Dynamic Occlusion Handling for Real Time Object Perception

Ignacio Cuiral-Zueco and Gonzalo López-Nicolás
Instituto de Investigación en Ingeniería de Aragón
Universidad de Zaragoza, Spain
Email: ignaciocuiral@unizar.es, gonlopez@unizar.es

***Abstract*—An RGB-D based occlusion-handling camera position computation method for proper object perception has been designed and implemented. This proposal is an improved alternative to our previous optimisation-based approach where the contribution is twofold: this new method is geometric-based and it is also able to handle dynamic occlusions. This approach makes extensive use of a ray-projection model where a key aspect is that the solution space is defined within a sphere surface around the object. The method has been designed with a view to robotic applications and therefore provides robust and versatile features. Therefore, it does not require training nor prior knowledge of the scene, making it suitable for diverse applications and scenarios. Satisfactory results have been obtained with real time experiments.**

***Index Terms*—computer vision; deformable object tracking; occlusion handling; best next view**

I. Introduction

Occlusion handling represents a challenge in a variety of problems and applications such as object tracking and optimal object perception. Robot manipulation tasks performed in industrial environments require proper visual perception of target objects. Occasionally, perception is hampered by elements in the environment that lie between the object and the sensor, thus causing occlusions. Identifying occlusions and finding new sensor positions that prevent such occlusion can allow for a more consistent perception.

Multiple object tracking methods tackle occlusion handling. There is a variety of monocular approaches which, for example, adapt to changing visual features and detect occlusions using area and distance heuristics [1], or present a tracking method that makes use of structural information captured in superpixels [2]. Recently, a model-guided method that uses learning-based segmentation and optimisation techniques for pose-estimation in order to define occlusion masks was presented [3]. Another approach tracks the object in a coarse-to-fine manner and formulates the background as a Gaussian model [4]. Unconventional approaches like [5] handle occlusions with the use of correlation filtering and probabilistic finite state machines (FSMs). Not relying on RGB-D sensors, the work in [6] proposes a learning-based method that handles 3D texture-less objects and occlusions. On the other hand, other methods rely on 3D information: Using multi-sensor information, static occlusions that may occur during tracking are predicted and dynamically handled [7]. Alternatively, work in [8] represents targets as 3D point clouds and defines occlusions by solving a partitioning problem.

The papers mentioned above solve the object tracking problem but they define the occlusion handling problem from two different perspectives: (i) some methods, namely [1] [2], focus on 2D tracking in generic images and consider occlusions as elements that may hinder the object tracking process while (ii) other methods like [7] adopt a more active robotics-oriented approach and consider occlusions as elements of the environment that may affect tasks to be performed. The later is the approach adopted in this paper as occlusions will be considered in a camera position computation process, with a view to future applications in the handling of deformable objects by robots. Regarding the camera position optimisation problem, a widespread approach is the Next Best View (NBV) problem (defined in [9]) with problem formulations like the one discussed in [10]. The NBV is usually present in the autonomous generation of complete 3D model of objects [11]. The problem of time-varying scenes requires multi-sensor approaches like [12]. It is also common to find NBV in applications for exploring unknown environments and objects [13]. Solving the NBV problem involves the computation of the camera position for optimal perception, which is related to Visual Servoing (VS) [14]. Visual Servoing aims to control the motion of a robot by, for instance, detecting and following the object of interest. Therefore a target camera position must be generated to allow the tracking of the object at every moment while avoiding occlusions. However, standard VS focuses on rigid objects while VS with non-rigid objects, although it has been addressed in the literature [15], is still an ongoing and researched topic.

This paper is an extension of [16], in which a method for RGB-D deformable object tracking and camera position estimation for optimal object perception is presented. However, in [16], the camera optimisation method does not consider occlusions. Making use of the scene information provided by the deformable object tracking method, the main contribution of this paper is a new occlusion-handling camera position computation method for proper object perception. This method shows a different approach to the one presented in [16], which is energy-function based. A new geometric-based analytical method that can handle both: deformable/dynamic scene elements and moving RGB-D camera, is introduced.

This work was supported by the Spanish Government/European Union through projects PGC2018-098719-B-I00 (MCIU/AEI/FEDER, UE) and COMMANDIA SOE2/P1/F0638 (Interreg Sudoe Programme, ERDF).

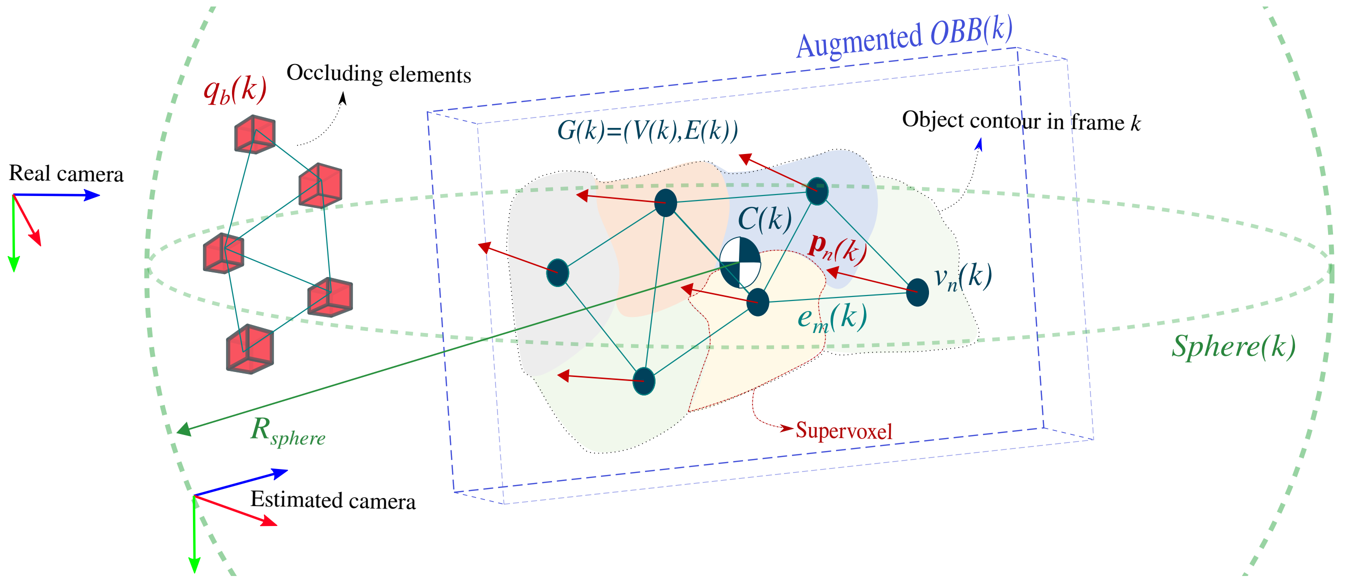


Fig. 1: Elements involved in the problem definition: graph $G(k) = (V(k), E(k))$ with vertices, $V(k)$, defined by the object's $N(k)$ supervoxel centroids and edges, $E(k)$. Associated supervoxel normal vectors $\mathbf{p}_n(k) \in \mathbf{P}(k)$. Occluding elements $q_b(k) \in Q(k)$ that lie within $Sphere(k)$, which is defined by the object centroid $C(k)$ and a radius R_{sphere}. All of the object's supervoxels lie within the augmented $OBB(k)$. The Real camera position and the Estimated camera position are represented with reference frames. The Real camera is obtained from the RGB-D SLAM system while the estimated camera represents the target position from which the occluding elements do not interfere with the object perception.

II. Problem formulation

In this section, elements and assumptions required to define and address the camera location problem are introduced.

A. Target Object.

The system presented in [16] receives RGB and depth-map video frames as input. For each new frame $k \in \mathbb{Z}_{\geq 0}$, the deformable object tracking method provides the set of $N(k)$ Supervoxels [17] that conform the object state in frame k (see Fig. 1). These supervoxels are distributed uniformly in 3D space creating a grid of step-size R_{seed}. They conform a graph $G(k) = (V(k), E(k))$ with vertices, $V(k) = \{v_n(k),\ n = 1, \ldots, N(k)\}$, defined by the object's $N(k)$ supervoxel centroids and edges, $E(k) = \{e_m(k),\ m = 1, \ldots, M(k)\}$, defined by the $M(k)$ connections between neighbouring vertices within an R_{adj}. Radius R_{adj} is closely related to R_{seed}:

$$R_{adj} = \alpha R_{seed}, \tag{1}$$

where α, $1 \leq \alpha \leq 3/2$, is defined so adjacency between supervoxels is ensured after the supervoxel growing process has taken place. Each graph vertex $v_n(k)$ has an associated supervoxel normal $\mathbf{p}_n(k) \in \mathbf{P}(k) = \{\mathbf{p}_n(k),\ n = 1, \ldots, N(k)\}$. The object's centroid $C(k)$ and the voxel cloud associated to the object's supervoxels are used to define an augmented Oriented Bounding Box (OBB) which is obtained through Principal Component Analysis (PCA).

Every frame k, the real camera affine transformation $\mathbf{T}_{real}(k) = [\mathbf{R}_{real} \,|\, \mathbf{t}_{real}]$, where camera orientation $\mathbf{R}_{real}(k) \in \mathbb{R}^{3\times3}$ and position $\mathbf{t}_{real}(k) \in \mathbb{R}^3$ are defined in a fixed global reference frame $\mathbf{W}$, is provided by an RGB-D SLAM system (see [16] for additional details).

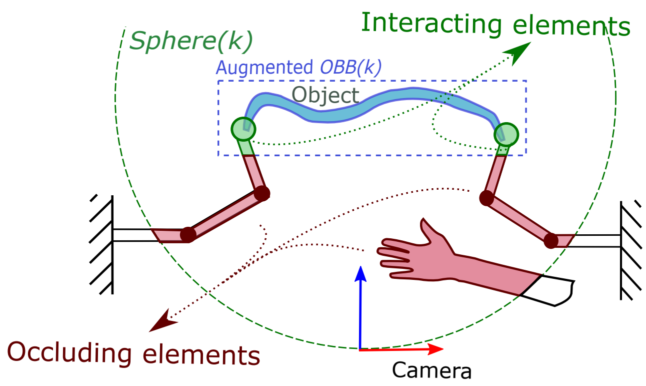


Fig. 2: Scene element classification: Inside the augmented OBB the object and the interacting elements (green). Outside the augmented OBB and inside $Sphere(k)$ the occluding elements (red). On the sphere's surface, the real camera is represented by a reference frame. The aim is to compute a new camera position in order to prevent the occluding elements from interfering with the object.

B. Occluding elements.

In the object tracking process, the scene's point cloud is discretised into supervoxels in every frame. Some of these supervoxels become part of the object, the rest are classified into two different sub-categories:

1) Interacting elements: those supervoxels that lie within the augmented *OBB* but are not part of the object. Usually the object holder or manipulator (Fig. 2).
2) Occluding elements: An $Sphere(k)$ can be defined with centre $C(k)$ and radius:

$$R_{sphere} = d_{pref}, \tag{2}$$

where d_{pref} is the preferred camera-to-object distance (introduced in [16]). $Sphere's$ surface becomes the solution space for the camera position estimation. Supervoxel centroids that lie outside the augmented *OBB* but inside $Sphere(k)$ become occluding elements $Q(k) = \{q_b(k),\ b = 1,\dots,B(k)\}$ (Fig. 1) as they may lie between the object and the desired optimal camera position. The difference between interacting and occluding elements is illustrated and exemplified in Fig.3.

C. Problem definition.

The objective is to determine, in each video frame, a position of the camera on the sphere's surface of radious R_{sphere} (2) seeking perpendicularity to the object's surface. In the case of occlusions interfering with the estimated camera position, positions lying as close as possible to the initial solution that ensure occlusions are avoided will be obtained. There is no prior object knowledge and no assumptions are made about the object characteristics (it may be deformable, textureless, etc.) therefore its initial state should be correctly defined:

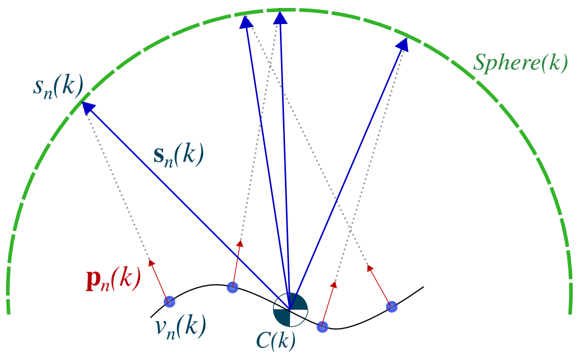
Assumption 1. *The initial position of the sensor (frame $k = 0$) allows complete (not necessarily optimal) observation of the target object's region of interest.*

Although the sensor information is discrete in time, it is assumed that the events in the scene have enough continuity:

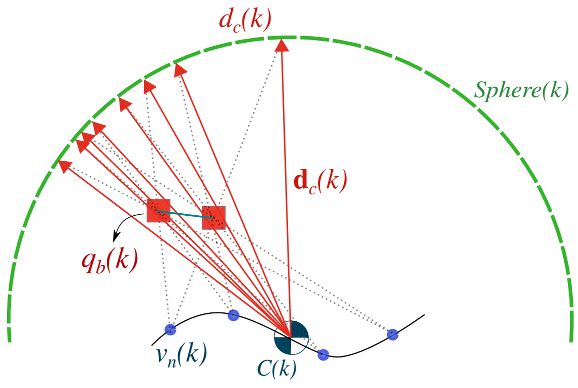
Assumption 2. *The object and the occluding elements may undergo great deformations and movements, but never within time intervals significantly shorter than the sensor's frame rate.*

III. Occlusion handling method

Our occlusion handling method makes use of two sets of sphere-centred rays. These sets are used to compute an initial optimal camera position that, in the case of lying within and occlusion affected area, is iterated until it reaches an occlusion-free position while staying as close as possible from the initial optimal solution. It is worth mentioning in advance that the method explanation elements (camera positions, sphere rays, etc.) are all object-referenced. The object reference is defined by its centroid $C(k)$, and the orientation axes are set, arbitrarily, as the main inertia axes of the objec's voxel cloud.



(a) Normals-to-sphere intersection rays $\mathbf{s}_n(k)$.



(b) Occlusion-to-sphere intersection rays $\mathbf{d}_c(k)$.

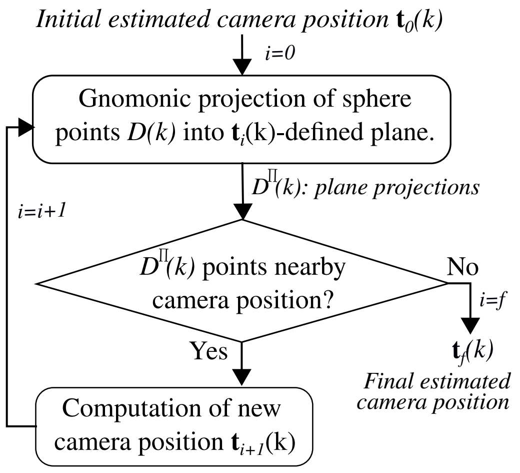
Fig. 3: This figure exemplifies the generation of the two sets of sphere-centred rays. (a) shows how object normals $\mathbf{p}_n(k)$ intersect $Sphere(k)$ in $s_n(k)$ points, which allows defining normals-to-sphere rays $\mathbf{s}_n(k)$. In (b), supervoxel-to-occlusion rays intersect $Sphere(k)$ in $d_c(k)$ points and thus generating occlusion-to-sphere rays $\mathbf{d}_c(k)$. Blue dots denote object's supervoxels and red squares the occluding elements.

A. Generation of sphere-centred sets of rays.

The two sphere-centred set of rays are: (i) the normals-to-sphere rays and the (ii) occlusion-to-sphere rays (Fig. 3). They enclose, respectively, information on the object's surface configuration and the relative position of occlusions with respect to the object. The normals-to-sphere set is defined by sphere surface points $s_n(k)$, $S(k) = \{s_n(k),\ n = 1,\dots,N(k)\}$, obtained from the intersection of the object supervoxel normals $\mathbf{p}_n(k)$ with $Sphere(k)$. These intersection points $s_n(k)$ and the sphere centre $C(k)$ define the set of rays $\mathbf{S}(k) = \{\mathbf{s}_n(k),\ n = 1,\dots,N(k)\}$. These rays are computed as:

$$\mathbf{s}_n(k) = s_n(k) - C(k). \tag{3}$$

On the other hand, occlusion-to-sphere rays are defined by $d_c(k)$ sphere surface points. These points are obtained with the rays defined by supervoxel centroids $v_n(k)$ and the occluding elements positions $q_b(k)$ intersecting $Sphere(k)$ of radius R_{sphere} (2). Intersection points $d_c(k)$, $D(k) =$



Initial estimated camera position $\mathbf{t}_0(k)$
i=0
Gnomonic projection of sphere points D(k) into $\mathbf{t}_i$(k)-defined plane.
$D^{\Pi}(k)$: plane projections
$D^{\Pi}(k)$ points nearby camera position?
No
i=f
$\mathbf{t}_f(k)$
Final estimated camera position
Yes
Computation of new camera position $\mathbf{t}_{i+1}$(k)
i=i+1

Fig. 4: Flowchart representing the iterative process for the final camera position $\mathbf{t}_f(k)$ computation. Using the initial camera position $\mathbf{t}_0(k)$ as input, the iterative process performs a gnomonic projection of the occlusion-to-sphere points $D(k)$ into a plane tangent to $Sphere(k)$ in point $\mathbf{t}_i(k)$. If no projected points $D^{\Pi}(k)$ lie nearby $\mathbf{t}_i(k)$, $\mathbf{t}_{i=f}(k)$ becomes the final camera position.

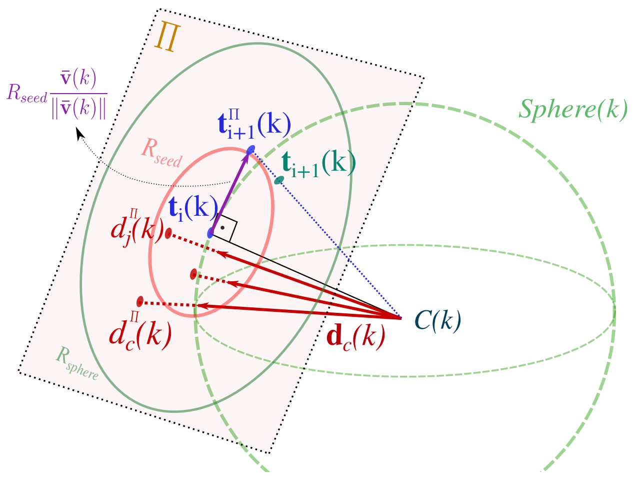


Fig. 5: Illustration of the $\mathbf{t}_{i+1}(k)$ camera position computation process. In each iteration, the method makes use of the gnomonic projection of occlusion-to-sphere intersection rays $\mathbf{d}_c(k) \in \mathbf{D}(k)$. The projection points, $D^{\Pi}(k)$, are then used to obtain vector $\bar{\mathbf{v}}(k)$. Adding scaled $\bar{\mathbf{v}}(k)$ to $\mathbf{t}_i(k)$ allows obtaining $\mathbf{t}^{\Pi}_{i+1}(k)$, which is then mapped back to the sphere surface as $\mathbf{t}_{i+1}(k)$.

$\{d_c(k),\ c = 1,\dots,N_C(k)\}$, and the sphere centre $C(k)$ define the set of occlusion-to-sphere rays $\mathbf{D}(k) = \{\mathbf{d}_c(k),\ c = 1,\dots,N_C(k)\}$, being $N_C(k) = N(k)B(k)$, and $\mathbf{d}_c(k) = d_c(k) - C(k)$.

B. Obtention of initial camera estimation.

Normals-to-sphere rays $\mathbf{s}_n(k)$ serve the purpose of defining the camera position and optical axis orientation in such a way that perpendicularity is seeked for each supervoxel $v_n(k)$ and the object centroid $C(k)$ is pointed at. The initial camera pose is the position and orientation the camera would need to adopt if no occlusions existed. It is obtained by computing the angular average of normals-to-sphere rays $\bar{\mathbf{s}}(k)$. This average is computed in spherical coordinates $(R_{sphere}, \bar{\theta}, \bar{\varphi})$ as follows:

$$\bar{\theta} = \operatorname{atan2}\left(\frac{1}{N}\sum_{n=1}^{N}\cos(\theta_n), \frac{1}{N}\sum_{n=1}^{N}\sin(\theta_n)\right), \tag{4}$$

$$\bar{\varphi} = \operatorname{atan2}\left(\frac{1}{N}\sum_{n=1}^{N}\cos(\varphi_n), \frac{1}{N}\sum_{n=1}^{N}\sin(\varphi_n)\right). \tag{5}$$

The intersection of $\bar{\mathbf{s}}(k)$ with $Sphere(k)$ generates point $\bar{s}(k)$, which is the initial camera position $\mathbf{t}_0(k) = \bar{s}(k)$. The camera's $\mathbf{Z}_0(k)$ axis orientation is $\mathbf{Z}_0(k) = -\bar{\mathbf{s}}(k)/\|\bar{\mathbf{s}}(k)\|$, as this ensures the camera is pointing towards the sphere centre $C(k)$.

C. Iterative occlusion-free camera pose estimation.

Occlusion-to-sphere rays encapsulate the information of the positions $d_c(k) \in D(k)$ contained in $Sphere(k)$ from which a supervoxel $v_n(k)$ of the object is occluded by element $q_b(k)$. In order to avoid occlusions and provide alternative camera positions to the initial one, an iterative process has been designed and implemented. The iterative process for the final camera position $\mathbf{t}_f(k)$ computation uses the initial optimal camera position $\mathbf{t}_0(k) = \bar{s}(k)$ as input (Fig. 4). Beginning with $\mathbf{t}_0(k)$ and for the following camera positions $\mathbf{t}_i(k)$, iteration $i = 1,...,f$ performs a gnomonic projection of the occlusion-to-sphere points $D(k)$, which are projected into a plane $\Pi_i(k)$ tangent to $Sphere(k)$ in point $\mathbf{t}_i(k)$ (Fig. 5). The normal vector of the plane is the camera's $\mathbf{Z}_i(k)$ axis, which is computed as $\mathbf{Z}_i(k) = -\mathbf{t}_i(k)/\|\mathbf{t}_i(k)\|$. Projected plane points $d^{\Pi}_c(k) \in D^{\Pi}(k)$ for iteration i are:

$$d^{\Pi}_c(k) = C(k) + \mathbf{d}_c(k)\,\frac{(\mathbf{t}_i(k) - C(k))^{\intercal}(-\mathbf{Z}_i(k))}{\mathbf{d}_c(k)^{\intercal}(-\mathbf{Z}_i(k))}. \tag{6}$$

If no projected points $D^{\Pi}(k)$ lie within a plane-contained R_{seed} radius circumference centred in $\mathbf{t}_i(k)$, $\mathbf{t}_{i=f}(k)$ becomes the final camera position, otherwise a new camera position $\mathbf{t}_{i+1}(k)$ is computed and the iterative process continues. The computation of the new camera position is performed by making use of the projected occlusion-to-sphere points $d^{\Pi}_c \in D^{\Pi}(k)$. Projected points that lie within a plane-contained R_{sphere} radius circumference centred in $\mathbf{t}_i(k)$ are denoted as $d^{\Pi}_j(k) \in D^{\Pi}(k)$ (Fig. 5). A set of vectors $\mathbf{V}(k) = \{\mathbf{v}_j(k),\ j = 1,\dots,J(k) \ni \mathbf{v}_j(k) = \overrightarrow{d^{\Pi}_j(k)\,\mathbf{t}_i(k)}\}$ is defined with a common ending in $\mathbf{t}_i(k)$ and origins in each $d^{\Pi}_j(k) \in D^{\Pi}(k)$. Vector $\bar{\mathbf{v}}(k)$ is the vector sum of all $\mathbf{v}_j(k)$ re-sized and added to $\mathbf{t}_i(k)$:

$$\mathbf{t}^{\Pi}_{i+1}(k) = \mathbf{t}_i(k) + R_{seed}\frac{\bar{\mathbf{v}}(k)}{\|\bar{\mathbf{v}}(k)\|}. \tag{7}$$

Once $\mathbf{t}^{\Pi}_{i+1}(k)$ is obtained, it is mapped back to the sphere surface using the same gnomonic projection model and thus obtaining $\mathbf{t}_{i+1}(k)$:

$$\mathbf{t}_{i+1}(k) = R_{sphere}\frac{\mathbf{t}^{\Pi}_{i+1}(k) - C(k)}{\|\mathbf{t}^{\Pi}_{i+1}(k) - C(k)\|}. \tag{8}$$

Once the iteration process comes to an end, $\mathbf{t}_{i=f}(k)$ becomes the occlusion-free camera position of frame k. The camera's $\mathbf{Z}_f(k)$ axis orientation is $\mathbf{Z}_f(k) = -\mathbf{t}_f(k)/\|\mathbf{t}_f(k)\|$ and thus ensuring the camera is pointing towards the sphere centre $C(k)$.

IV. Experimental results

The system's program is run and fed with the information obtained from an RGB-D camera in real time. Although the method has been designed to handle dynamic objects and cameras, in the experiment we focus on the occlusion avoidance feature and, for the sake of simplicity, both the camera and the target object are static. Numeric values of relevant parameters used in the experiment are $R_{seed} = 0.02\,[m]$ and $R_{sphere} = 0.5\,[m]$. Experiments have been conducted in a workspace of approximately 1 cubic meter. Sequences were recorded in real time with the Realsense D435 RGB-D camera. The programming language used is C++ and all measurements have been recorded on an Intel Core i7 1.8 GHz processor.

The target object, a DVD case, is a flat plastic object that allows to intuitively figure out what position of the camera might provide a solution. In Fig. 6 three frames of the processed sequence are shown. The real camera position $\mathbf{t}_{real}$, the initial camera estimation $\mathbf{t}_0$ and the final camera estimation $\mathbf{t}_f$ are displayed along with the object's voxel cloud, the occluding elements and other elements of the method. The first frame is the base case scenario where no occluding elements interfere and thus $\mathbf{t}_0 = \mathbf{t}_f$. In the second displayed frame, an occluding element (hand) shifts the initial camera position estimation to the left while in the third one two occluding elements (two hands) force the estimated camera to move upwards in order to have a proper object perception. Normals-to-sphere intersection rays $\mathbf{S_n}$ are represented as blue lines. It is rather noticeable how most of them are grouped in front of the DVD case while some others seem to disperse up and left. This second group corresponds to the normal vectors of the supervoxels in the DVD case edges. A comparison of the angular error $\varepsilon_{n,i}$ between the initial and the final estimated positions has been carried out. $\varepsilon_{n,i}$ represents how far the angular alignment between camera position $\mathbf{t}_i$ and ray $\mathbf{s}_n$ is. Ideally, angle between these vectors should be $0\,[^{\circ}]$ as to perceive supervoxel n more perpendicularly while pointing at the object's centroid $C(k)$. The angular error between an object normal $\mathbf{p}_n$ and a camera estimated position $\mathbf{t}_i$ is computed as:

$$\varepsilon_{n,i} = \operatorname{atan2}\left((\mathbf{s}_n \times \mathbf{t}_i)^{\intercal}\frac{\mathbf{s}_n \times \mathbf{t}_i}{\|\mathbf{s}_n \times \mathbf{t}_i\|}, -\mathbf{t}_i^{\intercal}\mathbf{s}_n\right). \tag{9}$$

TABLE I: Mean $\bar{\varepsilon}$ and standard deviations σ of the angular errors in the displayed frames for initial $\mathbf{t}_0$ and final $\mathbf{t}_f$.

	Camera	$\bar{\varepsilon}$ [°]	σ [°]
Frame $a)$	$\mathbf{t}_0$	11.12	7.93
	$\mathbf{t}_f$	11.12	7.93
Frame $b)$	$\mathbf{t}_0$	12.86	10.04
	$\mathbf{t}_f$	20.38	8.33
Frame $c)$	$\mathbf{t}_0$	14.15	9.03
	$\mathbf{t}_f$	33.03	8.17

Table I shows the mean $\bar{\varepsilon}$ and standard deviations σ of the angular errors in the displayed frames for both estimated cameras: initial ($\mathbf{t}_0$) and final ($\mathbf{t}_f$). Comparing both cameras in the second and the third frames, angular error means are approximately doubled thus highlighting the trade-off between adequate perception and occlusion avoidance. Number of object supervoxels in each frame are $N = 25$, $N = 27$ and $N = 17$ (a, b and c respectively). Note how, in the experiments, the real camera is not moved to the estimated camera position: Robot-based camera positioning (Visual Servoing) is left for upcoming research as this paper focuses on the camera position estimation problem.

V. Discussion and conclusions

A method for calculating a camera position that allows perceiving a target object while avoiding occlusions has been designed and implemented. The system uses RGB and depth information and can be applied to deformable objects and dynamic occlusions. The solution space has been defined within a sphere surface around the object which allows to choose the desired perception distance, something useful in possible robotic applications where a tools require a constant working distance. Previous methods [1] [2] [8] have a more visual-based approach for object tracking. Approaches in [6] or [7] take a perspective more suitable for physical applications while others like [18] tackle the drone physical occlusion avoidance problem directly. The value of our proposal relies on its practical approach for application in object manipulation scenarios. Similarly to [18], this method has been designed with a view to robotic applications where interaction with the environment is expected. Amongst its strengths, computation times of the method are within the order of milliseconds. Other advantages are: the method flexibility and adaptability. It can be adjusted to handle many applications and a variety of scenarios as it does not require prior information or training. Relying on a geometric-based design, this approach provides an analytical scene evaluation thus making it robust and resilient as can be seen in the experiment (Fig. 6, c), where a double occlusion is handled. Making extensive use of RGB-D information and not only focusing on colour or texture allows the method to overcome texture-less elements (occluders and objects).

Future applications of this occlusion handling method include camera servoing for deformable object state feedback in shape manipulation control: Even in controlled environments, the manipulator robots can create occlusions.

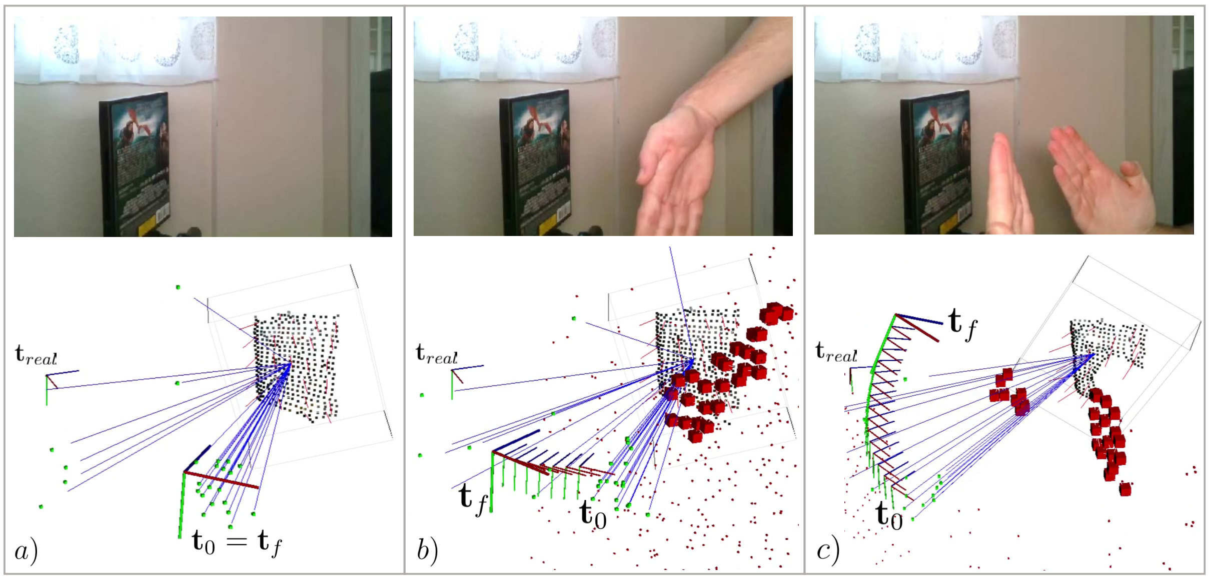


Fig. 6: Three frames of the processed sequence are shown (top) along with several scene elements. The real camera position $\mathbf{t}_{real}$, the initial camera estimation $\mathbf{t}_0$ and the final camera estimation $\mathbf{t}_f$ are displayed along with the object's voxel cloud, the normals-to-sphere intersection rays $\mathbf{S_n}$ (blue lines), the occluding elements Q (red squares) and the gnomonic projection of D points: D^{Π} (small red dots). The object's augmented *OBB* and supervoxel normals (in red) are also shown. In a) the base case scenario is displayed and, since there are no occluding elements interfering, $\mathbf{t}_0 = \mathbf{t}_f$. b) an occluding element (hand) shifts the initial camera position estimation to the left while in c) two occluding elements force it to move upwards.

A proper perception system capable of handling such occlusions, like the one presented in this paper, could be of use in scenarios such as those presented in [19].

REFERENCES

[1] A. Yilmaz, X. Li, and M. Shah. Contour-based object tracking with occlusion handling in video acquired using mobile cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1531–1536, 2004.

[2] F. Yang, H. Lu, and M.H. Yang. Robust superpixel tracking. *Transactions on Image Processing*, 23(4):1639–1651, 2014.

[3] L. Zhong, Y. Zhang, H. Zhao, A. Chang, W. Xiang, S. Zhang, and L. Zhang. Seeing through the occluders: Robust monocular 6-dof object pose tracking via model-guided video object segmentation. *IEEE Robotics and Automation Letters*, 5(4):5159–5166, 2020.

[4] S. Zhao, S. Zhang, and L. Zhang. Towards occlusion handling: object tracking with background estimation. *IEEE Transactions on Cybernetics*, 48(7):2086–2100, 2017.

[5] C. Liu, D. Q. Huynh, and M. Reynolds. Toward occlusion handling in visual tracking via probabilistic finite state machines. *IEEE Transactions on Cybernetics*, 50(4):1726–1738, 2018.

[6] A. Crivellaro, M. Rad, Y. Verdie, K. M. Yi, P. Fua, and V. Lepetit. Robust 3D object tracking from monocular images using stable parts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(6):1465–1479, 2017.

[7] P. Wunsch and G. Hirzinger. Real-time visual tracking of 3D objects with dynamic handling of occlusion. In *IEEE International Conference on Robotics and Automation*, volume 4, pages 2868–2873, 1997.

[8] A. Cavagna, S. Melillo, L. Parisi, and F. Ricci-Tersenghi. SpaRTA Tracking across occlusions via partitioning of 3D clouds of points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[9] R. Pito. A solution to the next best view problem for automated surface acquisition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(10):1016–1030, 1999.

[10] J. Delmerico, S. Isler, R. Sabzevari, and D. Scaramuzza. A comparison of volumetric information gain metrics for active 3D object reconstruction. *Autonomous Robots*, 42(2):197–208, 2018.

[11] M. Krainin, B. Curless, and D. Fox. Autonomous generation of complete 3D object models using next best view manipulation planning. In *IEEE International Conference on Robotics and Automation*, pages 5031–5037, 2011.

[12] R. Herguedas, G. López-Nicolás, and C. Sagüés. Multi-camera coverage of deformable contour shapes. In *IEEE 15th International Conference on Automation Science and Engineering (CASE)*, pages 1597–1602, 2019.

[13] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart. Receding horizon "next-best-view" planner for 3D exploration. In *IEEE International Conference on Robotics and Automation*, pages 1462–1468, 2016.

[14] F. Chaumette and S. Hutchinson. Visual servo control. I. basic approaches. *IEEE Robotics & Automation Magazine*, 13(4):82–90, 2006.

[15] D. Santosh and C.V. Jawahar. Visual servoing in non-rigid environments: A space-time approach. In *IEEE International Conference on Robotics and Automation*, pages 2452–2457, 2007.

[16] I. Cuiral-Zueco and G. López-Nicolás. RGB-D tracking and optimal perception of deformable objects. *IEEE Access*, 8:136884–136897, 2020.

[17] J. Papon, A. Abramov, M. Schoeler, and F. Worgotter. Voxel cloud connectivity segmentation-supervoxels for point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2027–2034, 2013.

[18] B. Penin, P. R. Giordano, and F. Chaumette. Vision-based reactive planning for aggressive target tracking while avoiding collisions and occlusions. *IEEE Robotics and Automation Letters*, 3(4):3725–3732, 2018.

[19] D. Navarro-Alarcon, H.M. Yip, Zerui Wang, Y. Liu, F. Zhong, T. Zhang, and P. Li. Automatic 3D manipulation of soft objects by robotic arms with an adaptive deformation model. *IEEE Transactions on Robotics*, 32(2):429–441, 2016.