particle that best fits to the image features in each frame), our system has the limitation of resetting the process when the tracking is completely lost. It would be appropriate to insert for example, a tracking-by-detection step in order to estimate both the pose and the deformed 3D structure without the need of treating the surface as a rigid object to calculate the first camera pose. Furthermore, since our system is highly adaptable, the 3D reconstruction can be resolved not only with the reprojection error but also by adding other features such as colour recognition.

Appendix A. Supplementary material

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.cvIU.2014.12.002.

References
