Real-time Deformation, Registration and Tracking of Solids Based on Physical Simulation

Ibai Leizea∗ Hugo Álvarez‡ Iker Aguinaga† Diego Borro§

CEIT and Tecnun (University of Navarra), Spain

ABSTRACT

This paper proposes a novel approach to registering deformations of 3D non-rigid objects for Augmented Reality applications. Our prototype is able to handle different types of objects in real-time regardless of their geometry and appearance (with and without texture) with the support of an RGB-D camera. During an automatic offline stage, the model is processed in order to extract the data that serves as input for a physics-based simulation. Using its output, the deformations of the model are estimated by considering the simulated behaviour as a constraint. Furthermore, our framework incorporates a tracking method based on templates in order to detect the object in the scene and continuously update the camera pose without any user intervention. Therefore, it is a complete solution that extends from tracking to deformation formulation for either textured or untextured objects regardless of their geometrical shape. Our proposal focuses on providing a correct visual with a low computational cost. Experiments with real and synthetic data demonstrate the visual accuracy and the performance of our approach.

Index Terms:
I.4.8 [IMAGE PROCESSING AND COMPUTER VISION]: Scene Analysis—; I.6.8 [SIMULATION AND MODELING]: Types of Simulation—; I.2.10: ARTIFICIAL INTELLIGENCE— Vision and Scene Understanding

1 INTRODUCTION

Recovering the 3D shape of a non-rigid object is a highly ambiguous problem because many deformable objects (with completely different shapes) could have the same projection. In Computer Vision, there are few solutions that solve the registration of a non-rigid 3D object. Moreover, the number of approaches decreases when a real-time constraint is set. This is mainly caused by the complexity of the recognition and non-rigid registration steps.

This paper proposes a real-time non-rigid 3D object registration method that is able to track and register 3D models regardless of their geometric shape. Thus, our system can be used, for example, to provide visual feedback for applications such as assembly of flexible components in industry or medical surgery. Additionally, detection is not based on features, and therefore, we can operate with textured or untextured objects. This makes the approach robust against sudden illumination changes. The proposed method not only obtains the deformation, but also the camera pose for each frame. The detection and tracking system is based on templates (LINEMOD method [9]). Unlike other solutions, our approach does not require a static camera. In fact, the method handles large camera movements.

The process is divided into two steps. The first one takes advantage of an RGB-D camera to obtain both the colour and depth information. The second step, meanwhile, fits this information into a physical model. We rely on being able to model the mechanical properties of the solid to attain an approximation of the physical behaviour. More concretely, a Mass-Spring Model (MSM) is used to adjust the mesh and recover the new 3D shape due to the deformation. Finally, compared to existing solutions [7], our method offers accurate visual feedback (by fitting the model with different types of objects) and real-time performance over accuracy.

The rest of the paper is organized in 5 sections as follows. In Section 2 we provide an overview of some works related to recovering a deformable 3D shape and the justification for our contribution. Then, in Section 3 we address all the steps that have been carried out to solve the problem of recovering a non-rigid 3D object and detecting the camera pose that best fits the projection. Section 4 shows a test suite that demonstrates the good performance of our method. Finally, Section 5 presents the main conclusions drawn from this work.

2 RELATED WORK

Amongst the majority of approaches that recover the 3D structure of a non-rigid object, the problem is divided into two steps: image registration and shape inference. As far as image registration is concerned, the issue can be tackled by using two main approaches: feature-based methods or direct methods. Feature-based methods [1] focus on detecting points (also called features) that are distinguishable from their neighbourhood, and direct methods [4], in turn, use the intensity differences between two images to calculate the correspondences.

The shape inference process adjusts the acquired visual information to the shape. Physics-based systems adjust the shape according to its physical properties. Others [3], however, integrate geometrical or temporal constraints to the system and solve it through optimization techniques such as Second Order Cone Programming (SOCP). Similarly, there are approaches that learn the principal deformations of the objects to tackle the problem.

Image registration as well as shape inference solutions are combined in several ways in many of the approaches presented below.

In terms of feature-based vision methods, we can find examples in [13] and [14]. These works perform the reconstruction of non-rigid surfaces by learning the deformation-modes of the shape. They use Principal Component Analysis (PCA) to compute modes of deformation and generate a database of feasible shapes [2]. In [14], the 3D deformation modes are performed by varying angles between facets, while in [13], the mesh is subdivided into a set of patches that are combined linearly. In contrast, there are solutions that do not obtain the deformation-modes of the shape but still solve the visual part by using a feature strategy [17]. In this sense, local rigidity constraints [18] or unconstrained quadratic optimization [22] surface reconstruction can be carried out. However, the primary focus of many of these approaches [14] is on planar surfaces. Moreover, most of them have a high computational cost.

Following the idea of using visual cues, there are real-time approaches like [12], even using RGB-D cameras [8] and [19], which...
solve the registration problem. Nonetheless, they are focused on textured surfaces such as T-shirts. Similarly, approaches like [6, 7] work with stereo cameras and use a Finite Element Method (FEM) formulation in order to capture the behaviour of the object and calculate the deformation. These approaches, however, are very oriented to application in medical surgery, i.e., the formulation is specific, the position of the camera should be static, and in some cases, they are not executed in real-time [6]. Like the former approaches, [18] also uses a stereo camera to simplify the recovery process, but instead of a physics-based model, they use iterative processing [5] or mathematical tools like SOCP, which involve a complex optimization task to be carried out. The main drawback in most of the above cases lies in the fact that they have to overcome several challenges such as working with a limited set of feature points. In contrast to feature-based methods, direct-based or Analysis-by-Synthesis (AbS) methods, use the image data in combination with depth and colour information. In [10] the reconstruction of the model is based on a NURBS (non-uniform rational B-spline) deformations. Other approaches [15], in contrast to physics-based methods, use temporal constraints to minimize the error. Nevertheless, the main drawback of these methods is that they usually depend on a good initialization and need to make corrections over time in order to not be prone to drift.

2.1 Our Contribution

We propose a method that is able to track a 3D object and estimate its deformations in real-time. Compared to existing solutions, our method does not achieve the same accuracy level. In exchange, it returns correct visual results for the deformations of any kind of object (in terms of its geometrical shape). Moreover, our approach does not require the presence of textured surfaces. This paper presents, therefore, a complete framework for tracking and registering deformations of objects using a physics-based formulation.

For this purpose, we use an RGB-D camera that obtains both the colour and depth data, and avoids using other formulations that are not usually robust for texture-less surfaces or objects. In addition, the 3D registration and tracking of texture-less 3D objects is based on templates that exploit the use of the RGB-D camera.

This camera information, in turn, is used to feed the input of a physical model such as the Mass-Spring Model (MSM), which simulates the physical behaviour of the deformable object. Since the aim of the approach presented here is not to achieve an accurate deformation of the mesh, an MSM system has been chosen in order to obtain a visually correct solution that can be easily integrated with other AR technologies. Compared to other formulations such as FEM, MSM uses a simple structure and offers better performance (see Section 3). Furthermore, this physical model supports multiple deformation modes at one time. Besides, the information provided by the RGB-D camera has a lot of noise, with large holes, which provides an incorrect visual feedback, so this step is critical.

3 Proposed Method

The main objective of our method is the real-time recovery of the deformations of a non-rigid 3D shape and the simultaneous estimation of the camera pose. We adjust the geometry of the model, represented as a triangulated mesh, to the raw information acquired from an RGB-D camera. Furthermore, a physics-based method, as is the case of MSM, gives a realistic behaviour to the solid in order to obtain the deformations. Additionally, we take advantage of a template-based tracking method to continuously update the camera pose.

Figure 1 illustrates the framework, which is divided into two main phases: preprocessing and online execution. These two phases, in turn, are subdivided into two main stages. The first one focuses on the process of detecting and tracking rigid objects (presented in Section 3.1). On the other hand, the second stage retrieves the deformations of the objects (explained in Section 3.2). Therefore, each of these two steps generates prior information that is subsequently used in the online phase.

3.1 Tracking

The objective of this phase is to recognize and track 3D objects from a point cloud captured by a camera. The camera pose is obtained for each frame, just as is done with a rigid solid problem. This means that we rely on the LINEMOD template-based approach [9], as it guarantees the detection of texture-less objects in cluttered scenes. This method requires an RGB-D device and is divided into two major parts: learning and detecting the objects. In the first part, a range of templates of the object are trained and stored in a database alongside the 2D bounding box and their corresponding pose. The detection step in turn, generates a number of hypotheses and selects the template that best fits the input data from the database.

3.2 Object Deformation

3.2.1 Model Preprocessing

The model is preprocessed to generate the data structures required for its physical simulation. This procedure is divided into several steps: keypoint generation, voxelization and Mass-Spring Model initialization.

Keypoint Generation. Since we do not use features to calculate deformations, a list of keypoints is defined in order to relate the visual part with the 3D model. Moreover, these keypoints operate as control points in order to calculate the deformation in the following steps. They are uniformly distributed on the surface of the model (Fig. 2(c)). To carry out this task, a ray casting technique was developed. It traces rays from the bounding box to the centroid of the model and calculates the intersection points of the rays with the triangles that make up the mesh. The keypoints that are relatively close (based on a threshold) to the bounding box of the object are discarded to avoid noisy movements along the corners. Keypoints are used to calculate the correspondences between the vertices of the triangle mesh and the raw point cloud (section 3.2.2). Thus,
Mass-Spring Model (MSM) Initialization. As presented in steps: keypoint selection, correspondence matching and voxel disject. This stage belongs to the online execution and involves three which will be responsible for calculating the new shape of the ob-
frame. These matches will serve as input to the mass-spring model, current point cloud captured by the RGB-D camera for each input respondences between the offline keypoints (Section 3.2.1) and the
Once the camera pose is obtained in the tracking step (see Section 
3.2.2 Mesh Registration
3.2.2 Mesh Registration
Once the 2D position of the model’s bounding box is defined by the tracking method, its corresponding pose can be obtained from the trained template. However, the application of a delta transformation is neces-
sary to calculate the final pose (see Figure 3). First, a homog-
raphy transformation ($H$) is calculated between the reference bounding box and the detected bounding box in the image. In this way the resulting 3D delta transformation ($\Delta R$) can be
deduce from the stored reference 3D pose in order to match the projection of the reference bounding box to the currently detected bounding box.

Although an initial 3D pose is estimated, it is necessary to make a registration alignment between the input cloud and the reference model because it may be possible that they are in different scales. Thus, by projecting the 3D model keypoints with the former calculated pose, the corresponding 3D coordinates of the input frame can be obtained, establishing 3D-3D correspondences between the two point sets. We rely on [20] to estimate the transformation between the two point clouds, and thereby compute a more refined pose.

After the transformation is obtained, matching (finding corre-
spondences between the point cloud and the keypoints) pro-
cess begins. The main goal here is to find the associations between the visible keypoints with the points that have been deformed in the point cloud. Therefore, an intelligent search structured as a scale search is performed to achieve a low computational cost (See Fig. 4). This search is divided into two main tests. The first one is used to discard the keypoints that do not have any deformation, and the second one, in turn, serves to make the association between the keypoint and a sample of the input point cloud.

In order to carry out this search, two search areas are defined for all keypoints. These areas are represented by two OBBs
3.2.3 Mesh Physical Simulation

The MSM model is dynamically simulated, integrating the motion equations of the model in time using an explicit Verlet integrator. Once the displacement of the nodes of the voxel structure has been obtained from the previous step, the system can compute the elastic forces exerted by each spring on the nodes, i.e., the physical deformation of the voxels. This process can be done in parallel for each spring, accumulating the resulting forces for each node. The Verlet integrator is conditionally stable. The integration time step is limited by the stiffness of the system, with stiffer materials requiring shorter time steps for the simulation to be stable. In our case the deformation is applied to objects of low rigidity such as foams, that can be easily deformed with the hand. The low stiffness allows a time step value high enough to guarantee a real-time performance of the simulation. Moreover, this simulation is performed a number of iterations per each frame until the deformation converges to a stable configuration.

Figure 3: The reference pose is obtained by calculating a delta transformation between the reference bounding box and the detected one.

Figure 4: Correspondence matching. Once the search areas are defined (a), two test are applied in order to relate the keypoints to the point cloud: OBB test-1 (c) and OBB test-2 (d).

Voxel Vertex Displacement. Finally, the displacement of the vertices of the voxels must be calculated in order to extrapolate the nodes of the physical structure (See Fig. 2 (c)). This process consists of detecting the voxel that the keypoint belongs to and applying, in a weighted manner, the displacement to the vertices of the nearest face of the voxel. This weight is represented by an interpolation between the keypoint and the vertices of the voxel it belongs to. Following this, we obtain all the displacements, i.e., the total movements of each node of the mesh.

3.2.4 Model Fitting

The position of the nodes of the voxels obtained in the previous step let us determine the new position of all the vertices of the triangle mesh (Fig. 2 (i)). The vertices of the triangle mesh are interpolated with regards to the 8 nodes of the containing voxel. For that purpose, in a preprocessing step we calculate the isoparametric coordinates of each vertex. These coordinates relate the mesh represented by the voxels and the triangle mesh. The isoparametric coordinates represent a mapping from local coordinates to a general coordinate system. Therefore, given a point in the isoparametric coordinates we can obtain the corresponding global coordinates of the point using the mapping equation. Knowing this, we define the interpolation function with tree coordinates $\xi$, $\eta$ and $\zeta$ with a range between -1 to +1, i.e., we define this local system like a 2-side cube, using trilinear interpolation. See [21] for more details about the formulation.

4 EXPERIMENTS

In this section we present the results of our proposal. A set of experiments that evaluate performance (primarily the error estimation and execution times), robustness against noisy data and the adaptability of the method are shown for some geometrically shaped objects. The hardware setup consists of an Intel Core 2-Quad Q9550 at 2.83GHz and 4 GB of RAM equipped with a Kinect XBOX 360. For this purpose, 4 different models have been used to test the performance of our framework: Teddy Bear, Sponge Bob, Ball and Cushion. The representation of Teddy Bear can be defined as a highly complex shape with an homogeneous texture (texture-less). Sponge Bob has a simple geometrical shape since the limbs of the object are discarded and Cushion is along with the Sponge Bob model the simplest geometric shape. Both the Sponge Bob and the Cushion models, consist of box-like shapes (though Cushion is simpler than Sponge Bob) with different kind of textures (irregular texture for Sponge Bob case and regular texture for Cushion). Finally, the Ball example provides a sphere geometric shape.

4.1 Performance

4.1.1 Computational Time

Figure 5 presents the execution times of the physics registration modules for the four models, detailing the computational cost of each step. As we can see, the deformation framework can obtain the non-rigid transformation of the shape in between 8.3 and 12.65 ms, depending on the model complexity.

In terms of the physics simulation module, the times of the mesh fitness step, according to the number of iterations of integration that the system needs to achieve good physical behaviour, increases a 10% from 5 to 50 iterations (from 1.93 to 3.12 ms for 5 iterations, and from 19.01 to 30.01 ms for 50 iterations). However, the mean error does not vary significantly (from 6.19 to 9.31 mm for 5 iterations, and from 5.3 to 6.15 mm for 50 iterations). Therefore,
we have not considered it necessary to use a large number of iterations in order to achieve acceptable visual results, as observed in Figure 6.

It is also worth mentioning that the tracking algorithm runs in an average time of \( \sim 270 \) ms for the detection step, while the frame-to-frame tracking step varies from 60 to 114 ms due to the density of the point cloud for each of the four models. Although these two modules are not real-time, they are not our main contribution, so these two modules can be replaced by other approaches. However, in order to provide a complete solution, this proposal is well suited to our framework because it uses a template-based methodology instead of using features (not required in our physical simulation step).

### 4.1.2 Accuracy level

Error estimation was also performed in order to check the response of our method. Taking into account the results described above, we chose a configuration of 300 keypoints (Sec. 3.2.2), while the physical simulation step ran in 5 iterations. In addition, we used a set of synthetic data to evaluate the accuracy of the registration. We built a set of 25 random deformations of the mesh for each of the 4 different models and the accuracy of each deformation was calculated from 9 different points of view. Each point of view sets the visible keypoints, and consequently, the result of the deformation can be different. Thus, the mean errors (in mm) from the ground truth vertices mesh and the resulting mesh are 9.31 (Teddy Bear), 4.05 (Sponge Bob), 6.61 (Ball) and 6.19 (Cushion), with standard deviations that vary from 1.38 to 2.78 mm. These results demonstrate the accuracy of the system for different types of objects and camera points of view. The reason why there are error differences between the models lies in the fact that the geometric shapes are different. Even so, the errors for every case remain low enough and the visual quality of the reconstructions is good.

### 4.2 Robustness

This second experiment shows the limits of the proposed method in terms of robustness when applying some noise, i.e., while distorting the movement of the points. It consists of applying a set of transformations to the input cloud in order to disorder it and then testing the behaviour of the method. The transformations applied to the Teddy Bear model were divided into two main types: translations and rotations, along the three axes \( x \), \( y \) and \( z \).

For translations, movements from 0 to 30 mm were executed with a step of 2.5 along the three axes. Resulting errors varied from 0.2 to 0.5 mm in case of a 20 mm displacement. In the rotation transformation, in turn, we used a strategy of rotating the point cloud from 0 to 15 degrees. Resulting errors varied from 0.2 to 0.8 mm in the case of 15 degrees. Analysing the results, we can conclude that our method maintains the error at reasonable levels despite the presence of noise in the data and that it correctly deduces the deformation being applied.

### 4.3 Adaptability

Figure 6 shows the correct visual output obtained by the proposed approach and stresses the adaptability to many different types of geometrical shapes. The Teddy Bear example represents the reconstruction of a complex deforming shape after a video sequence of 490 frames. Sponge Bob is a 325 frame video sequence of a squared model with a rich texture (that is not used). In the same way, Cushion, is a 565 frame video sequence reconstruction of a parallelepiped object with texture. Finally, the Ball example, is a 410 frame sequence that consists of a curvilinear and texture-less object. It should be noted that in all of these examples the results are sufficiently valid to obtain a correct visualization and develop Augmented Reality applications.

### 4.4 Discussion

Our system might loose the camera pose if large occlusions are applied (as with non-textured tracking systems). Nonetheless our approach can handle deformations with small occlusions (for example tools manipulating an object [7]).

Furthermore, it has to be emphasized that the modularity of the proposed pipeline allows the incorporation or replacement of certain modules. Thus, the performance of the framework can be improved using, for example, new tracking methods [11] that fit with the properties of our algorithm. Similarly, combining the proposed approach with a physical model that includes a rigid skeleton local deformations and junctions movements can also be computed.

### 5 Conclusions and Future Work

In contrast to the existing solutions, we propose an approach to carry out deformable registration in real-time for non-rigid 3D object. The system relies on a physics-based model simulation that lets us preserve the physical behavior of the solid. The proposed system uses an RGB-D camera that, together with the physical simulation, enables the deduction of the new transformation of the 3D triangles of the mesh without the need of well-textured objects. In addition, our framework includes a tracking module based on templates to deal with scenes that lack texture and to retrieve the camera pose. Thus, the object can be moved while its deformations are calculated. Additionally, a set of experiments has allowed us to verify the quality of the results, demonstrating a correct visual representation of the solid in line with a low computational cost.

It should be noted that a methodology that ensures correct matching and reduces the computational cost is an issue that will be addressed by the authors in the future.

### References


Figure 6: Visual results of 3D recovery shapes: Teddy Bear (a), Sponge Bob (b), Ball (c) and Cushion (d). For each of the four examples: the model, the original image with the projection of the recovered 3D mesh (first row), the recovered 3D mesh in terms of the physics model (second row) and the same recovered mesh with the raw data acquired from the depth camera (third row).