3D reconstruction of deformable objects from RGB-D cameras: an omnidirectional inward-facing multi-camera system

Eva Curto[1][a], Helder Araujo [1][b]
[1]*Institute for Systems and Robotics, University of Coimbra, R. Silvio Lima, Coimbra, Portugal*
{*evacurto, helder*}*@isr.uc.pt*

Keywords: Reconstruction, RGB-D, Omnidirectional, Multi-camera, Deformations

Abstract: This is a paper describing a system made up of several inward-facing cameras able to perform reconstruction of deformable objects through synchronous acquisition of RGBD data. The configuration of the camera system allows the acquisition of 3D omnidirectional images of the objects. The paper describes the structure of the system as well as an approach for the extrinsic calibration, which allows the estimation of the coordinate transformations between the cameras. Reconstruction results are also presented.

1 INTRODUCTION

In this paper a system for the 3D reconstruction of deformable objects is described. The system is made up of several inward-facing cameras to allow for the acquisition of the whole surface of the object. The system performs the synchronous and time-stamped acquisition of RGB-D images enabling the synchronous acquisition of 3D images of deformations. Therefore, the main contribution of this work is the assembly of the system itself, putting together and setting up the appropriate hardware and software.

The first algorithms for RGB-D-based dense 3D geometry reconstruction were developed only for static scenes. (Curless and Levoy, 1996) introduced the fundamental work of volumetric fusion and inspired the most modern approaches. The ability to provide real-time RGB-D reconstruction appears in 2002 with a system based on a 60 Hz structured-light rangefinder (Rusinkiewicz et al., 2002). Although it is no longer a recent algorithm, KinectFusion (Newcombe et al., 2011) had a significant impact on the computer graphics and vision communities. This work was the basis for many new methods of 3D reconstruction of static and dynamic scenes. They proposed the fusion of all data streamed from a Kinect sensor into a single global implicit surface model of the observed (static) scene in real-time. The current sensor pose is simultaneously obtained by tracking the live depth frame relative to the global model using a coarse-to-fine Iterative Closest Point (ICP) algorithm, which uses all the observed depth data available. The real-time system of (Whelan et al., 2016) is capable of capturing comprehensive dense globally consistent surfel-based maps of room scale environments. The online BundleFusion approach of (Dai et al., 2017) allows a robust pose estimation, optimizing per frame for a global set of camera poses by considering the complete history of RGB-D input with an efficient hierarchical method.

The first approach to handle online deformable tracking of arbitrary general deforming objects was the template-based method presented by (Zollhöfer et al., 2014). In VolumeDeform (Innmann et al., 2016), they propose using sparse RGB feature matching to improve tracking robustness and handle scenes with a little geometric variation. Besides, they propose an alternative representation for the deformation warp field. Unlike DynamicFusion (Newcombe et al., 2015), VolumeDeform uses the same volumetric model to represent the reconstructed space. The previous methods, (Newcombe et al., 2015), (Innmann et al., 2016), achieved excellent results. However, they have a few limitations. The intermittent conversion from Signed Distance Field (SDF) to mesh for correspondence estimation leads to loss of accuracy, computational speed, and the capability to capture topological changes conveniently. Additionally, both require 6D motion to be estimated per grid point, while a 3D flow field is sufficient in Miroslava et al. (Slavcheva et al., 2017) method - KillingFusion - due to the dense smooth nature of the SDF representation and the use of alignment constraints directly over the

[a] https://orcid.org/0000-0002-0477-0091
[b] https://orcid.org/0000-0002-9544-424X

field. Therefore, KillingFusion provides a non-rigid reconstruction pipeline, based on a single data representation – SDF, which does not require explicit correspondences and can handle topological changes.

The previous method employs a combination of two regularizers, which are challenging to balance and thus result in over-smoothing and loss of high-frequency details. SobolevFusion (Slavcheva et al., 2018) proposes to define the gradient flow in Sobolev space H^{-1} instead of a gradient flow based on an L^2 inner product, which is known to be susceptible to local minima.

This section introduced the work discussed in this paper as well as referred some state-of-art techniques in 3D reconstruction. This paper has the following structure: Section 2 describes the camera's system and the surrounding setup and explains how the cameras were synchronized by hardware. The third section explains the method of extrinsic calibration used to obtain the relative position and orientation. Then, in Section 4, we have the reconstruction phase described and illustrated with reconstructed objects. The final considerations of this paper are stated in section 5.

2 SYSTEM DESCRIPTION

2.1 Cameras

Our camera system is composed of four Intel Realsense D415 cameras. The choice of these cameras took into account several factors:

- Low cost;
- The D415 come with Intel's RealSense SDK 2.0, which is an open-source, cross-platform SDK;
- Their field of view is well suited for high accuracy applications such as 3D scanning;
- The rolling shutter on the depth sensor allows us to have highest depth quality per degree.
- Theses cameras can all be hardware synchronized to capture at identical times and frame rates.

Considering that this work proposes the omnidirectional reconstruction of objects, these specifications are satisfactory for the applications envisaged. The D415, showed in Figure 1, has two main components, the vision processor and the depth module. The vision processor D4 is either on the host processor motherboard or on a discrete board with either USB3.0 Gen1 or MIPI connection to the host processor. The depth module includes left and right imagers for stereo vision with the optional IR projector and

Figure 1: Image of a RealSense D415 camera.

RGB color sensor.
The essential specifications of the D415 camera are indicated in Table 1.

Table 1: Specifications of Intel RealSense D415.

Features	Use Environment: Indoor/Outdoor
	Image Sensor Technology: Rolling Shutter, 1.4μm $\times$ 1.4μm pixel size
	Maximum Range: Approx. 10 meters.
Depth	Depth Technology: Active IR Stereo
	Minimum Depth Distance (Min-Z): 0.16m
	Depth Field of View (FOV): 65°±2 ° × 40°±1 ° × 72°±2°
	Depth Output Resolution: Up to 1280 × 720
	Depth Frame Rate: Up to 90 fps
RGB	RGB Sensor Resolution: 1920 × 1080
	RGB Sensor FOV (H x V x D): 69.4° × 42.5° × 77° (±3°)
	RGB Frame Rate: 30 fps

2.2 Experimental setup

Each camera is mounted on a C clamp camera support, which is fixed to the table. The cameras are equally distant between neighboring cameras since we intend to maximize the horizontal fields of view. The objects that will be reconstructed are illuminated by led light in addition to natural light. In order to avoid problems with specular reflections that could induce more noise in RGBD acquisition and consequently conduce to poor reconstruction results, we opted to use a metallic table painted with a matte black color. Matte allows avoiding specular reflections while metallic allows for the absorption of the IR energy.

The synchronous acquisition of RGB-D images from multiple cameras (four in the case) requires a host system with enough processing power to read from the USB ports streaming the high-bandwidth data, and doing some amount of the real-time post-processing, rendering, and analysis. All the work from the data acquisition, calibration to the reconstruction was made on a PC. This was a processor with a Intel Core i9-9900k CPU @ 3.60GHz x 16, a GeForce RTX 2070/PCle/SSE2, running Ubuntu 16.06 LTS.

The setup described in this section is shown in Figure 2.

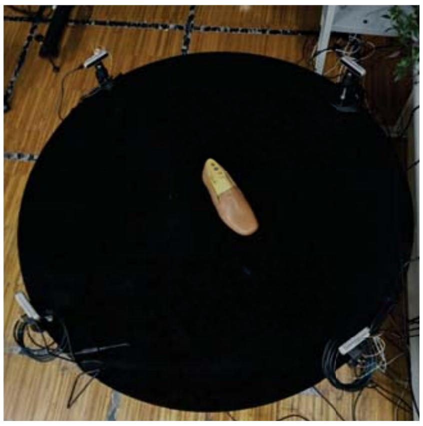

Figure 2: Picture of the omnidirectional camera system.

2.3 Hardware Synchronization

One of the requirements for our setup is the synchronization between cameras, since it should also perform the estimation of 3D deformations.

Hardware synchronization is described in (Grunnet-Jepsen et al., 2018), from Intel. Following this reference, we connected the cameras via synchronization cables, considering three of the cameras as slaves and the fourth one as master. For each camera, depth and color frames are saved as well as the metadata. For each frame the following information is saved: the serial number of the camera, the type of stream (color or depth), the frame timestamp, the sensor timestamp, the actual exposure, the gain level, the boolean value of auto-exposure, the time of arrival, the backend timestamp and the actual fps.

Using a hub to connect the four cameras to the PC, the higher resolution that we achieve with the hardware synchronization and all the color and depth streams activated was 640×360. Thus, all the acquisitions were made with this resolution setting.

3 Extrinsic Camera Calibration

This section begins with a brief description of the extrinsic calibration. Then the multi-camera method used is described.

3.1 Theory

Considering m cameras and n object points $\tilde{\boldsymbol{X}}_j = [X_j, Y_j, Z_j, 1]^T, j = 1, ..., n$. We assume the pinhole-camera model. The 3D points $\tilde{\boldsymbol{X}}_j$ are projected to 2D image points $\tilde{\boldsymbol{x}}_j^i$ as

$$\lambda_j^i \begin{bmatrix} u_j^i \\ v_j^i \\ 1 \end{bmatrix} = \lambda_j^i \tilde{\boldsymbol{x}}_j^i = P^i \tilde{\boldsymbol{X}}_j, \lambda_j^i \in R^+ \tag{1}$$

where u, v are pixel coordinates, λ_j^i the scale factors and P^i is the projection matrix of a given camera. This 3 x 4 matrix has 11 degrees of freedom. This projection matrix can be further decomposed as:

$$P^i = K^i [R^i \; t^i], \tag{2}$$

where K^i is the matrix of the intrinsic parameters, R^i is the rotation matrix relative to the world coordinate system and t^i is the translation vector relative to the world coordinate system.

We aim at estimating the relative positions and orientations between the reference coordinate systems of the depth cameras. The relative positions and orientations are described by 6 parameters, being 3 for rotation and 3 for translation. These are the external/extrinsic parameters. In the case of this setup the intrinsic parameters as well as the extrinsic parameters between the cameras of each stereo pair and the rgb camera and depth cameras are obtained from the RealSense SDK. Then, the goal of the calibration is to estimate the relative rotations and translations between the four different depth cameras (each of which uses a coordinate system attached to the left camera of each pair).

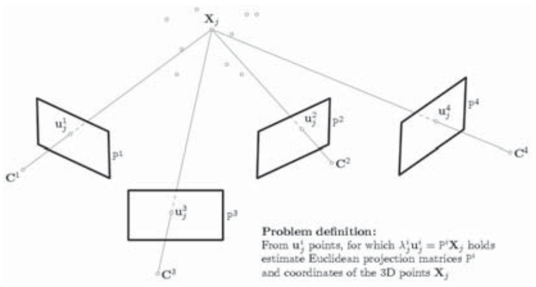
$\mathbf{X}_j$
$\mathbf{u}_j^1$
P^1
$\mathbf{C}^1$
$\mathbf{u}_j^2$
P^2
$\mathbf{C}^2$
$\mathbf{u}_j^3$
P^3
$\mathbf{C}^3$
$\mathbf{u}_j^4$
P^4
$\mathbf{C}^4$
Problem definition:
From $\mathbf{u}_j^i$ points, for which $\lambda_j^i \mathbf{u}_j^i = P^i \mathbf{X}_j$ holds
estimate Euclidean projection matrices P^i
and coordinates of the 3D points $\mathbf{X}_j$

Figure 3: Multi-camera calibration problem (Svoboda et al., 2005)

Figure 3 shows a schematic diagram of a four-camera setup.

3.2 Overview of the multi-camera method

The point clouds obtained by each depth camera are expressed in their coordinate system. To obtain the relative transformation we based our approach on the method described in (Matsumoto and Aguilar-Rivera, 2018). This multi-camera calibration method is, on the other hand, based on the approach described in (Svoboda et al., 2005), where a small and easily detectable bright spot is used to create a virtual calibration object. This bright spot is simultaneously visible in all cameras avoiding the occurrence of occlusion. Since the cameras are synchronized, the user has only to wave the light through the working volume, therefore generating the required data. The remaining calibration procedure is fully automatic. The bright spot projections are detected independently in each RGB camera, so the correspondences are established by the time stamps. Each detected point is also mapped into the depth image. Before starting to generate the 3D trajectory for calibration, the environment illumination is dimmed and the user can adjust the brightness threshold for pointer detection and tracking. As a result, the detection of the light spot both in the RGB image and in the infrared images (depth) is facilitated and robust. The pointer location in the RGB image is converted to the corresponding location in the depth image. From the depth image, the 3D position of the pointer (relative to the camera) was estimated. Figure 4 illustrates the four 3D trajectories, each one of them regarding one specific camera.

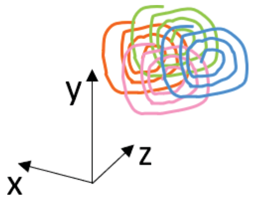

Figure 4: Illustration of 3D trajectories for extrinsic calibration.

These trajectories are then filtered to remove outliers. The resulting filtered 3D points of each trajectory are then used to estimate the relative orientations and translations between cameras.

3.3 Finding optimal rotation and translation between corresponding 3D points

The optimal rigid 3D registration problem can be characterized, according to (Arun et al., 1987) with:

$$RA + t = B, \tag{3}$$

for noise-free data. Since the data is noisy, the least-squares error is minimized by:

$$err = \sum_{i=1}^{N} ||RA^i + t - B^i||^2, \tag{4}$$

where A and B are sets of 3D points with known correspondences. R is a 3×3 rotation matrix and t is the 3×1 translation vector.

To estimate the optimal rigid transformation, both point clouds are centered at the origins of their coordinate systems. Therefore, the centroids of both datasets are first estimated:

$$centroid_A = \frac{1}{N}\sum_{i}^{N} A^i, \tag{5}$$

$$centroid_B = \frac{1}{N}\sum_{i}^{N} A^i, \tag{6}$$

where A^i and B^i are 3×1 vectors, corresponding to the point pair i, with the coordinates of the 3D points, i.e., $[X,Y,Z]^T$.

To find the optimal rotation, we first re-center both datasets so that both centroids are at origin. This removes the translation component, leaving only the rotation to estimate. The rotation is estimated using the SVD method by Arun, performed on the point-set cross-covariance matrix given by (Kanatani, 1994):

$$H = (A - centroid_A)(B - centroid_B)^T, \tag{7}$$

$$[U,S,V] = SVD(H), \tag{8}$$

$$R = VU^T, \tag{9}$$

where H is the point-set cross-covariance matrix and $A - centroid_A$ is an operation that subtracts each column in A by $centroid_A$. When finding the rotation matrix, we have to take into account the case of the reflection matrix. That is, sometimes, the SVD method returns this reflection matrix, which is numerically correct but is nonsense. This is addressed by checking the determinant of R and checking if it is negative (-1). If it is, then the 3rd column of V is multiplied by -1.

After the rotation matrix is found, we estimate t using the initial equation (3) $RA + t = B$ but using the centroids:

$$R \times centroid_A + t = centroid_B, \tag{10}$$

$$t = centroid_B - R \times centroid_A. \tag{11}$$

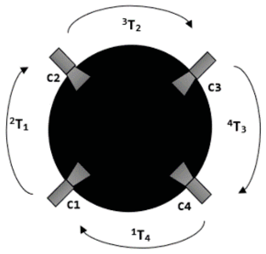

Figure 5: Schematic diagram representing the transformations between cameras.

3.4 Evaluation of the Calibration

Using the data acquired as previously described, the estimation of the relative rotation matrices and relative translation vectors can be performed. Since we are dealing with an omnidirectional system (Figure 5), a relative simple criterion can be applied to estimate the overall estimation error. Assume that $T_j^i = \begin{bmatrix} R_j^i & t_j^i \\ 0_{1\times3} & 1 \end{bmatrix}$ represents the coordinate transformation from coordinate system i to coordinate system j. Then, in the specific case of four coordinate systems the following condition holds:

$$T_1^2.T_2^3.T_3^4.T_4^1 = I_{4\times4} \tag{12}$$

This condition can be used to obtain an estimation of the overall error in the four coordinate transformations. In general the errors obtained are small, with the overall translation error smaller than 5% of the distance between consecutive images. Errors in each coordinate transformation can be minimized by using the above error criterion in a global optimization procedure.

4 Reconstruction

The reconstruction of a deformable object is possible since we have a synchronous acquisition system of RGBD data and the relative positions and orientations of the cameras are known. These transformations allow us to combine the four point clouds. The coordinate system of one of the cameras is used as a reference coordinate system.

Since the visual fields of adjacent cameras overlap, duplicated points occur in the omnidirectional point cloud. This can lead to a non-homgeneous reconstruction. To overcome this issue, the omnidirectional merged point cloud is filtered using the VoxelGrid filter from PCL library. The VoxelGrid filter downsamples the point cloud by taking a spatial average of the points in the cloud confined by each voxel. The set of points which lie within the bounds of a voxel are assigned to that voxel and are statistically combined into one output point.

In this paper, diverse examples of object reconstruction are presented. Firstly, we analyze the reconstructions of a small wooden box and of a shoe with a mold—two different objects in terms of shape, material and color. Then, we reconstruct two white polystyrene spheres with different radius. For the reconstruction of the spheres, in an attempt to mitigate the noise involving the object, three acquisitions of each camera were used to generate a mean point cloud for the respective camera. The four mean point clouds (corresponding to the four cameras) are then transformed and filtered in one merged point cloud, similar to the previous reconstructions. Finally, the reconstruction of a deformable object is presented, a hippo balloon, in different stages of emptying.

4.1 Reconstruction of a wooden box and a shoe

The omnidirectional system synchronously acquired RGB-D data viewed by each camera pointed at the box. The different views taken at the same timestamp of the wooden box are shown in Figure 6.

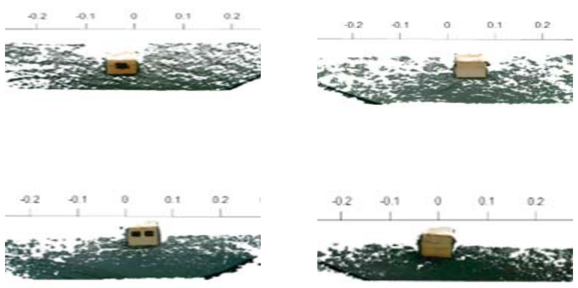

Figure 6: The four different views of the wooden box.

Figures 7 and 8 show the reconstruction of the wooden box in different perspectives. We can notice some noise around the corners of the box and also the lack of sharpness of the upper face.

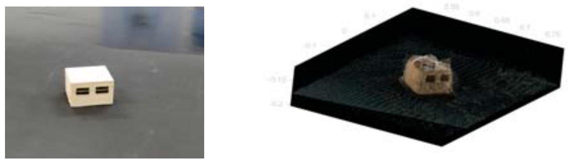

Figure 7: Real box on the left and reconstructed box on the right: first perspective view.

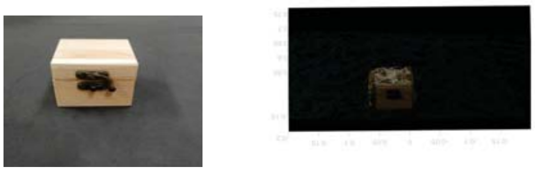
Figure 8: Real box on the left and reconstructed box on the right: second perspective view.

The shoe, unlike the box, is very curved and made of a much brighter material. The different views of the shoe taken at the same timestamp are in Figure 9.

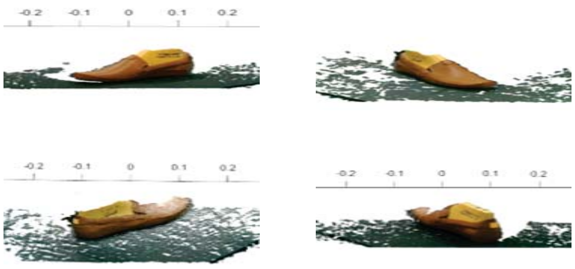
Figure 9: The four different views of the shoe.

In the shoe reconstructions, presented in Figures 10 and 11, it is possible to observe that, since it has no corners, there is not as much noise as in the case of the box. On the other hand, we have some holes, well visible in Figure 11, resulting from specular reflections.

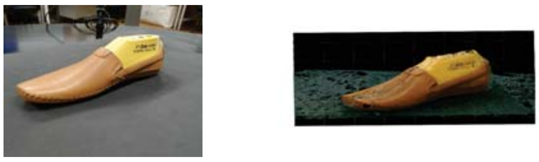
Figure 10: Real shoe on the left and reconstructed shoe on the right: first perspective view.

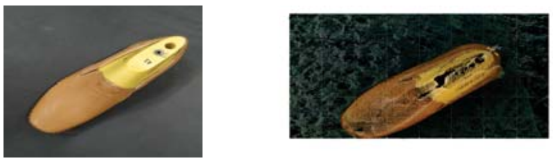
Figure 11: Real shoe on the left and reconstructed shoe on the right: second perspective view.

4.2 Reconstruction of two white polystyrene spheres

The two spheres used for reconstruction are made of white polystyrene and have a very soft surface. The smaller sphere has approximately 3cm of radius, while the bigger has approximately 7.5cm. The two spheres can be seen in Figure 12.

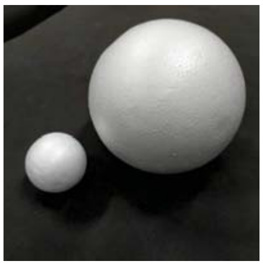
Figure 12: Picture of the two spheres. The smaller sphere (3cm radius) on the left and the bigger (7.5cm radius) on the right side.

As mentioned before, for the reconstruction of the spheres three point clouds from each camera were acquired, with the aim of generating an average point cloud. In Figure 13 the average point clouds for the biggest sphere are presented.

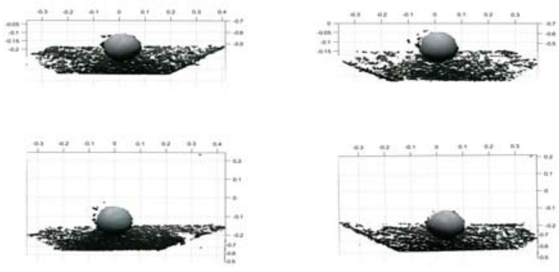
Figure 13: The four different views of the sphere.

Similar to the other reconstructions, these views are then used to build the omnidirectional point cloud. In order to analyze the quality of the reconstructions, for both, the 3cm radius and the 7,5cm radius sphere, the approximated model of the spheres was estimated. The parameters of the models were obtained using a robust estimator, the M-estimator SAmple Consensus (MSAC) algorithm (Torr and Zisserman, 2000). This RANSAC variation is based on the following steps: drawing randomly a minimal sample set; estimating the model and then evaluating the model. This process is repeated until the last iteration that corresponds to the best model.
In Figures 14 and 15, we can view the estimated model of the spheres fitting the point clouds of the real spheres.

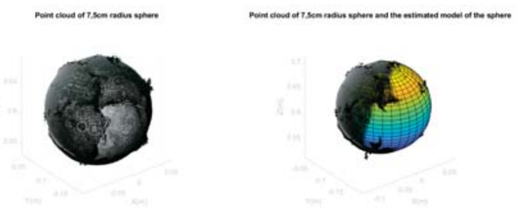
Point cloud of 7,5cm radius sphere
Point cloud of 7,5cm radius sphere and the estimated model of the sphere

Figure 14: On the left side we have the point cloud of the 7,5cm radius sphere and on the right side, the point cloud and the plot of the sphere model.

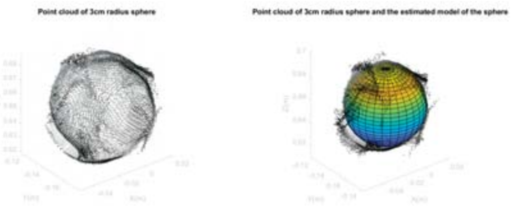
Point cloud of 3cm radius sphere
Point cloud of 3cm radius sphere and the estimated model of the sphere

Figure 15: On the left side we have the point cloud of the 3cm radius sphere and on the right side, the point cloud and the plot of the sphere model.

To analyze the reconstruction of the spheres we used the mean error, that is, the average of the distances from the inliers points (points belonging to the point cloud that were used to estimate the parametric model of the sphere) to the surface of the sphere generated by the parametric model. In table 2, the average errors for the two spheres are presented for four cases namely: the reconstruction made with only one acquisition per camera, 1st acquisition, 2nd acquisition and 3rd acquisition and the reconstruction made with the mean point clouds. The smallest average error is obtained for the reconstruction performed with the mean point clouds.

Table 2: The mean errors obtained for the different estimates of the parametric models.

	Sphere of radius 7.5cm	Sphere of radius 3cm
1st acq.	0.0034m	0.0020m
2nd acq.	0.0029m	0.0020m
3rd acq.	0.0033m	0.0021m
Mean of acq.	0.0021m	0.0017m

4.3 Reconstruction of a hippo balloon deforming

For the reconstruction of the balloon in Figure 16, point clouds were acquired during its emptying.

The balloon was reconstructed in three different stages/time instants, considering that in each stage, the images from each camera are synchronized. The reconstructions are shown in Figure 17, where a plot with the three point clouds together is also presented.

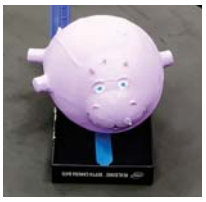

Figure 16: Picture of the hippo-shaped balloon.

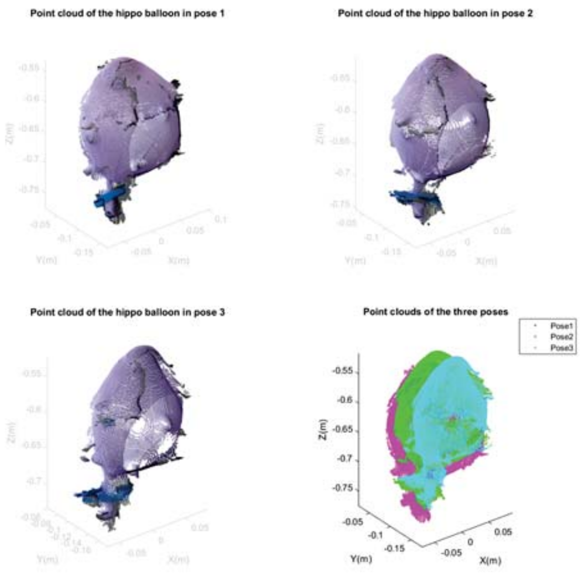
Point cloud of the hippo balloon in pose 1
Point cloud of the hippo balloon in pose 2
Point cloud of the hippo balloon in pose 3
Point clouds of the three poses
Pose1
Pose2
Pose3
Z(m)
Y(m)
X(m)

Figure 17: The first three plots show the hippo balloon's reconstructions in three different sequential phases of the emptying. The fourth image illustrates the three point clouds, being notorious the deformation occurring with the emptying.

5 Conclusion

This paper describes a system designed to acquire synchronized 3D omnidirectional images of objects. That allows for the 3D reconstruction of objects that are articulated or deformable. The experimental results show that specular surfaces as well as sharp corners do not yield good quality reconstructions. Since no controlled illumination is used in the system, we plan to add an illumination system to improve the reconstruction quality. The reconstructions of spheres allow us to conclude that the reconstructions that use the mean of point clouds from each camera seem to have a lower mean error relative to its sphere models, which is an indicator that reconstruction itself is also better. Finally, the balloon's reconstruction shows that this system is suitable for the reconstruction of objects that deform.

ACKNOWLEDGEMENTS

This work was partially supported by Project COMMANDIA SOE2/P1/F0638, from the Interreg Sudoe Programme, European Regional Development Fund (ERDF), and by the Portuguese Government FCT, project no. 006906, reference UID/EEA/00048/2013.

REFERENCES

Arun, K. S., Huang, T. S., and Blostein, S. D. (1987). Least-squares fitting of two 3-d point sets. *IEEE Transactions on pattern analysis and machine intelligence*, (5):698–700.

Curless, B. and Levoy, M. (1996). A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312.

Dai, A., Nießner, M., Zollhöfer, M., Izadi, S., and Theobalt, C. (2017). Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics (ToG)*, 36(4):1.

Grunnet-Jepsen, A., Winer, P., Takagi, A., Sweetser, J., Zhao, K., Khuong, T., Nie, D., and Woodfill, J. (2018). Using the Intel ® RealSense TM Depth cameras D4xx in Multi-Camera Configurations.

Innmann, M., Zollhöfer, M., Nießner, M., Theobalt, C., and Stamminger, M. (2016). Volumedeform: Real-time volumetric non-rigid reconstruction. In *European Conference on Computer Vision*, pages 362–379. Springer.

Kanatani, K.-i. (1994). Analysis of 3-d rotation fitting. *IEEE Transactions on pattern analysis and machine intelligence*, 16(5):543–549.

Matsumoto, J. and Aguilar-Rivera, M. (2018). 3DTracker-FAB documentation.

Newcombe, R. A., Fox, D., and Seitz, S. M. (2015). Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 343–352.

Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohi, P., Shotton, J., Hodges, S., and Fitzgibbon, A. (2011). Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136. IEEE.

Rusinkiewicz, S., Hall-Holt, O., and Levoy, M. (2002). Real-time 3d model acquisition. *ACM Transactions on Graphics (TOG)*, 21(3):438–446.

Slavcheva, M., Baust, M., Cremers, D., and Ilic, S. (2017). Killingfusion: Non-rigid 3d reconstruction without correspondences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1386–1395.

Slavcheva, M., Baust, M., and Ilic, S. (2018). Sobolevfusion: 3d reconstruction of scenes undergoing free non-rigid motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2646–2655.

Svoboda, T., Martinec, D., and Pajdla, T. (2005). A convenient multicamera self-calibration for virtual environments. *Presence Teleoperators Virtual Environ.*, 14(4):407–422.

Torr, P. H. and Zisserman, A. (2000). Mlesac: A new robust estimator with application to estimating image geometry. *Computer vision and image understanding*, 78(1):138–156.

Whelan, T., Salas-Moreno, R. F., Glocker, B., Davison, A. J., and Leutenegger, S. (2016). Elasticfusion: Real-time dense slam and light source estimation. *The International Journal of Robotics Research*, 35(14):1697–1716.

Zollhöfer, M., Nießner, M., Izadi, S., Rehmann, C., Zach, C., Fisher, M., Wu, C., Fitzgibbon, A., Loop, C., Theobalt, C., et al. (2014). Real-time non-rigid reconstruction using an rgb-d camera. *ACM Transactions on Graphics (ToG)*, 33(4):1–12.