# A modular framework for model-based visual tracking using edge, texture and depth features

Souriya Trinh, Fabien Spindler, Eric Marchand and François Chaumette

***Abstract*— We present in this paper a modular real-time model-based visual tracker. It is able to fuse different types of measurement, that is, edge points, textured points, and depth map, provided by one or multiple vision sensors. A confidence index is also proposed for determining if the outputs of the tracker are reliable or not. As expected, experimental results show that the more various measurements are combined, the more accurate and robust is the tracker. The corresponding C++ source code is available for the community in the ViSP library.**

## I. Introduction

The ability to accurately localize a camera with respect to an object of interest is a crucial step toward bringing dynamic manipulation in robotic vision. With a real-time process, complex tasks such as object grasping or robot positioning can then be performed in closed-loop by visual servoing to take into account perturbations and dynamic environment [1]. Augmented reality applications [2] and indoor navigation of mobile robots [3] can also be considered. Main challenges of object tracking concern the ability to not only accurately track the object but also to be able to bring reliability and robustness.

We describe in this paper the main building blocks of the general model-based tracking framework implemented in the latest version of ViSP open-source C++ library[1]. ViSP [4] is a modular cross-platform library that allows prototyping and developing applications related to visual tracking and visual servoing (VS). Although ViSP has been initially developed mainly for visual servoing purposes with basic tracking capabilities [4], such capabilities have been extended over the time. Along with direct image registration or template-based trackers [5], [6], a first model-based tracker was introduced in the library using edges only [7].

Model-based tracking [8], [9], [7], [10] aims at computing the object pose with respect to the camera from the knowledge of the geometric object model. The basic principle consists in estimating the pose by minimizing the norm of the reprojection error using iterative non-linear minimization techniques, such as a Gauss-Newton or Levenberg-Marquardt. Minimizing this reprojection error provides the Maximum Likelihood estimate when a Gaussian noise is assumed on the measurements.

The earliest methods considered models composed of object edges and distance to contour as basic measurement. Nevertheless, these methods may fail when the object or background is textured, and the robustness deteriorates when ambiguities between some edges occur. One way to address this issue is to fuse the information coming from edges with information given by keypoints [10], [11], [12]. Another solution, considered in this paper, is to exploit RGB-D images that provide depth along with photometric data [13].

In this paper, we emphasize on the genericity of the model-based tracking method that allows combining easily different types of visual features (object contours, textured points of interest, depth features) observed from one or several vision sensors (e.g. RGB-D cameras). In particular, we present how depth information can be integrated for improving tracking robustness and accuracy. A confidence index is also proposed for determining if the outputs of the tracker are reliable or not. Numerous experimental results are finally provided in Section V.

## II. Related Work

As stated in the previous section, the new model-based tracker described in this paper and implemented into the latest available version of ViSP has a direct filiation with [7]. Using Moving-Edges (ME) algorithm [14] allows tracking efficiently the object contours and provides real-time performance. Integration of robust M-estimators in the estimation process allows handling partial occlusions and outliers. A closely related work [15] is based on a 1D-search along the object contours and a Lie algebra formulation to update the estimated pose. An extension to multiple cameras was proposed in [16]. Our model-based tracker has then been extended in [11] to include texture information for improving tracking robustness. A similar approach using a GPU to handle object visibility is described in [17] with a complete framework to initialize the tracker. In [10] a particle filter on $SE(3)$ is used to maintain multiple hypotheses for the estimated pose, which allows avoiding the estimation to converge to local minimum. In [18], the GPU capabilities are exploited to render the object and a particle filter is also used to update the pose. [19] focuses on taking into account region appearance to deal with highly cluttered backgrounds.

With the availability of low-cost RGB-D sensors, depth map is a valuable visual cue for improving the tracking accuracy and robustness [13]. Using a Gaussian filter approach, [20] considers only the depth map for estimating the pose that best registers the observed and rendered depth maps. Within a particle filter framework, [21] uses RGB-D data and a proposal distribution to improve the energy-based observation model and reduce the number of particles. A

This work was supported by the H2020 COMANOID EU project.

The authors are with Inria, Univ Rennes, CNRS, IRISA, France first.last-name@inria.fr.

[1] https://visp.inria.fr

probabilistic model is used in [22] to track multiple similar objects with RGB-D images and histograms as appearance models for the registration. Another region-based approach is proposed in [23] with temporally consistent, local color histograms for real-time pose detection and tracking of rigid objects. Multiple cameras are used in [24] to increase the tracking accuracy. Different visual cues such as depth, normals or object appearances are fused with joint state of the robot to track multiple objects in presence of occlusions. The idea of fusing multiple visual cues is also considered in [12] with edge, point, and color features [25] for robust tracking in presence of motion blur and challenging illumination conditions. GPU can be used to render directly the object silhouette for edge tracking and allows avoiding the need to pre-process the object CAD model [12], [10], [26].

## III. Model-based tracking

### A. Problem formulation

Let us denote $\mathcal{F}_c$ the camera frame and ${}^c\mathbf{T}_o$ the transformation that defines the pose of the object frame $\mathcal{F}_o$ with respect to $\mathcal{F}_c$. ${}^c\mathbf{T}_o$, is a homogeneous matrix defined such that

$$ {}^c\mathbf{T}_o = \begin{pmatrix} {}^c\mathbf{R}_o & {}^c\mathbf{t}_o \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix} \tag{1} $$

where ${}^c\mathbf{R}_o$ and ${}^c\mathbf{t}_o$ are the rotation matrix and the translation vector defining the change of frames.

Pose estimation by model-based tracking is an optimization problem that can be solved using an iterative non-linear minimization technique, such as Gauss-Newton or Levenberg-Marquardt. Denoting $\mathbf{q} \in SE(3)$ a minimal representation of ${}^c\mathbf{T}_o$ ($\mathbf{q} = ({}^c\mathbf{t}_o, \theta\mathbf{u})$ where $\theta$ and $\mathbf{u}$ are the angle and the axis of the rotation ${}^c\mathbf{R}_o$), the objective consists in estimating the six independent parameters involved in $\mathbf{q}$ by registering the geometric object model $\mathbf{M}$ expressed in the sensor space with respect to visual features $\mathbf{x}^*$ extracted from the sensor. The problem can be formulated as

$$ \hat{\mathbf{q}} = \underset{\mathbf{q}}{\operatorname{argmin}} \sum_i \varepsilon_i^2 \left(\mathbf{x}_i^*, \mathbf{x}_i(\mathbf{M}_i, \mathbf{q}, \xi)\right) \tag{2} $$

where $\mathbf{M}_i$ are parts of the model and $\mathbf{x}_i(\mathbf{M}_i, \mathbf{q}, \xi)$ expresses the relation between the model and the image space for pose $\mathbf{q}$ and calibration parameters $\xi$. $\varepsilon(\mathbf{x}, \mathbf{x}')$ is the registration residual, that is typically the Euclidean distance between two visual features $\mathbf{x}$ and $\mathbf{x}'$ that have to be registered. If the set of visual features are well chosen, there exists only one single pose $\hat{\mathbf{q}}$ such that $\varepsilon_i^2 = 0, \forall i$.

We will see in Section III-C that $\mathbf{M}_i$, $\mathbf{x}_i$, $\mathbf{x}_i^*$, and thus the residual $\varepsilon_i$ can take many forms (distance to contour, distance between points, distance to 3D plane, etc.).

### B. Robust minimization

As explained before, the optimization consists in minimizing the norm of the error vector $\mathbf{e}$ obtained by stacking the set of residuals $\varepsilon_i$: $\mathbf{e} = (\varepsilon_1, \cdots, \varepsilon_n)$. The time derivative of this vector can be expressed as

$$ \dot{\mathbf{e}} = \frac{\partial \mathbf{e}}{\partial \mathbf{q}} \dot{\mathbf{q}} = \mathbf{L}_\mathbf{e}\mathbf{v} \tag{3} $$

where $\mathbf{v} \in se(3)$ is the velocity screw and $\mathbf{L}_\mathbf{e}$ is the interaction matrix related to $\mathbf{x}$ [1].

To achieve a decoupled exponential decrease of the residuals, that is, $\dot{\mathbf{e}} = -\lambda\mathbf{e}$ where $\lambda$ is a positive scalar, one minimization step would be given by $\mathbf{v} = -\lambda {\mathbf{L}_\mathbf{e}}^+ \mathbf{e}$. However, to deal with outliers, a weight $w_i$ is associated to each residual $\varepsilon_i$ for representing the confidence in the corresponding visual feature. This leads to introduce a diagonal matrix $\mathbf{W} = diag\left(w_1, \cdots, w_n\right)$ from which we obtain [7]

$$ \mathbf{v} = -\lambda \left(\mathbf{W}\mathbf{L}_\mathbf{e}\right)^+ \mathbf{W}\, \mathbf{e}. \tag{4} $$

Note that $\mathbf{v}$ can be seen as the velocity of a virtual camera (from which $\mathbf{x}(\mathbf{M}, \mathbf{q}, \xi)$ is computed) at each iteration of the optimization process. Then, the camera pose is updated with

$$ {}^{c_{(k+1)}}\mathbf{T}_o = \Delta\mathbf{T}^{-1}\ {}^{c_k}\mathbf{T}_o \tag{5} $$

where $k$ denotes the iteration number of the minimization process. The exponential map allows transforming the velocity $\mathbf{v}$ to the incremental pose $\Delta\mathbf{T} = {}^{c_k}\mathbf{T}_{c_{(k+1)}} = \exp\left(\mathbf{v}\right)$.

Each element of the diagonal weighting matrix $\mathbf{W}$ represents the confidence of the corresponding visual feature

$$ w_i = \frac{\psi\left(\delta_i/\sigma\right)}{\delta_i/\sigma} \tag{6} $$

where $\psi\left(u\right)$ is the influence function and $\delta_i$ is the normalized residual given by $\delta_i = \varepsilon_i - Med\left(\mathbf{e}\right)$ where $Med\left(\mathbf{e}\right)$ is the median of $\mathbf{e}$.

Many influence functions exist in the literature. Tukey has been chosen as it rejects completely the detected outliers by assigning them a zero weight. This way, a detected outlier will not influence the tracking. Details about the Tukey influence function are given in [27]. It is given by:

$$ \psi\left(u\right) = \begin{cases} u\left(C^2 - u^2\right)^2, & \text{if } |u| \leq C \\ 0, & \text{else} \end{cases} \tag{7} $$

where $C = 4.6851$ represents the proportionality factor for Tukey's function for 95 percent efficiency in the case of a Gaussian noise. It assumes a variance of 1 for the residuals distribution. So, an online scale estimation of the standard deviation $\sigma$ is performed using the median absolute deviation of the residuals [7].

### C. Visual features

As stated now, different types of visual features can be supported. In this paper, we consider the distance between a contour point in the image and the projected object contour (*edge-based features*), the distance between two keypoints in two successive images (*keypoint-based feature*), and finally the distance between a 3D point measured by a RGB-D camera and a 3D plane (*depth-based feature*).

*a) Edge-based feature:* Edge-based tracking relies on the ME algorithm [14] that tracks contour points along the normal of the projected object contours. We consider here as residual $\varepsilon_i = d_\perp\left(\mathbf{p}_i, \mathbf{l}_i\left(\mathbf{q}\right)\right)$ that corresponds to the distance in the image between a ME $\mathbf{p}$ and the projected straight line $\mathbf{l}\left(\mathbf{q}\right)$ (see Figure 1) represented in polar coordinates with

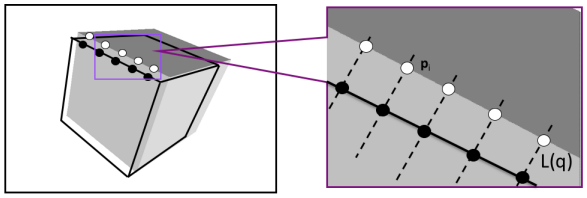


Fig. 1. Edge-based feature overview

$x\cos\theta + y\sin\theta = \rho, \forall (x,y) \in \mathbf{l}(\mathbf{q})$. For each ME, the residual is then defined by

$$\varepsilon_{d_\perp} = \rho - (x\cos\theta + y\sin\theta) \tag{8}$$

where $(x, y)$ are the ME coordinates. The corresponding interaction matrix is given in [7]

$$\mathbf{L}_{d_\perp} = \begin{bmatrix} \lambda_{d_\perp}\cos\theta \\ \lambda_{d_\perp}\sin\theta \\ -\lambda_{d_\perp}\rho \\ \left(1+\rho^2\right)\sin\theta - \alpha\rho\cos\theta \\ -\left(1+\rho^2\right)\cos\theta - \alpha\rho\sin\theta \\ -\alpha \end{bmatrix}^\top \tag{9}$$

where $\lambda_{d_\perp} = \lambda_\rho + \alpha\lambda_\theta$,

$$\begin{aligned} \lambda_\rho &= \left(A\rho\cos\theta + B\rho\sin\theta + C\right)/D, \\ \lambda_\theta &= \left(A\sin\theta - B\cos\theta\right)/D, \\ \alpha &= x\sin\theta - y\cos\theta \end{aligned}$$

and $(A, B, C, D)$ defines the 3D plane coordinates (expressed in the camera frame) to which the line belongs. These 3D parameters are obtained from $\mathbf{q}$ and the object model.

Any polyhedral object can be considered, as well as cylinders. 3D circles can also be considered. In this later case, a point-to-ellipse distance is used (see [7] for more details).

*b) Keypoint-based feature:* When dealing with keypoint-based features, we consider the relation between point coordinates in two successive images. Such keypoints can be detected thanks to the Harris detector and then tracked using the classical KLT algorithm [5] (alternatively any keypoint detection and matching process could be considered). Let us denote $\bar{\mathbf{p}} = (x, y, 1)$ the homogeneous coordinates of a keypoint in the first image where it has been detected (from a camera located in ${}^{c(0)}\mathbf{T}_o$) and $\bar{\mathbf{p}}^* = (x^*, y^*, 1)$ its coordinates in the current image. Let us also denote ${}^c\mathbf{T}_{c(0)}$ the pose between these two views. Since the keypoints belong to a plane (they are detected on the planar faces of the object), we have $\bar{\mathbf{p}}^* = {}^c\mathbf{H}_{c(0)}\ \bar{\mathbf{p}}$ where ${}^c\mathbf{H}_{c(0)}$ is a homography that depends of ${}^c\mathbf{T}_{c(0)}$. Assuming that ${}^{c(0)}\mathbf{T}_o$ is known from the pose estimation process at view $c(0)$, ${}^c\mathbf{H}_{c(0)}$ can of course be expressed as a function of the pose ${}^c\mathbf{T}_o$ to be estimated. More precisely, for each keypoint, we consider the point-to-point distance in the image between the point $\bar{\mathbf{p}}$ transfered in the current image (according to the estimated pose $\mathbf{q}$) and it correspondence $\bar{\mathbf{p}}^*$

$$\begin{pmatrix} \varepsilon_x \\ \varepsilon_y \end{pmatrix} = \begin{pmatrix} x(\mathbf{q}) - x^* \\ y(\mathbf{q}) - y^* \end{pmatrix} \tag{10}$$

where $(x(\mathbf{q}), y(\mathbf{q}), 1) = {}^c\mathbf{H}_{c(0)}\bar{\mathbf{p}}$ with

$${}^c\mathbf{H}_{c(0)} = {}^c\mathbf{R}_{c(0)} + \frac{{}^c\mathbf{t}_{c(0)}}{d}\mathbf{n}^\top \tag{11}$$

where $\mathbf{n}$ and $d$ are the normal and the distance to the camera center of the planar face expressed in the camera frame $c(0)$. The corresponding interaction matrix is given by, [1], [2]

$$\mathbf{L}_\mathbf{p} = \begin{bmatrix} -\frac{1}{Z} & 0 & \frac{x}{Z} & xy & -\left(1+x^2\right) & y \\ 0 & -\frac{1}{Z} & \frac{y}{Z} & 1+y^2 & -xy & -x \end{bmatrix} \tag{12}$$

The actual $1/Z$ value appearing in (12) is computed by

$$\frac{1}{Z} = \frac{\left({}^c\mathbf{R}_{c(0)}\ \mathbf{n}\right)^\top \bar{\mathbf{p}}^*}{{}^c\mathbf{t}_{c(0)}^\top\ {}^c\mathbf{R}_{c(0)}\ \mathbf{n} - d} \tag{13}$$

In the current implementation, keypoints can be considered for any planar faces of the object, such as circles for instance, and also on the surface of cylinders.

*c) Depth-based feature:* The introduction of the Kinect and derived RGB-D sensors allows getting easily the depth of a point in the image. To exploit these data, the residual $\varepsilon_Z$ is defined as the point-to-plane distance, in 3D, such that

$$\varepsilon_Z = Z(n_x x + n_y y + n_z) + D \tag{14}$$

where $Z$ is the depth of the point with image coordinates $(x, y)$ measured in the depth map and $n_x X + n_y Y + n_z Z + D = 0$ is the planar face equation expressed in the camera frame. The corresponding interaction matrix is given by

$$\mathbf{L}_\mathbf{z} = \begin{bmatrix} n_x & n_y & n_z & (n_z yZ - n_y Z) \\ & & & (n_x Z - n_z xZ) \quad (n_y xZ - n_x yZ) \end{bmatrix} \tag{15}$$

All terms involved in (14) and (15) are obtained either directly as measurements $(x, y, Z)$, or from the planar face of the object and the current camera pose $(n_x, n_y, n_z)$.

### D. Generic model-based tracking

Model-based tracking using a single type of visual features has already been proposed in the literature. So does the hybrid tracker combining edges and keypoints in [12] although with a different formulation. We propose to go further by proposing a general model-based tracker to let the user combine any types of visual features and multiple cameras. A straightforward example is to consider RGB-D sensor where edges and keypoints can be exploited from the color camera and depths from the depth sensor. Multiple (stereo) cameras are used in [16]. In such cases, it is necessary to know the homogeneous transformation between the different sensor frames which can be known through an a priori extrinsic calibration process.

Concretely, features combination consists in stacking the residuals in a vector set $\mathbf{e}$ and similarly for the corresponding interaction matrix. The iterative minimization process then regulates to zero the different feature errors. To deal with visual features extracted from different cameras, a reference sensor frame must be considered. The time variation of the visual features $\mathbf{e}_2$ observed in camera frame $\mathcal{F}_2$ expressed in camera frame $\mathcal{F}_1$ is then

$$\dot{\mathbf{e}}_2 = \mathbf{L}_{\mathbf{e}_2}\,{}^2\mathbf{V}_1\,{}^1\mathbf{v} \tag{16}$$

with ${}^2\mathbf{V}_1$ the velocity twist matrix

$$ {}^2\mathbf{V}_1 = \begin{bmatrix} {}^2\mathbf{R}_1 & \left[{}^2\mathbf{t}_1\right]_\times {}^2\mathbf{R}_1 \\ \mathbf{0}_{3\times 3} & {}^2\mathbf{R}_1 \end{bmatrix} \tag{17} $$

which allows computing a velocity ${}^1\mathbf{v}$ expressed in frame $\mathcal{F}_1$ in frame $\mathcal{F}_2$. The velocity to be applied in the complete minimization step is given by [16]

$$ {}^1\mathbf{v} = -\lambda \begin{pmatrix} \mathbf{L}_{\mathbf{e}_1} \\ \mathbf{L}_{\mathbf{e}_2} {}^2\mathbf{V}_1 \\ \mathbf{L}_{\mathbf{e}_3} {}^3\mathbf{V}_1 \\ \vdots \end{pmatrix}^+ \begin{pmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \\ \mathbf{e}_3 \\ \vdots \end{pmatrix} \tag{18} $$

Stacking visual features of different natures and from different viewpoints allows improving the tracking accuracy and robustness by exploiting the redundancy of the available measurements. Robust weights are computed for each type of visual features to maintain the coherence of the feature errors. An advantage of this approach is that it is not required to weight the visual features influence with respect to each other.

### E. Covariance matrix

It is possible to estimate the covariance matrix of the estimated pose. Considering the weighting matrix $\mathbf{W}$, it is given by

$$ \Gamma\left(\mathbf{q}\right) = \sigma^2 \left( {\mathbf{J}_\mathbf{e}}^\top \, \mathbf{W}^2 \, \mathbf{J}_\mathbf{e} \right)^{-1} \tag{19} $$

where $\mathbf{J}_\mathbf{e} = \mathbf{L}_\mathbf{e} \mathbf{L}_\mathbf{q}^{-1}$ is defined from $\dot{\mathbf{e}} = \mathbf{J}_\mathbf{e} \, \dot{\mathbf{q}} = \mathbf{L}_\mathbf{e} \mathbf{L}_\mathbf{q}^{-1} \dot{\mathbf{q}}$ (with $\mathbf{L}_\mathbf{q}$ given in [1]) and where

$$ \sigma^2 = \frac{1}{n} \left(\mathbf{e}_f - \mathbf{L}_\mathbf{e} \mathbf{v}_f\right)^\top \mathbf{W}^2 \left(\mathbf{e}_f - \mathbf{L}_\mathbf{e} \mathbf{v}_f\right) \tag{20} $$

where $\mathbf{e}_f$ is the final value of the residual vector after convergence of the estimation process, $\mathbf{v}_f$ the velocity computed at the last iteration, and $n$ the number of features.

## IV. Implementation details

Along with the overview of the general tracker presented in the previous section, practical aspects have to be considered to provide an efficient tracker[2].

### A. Object modeling

From any model of the object built in a CAD software, information about the object geometry (set of lines, circles, cylinders and planar faces) are easily extracted to be used by the model-based tracker.

### B. Visibility handling

Different strategies are implemented to handle object parts visibility. The former is based on the angle between a ray from the optical axis and the face normal. This allows very fast processing but is not enough for objects of complex shapes. In this case, a scan-line rendering algorithm is used and allows per-pixel visibility instead of per-primitive only, at the expense of higher computation time. Clipping is also used to restrain the object model to the camera view frustum.

[2] More information in "Markerless generic model-based tracking" tutorial: `http://visp-doc.inria.fr/doxygen/visp-daily/tutorial-tracking-mb-generic.html`.

### C. Tracking initialization

Automatic initialization[3] of the tracker is provided in ViSP by matching keypoints detected in the very first image and those extracted from training images, with an approach similar to [17]. The correspondences between points in the image and points in the training images (whose 3D coordinates are known) allows estimating the pose using a Perspective-n-Points (PnP) algorithm. RANSAC is considered for rejecting outliers coming from spurious data. This approach is valid mainly for textured objects, and more elaborate techniques exist to handle more complex cases, such as using RGB-D data [28], [29] or with deep-learning-based approaches [30].

### D. Confidence index

The covariance matrix of the estimated pose given in Section III-E is not reliable to detect drift in the tracking (since drifts generally correspond to the convergence of the estimation in a local minimum, where the residuals are small). That is why a confidence index has been implemented to detect when the tracking starts to drift. This is achieved by computing the mean angular error between the normals to the object contours model and the observed gradient orientations in the image. Measure points are sampled at regular locations from the projected model (see Fig. 1) and a derivative kernel is used to compute the gradient orientation. The average angular error (between 0 and 90°) remains low when the estimated pose is correct. When the tracking drifts, the model starts to be projected onto the background, and the confidence index returns a high value. Thresholding the confidence index allows detecting drift in the tracking. Coupled with the initialization process using the pose estimation method described above, this scheme allows automatic reinitialization of the tracking after a drift.

## V. Experimental results

We report in this section the tracking results on three image sequences and compare different combination of visual features. Two sensors are used. The first one is an Intel® RealSense™ SR300 sensor designed for close range (0.2 m to 1.5 m) applications. The second one is an ASUS Xtion PRO LIVE RGB-D sensor for a depth range starting around 0.8 m to approximatively 3.5 m. For each method, the tracking is stopped when the confidence index is higher than 20°. In a real application, the tracker would be reinitialized after a detected tracking drift according to IV-C.

### A. Performance measures

Computation time depends on the object complexity and the number of extracted visual features. Table I summarizes the mean tracking time for edge, keypoint, and edge + keypoint + depth trackers on a regular processor (Intel® Core™ i7-4600U) for the three image sequences analyzed in the next section. Edge and keypoint trackers run at video rate. Depth-based tracker involves much more features, which impacts directly the computation time. The depth map is

[3] `http://visp-doc.inria.fr/doxygen/visp-daily/tutorial-detection-object.html`

subsampled by a factor 4 to enable real-time tracking (more than 10 Hz). Due to the complex structure of the castle, scan-line rendering used to handle visibility tests for the castle sequence increases the computation time.

| | | Edge | Keypoint | Edge+ keypoint+depth |
|---|---|---|---|---|
| Castle | Mean time | 36 (ms) | 8 (ms) | 87 (ms) |
| | Mean nb feat. | 747 | 488 | 5052 |
| Printer | Mean time | 5 (ms) | 5 (ms) | 20 (ms) |
| | Mean nb feat. | 369 | 319 | 5970 |
| Breaker | Mean time | 7 (ms) | 5 (ms) | 23 (ms) |
| | Mean nb feat. | 459 | 110 | 6518 |

TABLE I

MEAN COMPUTATION TIME AND AVERAGE NUMBER OF VISUAL FEATURES. SCAN-LINE VISIBILITY IS USED FOR THE CASTLE SEQUENCE.

### B. *Model-based tracking of a miniature castle*

This experiment involves the tracking of a miniature castle using the SR300 sensor. The camera is static and the object is moved by hand. 688 images are acquired at 30 Hz.

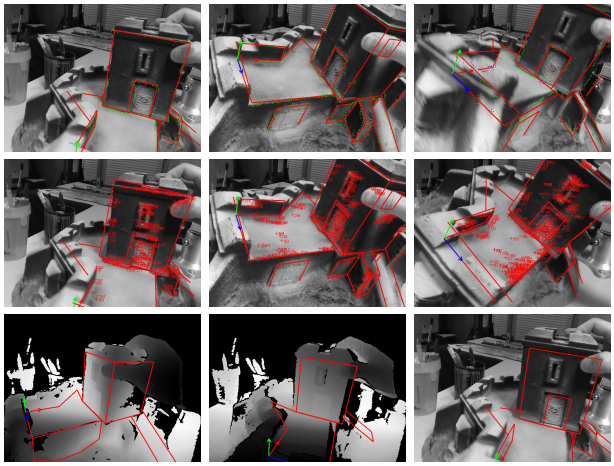

Fig. 2. Top row: edge tracker at frames 87, 172, 265. Middle row: keypoint tracker at frames 87, 172, 243. Bottom row: depth tracker at frames 50, 87 and corresponding color image at frame 87.

Fig. 2 shows the tracking results using a single type of visual features. The edge-based tracker is able to track on 265 images before starting to drift when too many ME are badly tracked on the left fortification to be handled by the M-estimators. Keypoint-based tracker drifts similarly and is stopped at frame 243 thanks to the confidence index. Depth-based tracker alone stops at frame 87. Indeed, when the front elements are not visible, this configuration does not allow observing the 6 parameters of the pose (interaction matrix is rank deficient). The tracker starts sliding when the miniature castle is moved perpendicularly to the hidden tower face.

Different visual features combinations (edge + keypoint, edge + depth, edge + keypoint + depth) for tracking are displayed on Fig. 3. Tracking with edge and keypoint features stops at frame 266. Indeed, the model is badly registered and the drift is correctly detected. Adding depth feature to

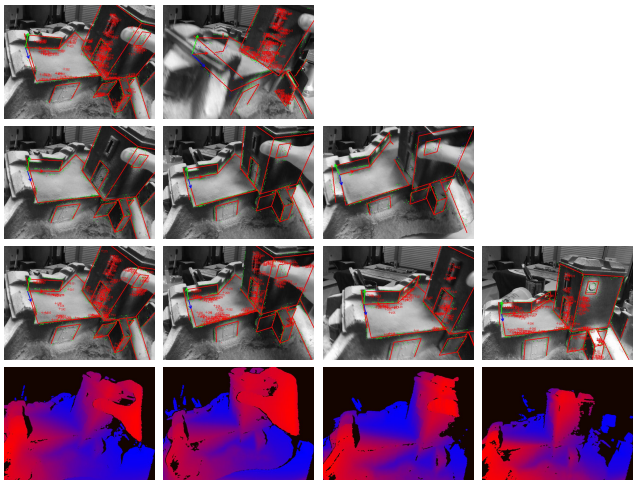

Fig. 3. Top row: edge + keypoint tracker at frames 172, 266. Second row: edge + depth tracker at frames 172, 344, 447. Third row: tracking using all features at frames 172, 344, 516, 688. Bottom row: visualization of the depth map at frames 172, 344, 516, 688.

edges allows tracking until frame 447. But then the tracking drifts when the ME are badly tracked. Indeed, depth features cannot prevent the drift since the left tower face is not visible. Only the full combination of features is able to track correctly the whole sequence. We can also note that robust M-estimators allows tracking the castle correctly even with large occlusions made by the hand.

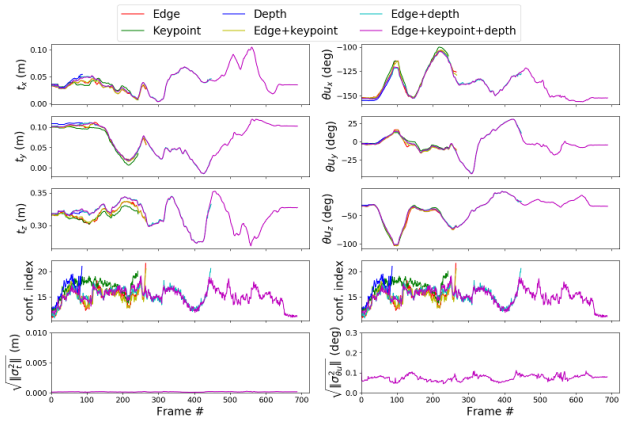


Fig. 4. Comparison of the pose estimated by the six methods. Confidence index is displayed on 4th row, left figure and replicated on the right figure. Square root of the covariance norm is displayed on the last row, in translation (left) and in rotation (right).

In Fig. 4, the estimated pose (translation and $\theta\mathbf{u}$ vectors) is plotted for the different trackers. A divergence in the estimated pose compared to the real one is quickly followed by an increase of the confidence index. This allows to correctly detect the observed tracking drifts. Last row in Fig. 4 shows the covariance in translation and in rotation. As stated in Section IV-D, covariance matrix of the estimated pose cannot be used to reliably detect a tracking drift. Nevertheless, order of magnitude of the covariance values can give an estimate of the accuracy of the pose estimation. Thus, using an identical scale for each covariance plot allows

comparing the accuracy of the pose estimation between each experiment. With edge, keypoint and depth features, precision of the pose estimation in translation is around the millimeter, and precision is around one-tenth degree in rotation. These values are coherent with the tracking results and the expected accuracy of the tracker at this distance.

### C. Model-based tracking of a printer

In this experiment, the tracking of a printer is done using the ASUS Xtion sensor. The sequence is composed of 1408 images acquired at 30 Hz. Results are presented in Fig. 5

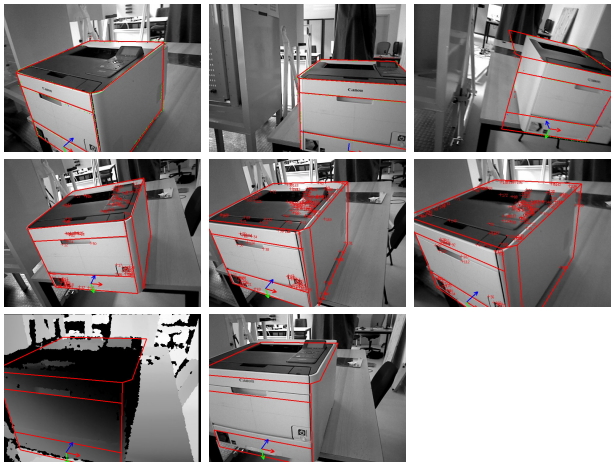

Fig. 5. Top row: edge tracker at frames 282, 564, 684. Middle row: keypoint tracker at frames 150, 200, 228. Bottom row: depth tracker at frame 1 and corresponding color image at frame 1.

Edge-based tracker correctly tracks the printer until around frame 610 when it starts to slightly drift. The greyish poorly contrasted background causes the ME to be falsely tracked and the estimated pose is thus not well registered with the object. The drift is detected at frame 684 when sufficient portion of the model is projected onto the background. Detection of the keypoint-tracker drift occurs at frame 228. The drift can be explained by the approximation made to model the top printer part by a planar face. Moreover, the printer is mostly textureless, which degrades the quality of this tracker. Once the model is badly registered, depth and keypoint features are badly computed and the tracking drifts until it triggers our drift detector. For the last case, the quality of the depth map is too poor to allow an accurate tracking using solely depth features. Fig. 5 shows the model projected into the color frame using the pose estimated with depth. Since the projected lines do not match with the printer contour, a drift is immediately detected.

Combining keypoints and edges allows avoiding the drift observed with the edge-tracker. This is not the case when combining edges and depths. Indeed, the depth map visible in Fig. 6 is not fully reliable and does not provide enough information to correct the wrong registration of the model. On the other hand, combining all the visual feature types allows successful tracking during the whole sequence.

From Fig. 7, the instants when the different trackers start to drift are visible on the estimated pose curves. As they

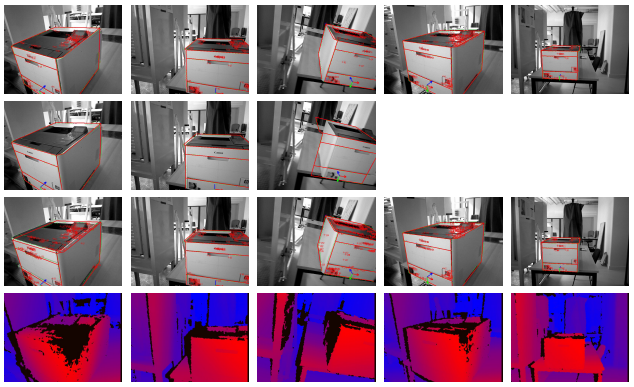

Fig. 6. First and third rows: edge + keypoint and edge + keypoint + depth tracker at frames 282, 564, 682, 1128, 1408. Second row: edge + depth tracker at frames 282, 564, 682. Bottom row: visualization of the depth map at same frames than first and third rows.

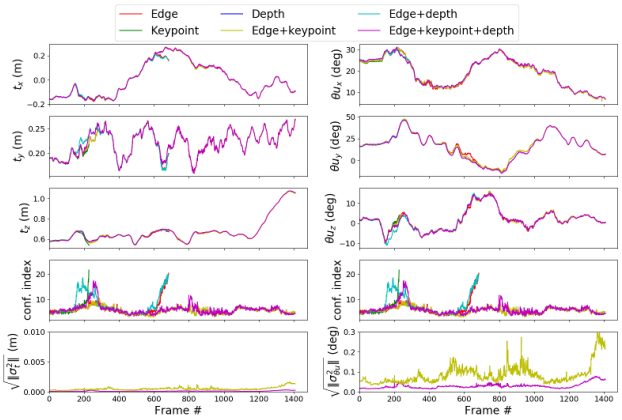


Fig. 7. Comparison of the pose estimated by the six methods. Confidence index is displayed on 4th row, left figure and replicated on the right figure. Square root of the covariance norm is displayed on the last row, in translation (left) and in rotation (right).

diverge, the confidence index rapidly increases until reaching the alert threshold. Covariance measures are displayed for both edge + keypoint and edge + keypoint + depth tracking, since they are successful to track the printer for the whole sequence. Good precision of the pose estimation is also obtained on this sequence. Adding depth features allows increasing the tracking accuracy, which is confirmed from the covariance curve. Augmentation of the covariance at the end of the sequence corresponds to the movement when the camera moves away from the printer. This experiment illustrates the complementarity between classical cameras and depth sensors. Indeed, using depth only can be a limiting factor and the possibility to combine visual features of different nature is clearly an advantage of our tracker.

### D. Model-based tracking of a circuit breaker

This last experiment involves the tracking of a circuit breaker using the ASUS Xtion sensor. The sequence is composed of 1362 images acquired at 30 Hz by moving manually the sensor.

Using a single type of visual features is not enough to correctly track the circuit breaker, as shown on Fig. 8. For

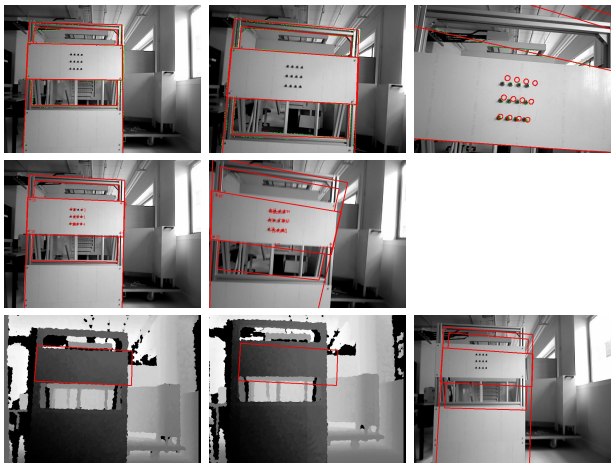

Fig. 8. Top row: edge tracker at frames 272, 544, 788. Middle row: keypoint tracker at frames 272, 495. Bottom row: depth tracker at frames 50 and 173 and corresponding color image at frame 173.

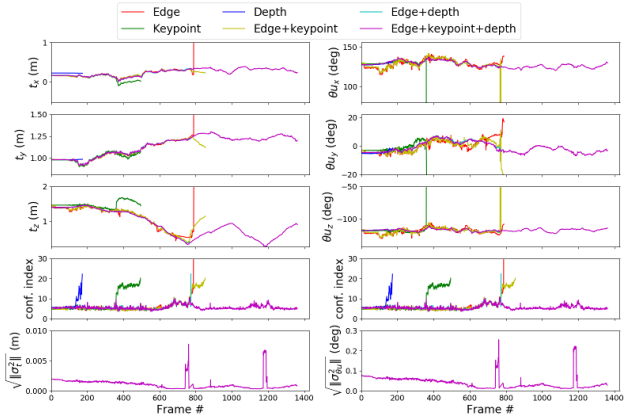


Fig. 10. Comparison of the pose estimated by the six methods. Confidence index is displayed on 4th row, left figure and replicated on the right figure. Square root of the covariance norm is displayed on the last row, in translation (left) and in rotation (right).

the edge tracker, the drift arises when the camera comes really close to the object and no more vertical anchors are visible. When the camera goes back, this movement cannot be accurately estimated and ME are then tracked on elements in the background. Keypoint tracker is not able to properly track since the object is mainly textureless. Finally, the depth tracker is not efficient. Indeed, it is not possible to correctly estimate the 6 parameters of the pose with only a planar face.

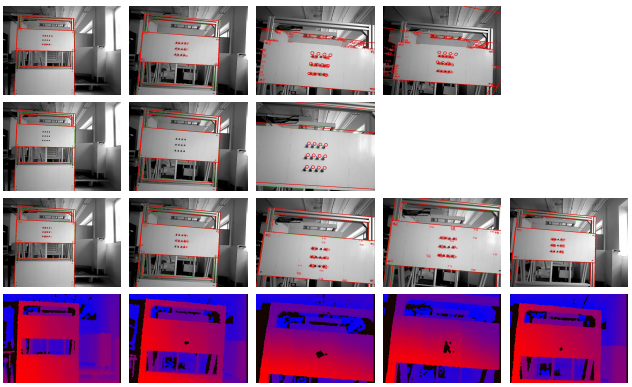

Fig. 9. Top row: edge + keypoint tracker at frames 272, 544, 816, 853. Second row: edge + depth tracker at frames 272, 544, 774. Third row: edge + keypoint + depth tracker at frames 272, 544, 816, 1088, 1362. Bottom row: visualization of the depth map at the same frames.

Combining edges and keypoints does not prevent the previously mentioned tracking drift. But thanks to the confidence index, it is correctly detected. As expected, combining edges and depths does not allow avoiding the issue. Only the full features combination allows successful tracking during the whole sequence. Indeed, while depth is useless to estimate the motions parallel to the panel, it does help to estimate correctly the distance to the panel and thus allows avoiding the tracking drift observed with edge and edge + keypoint features. From Fig. 10, tracking failures can be clearly seen on the estimated camera poses. Sudden increases in confidence index are directly linked to tracking drifts, while the confidence index always remains at a low value for the successful tracker. As the camera comes closer to the circuit breaker, the covariance values decrease. Indeed, accuracy in the pose estimation is lower when we are far from the object. Sudden peaks in the covariance curve correspond to the instants when the depth map cannot be exploited since the sensor is too close to the object. This is also noticeable by comparing the peaks with the $t_z$ translation plot.

In Fig. 11, the camera trajectories are displayed for edge and keypoint and edge, keypoint and depth trackers. Adding depth features allows having much more smoother trajectory, especially when the sensor is far from the object.

### *E. Availability*

The complete model-based tracker presented in this paper is fully available in the ViSP library from `https://visp.inria.fr` under the GPL licence.

## VI. Conclusions

We presented a general model-based tracker able to combine different types of visual features (edge, keypoint and depth) for improving the tracking robustness and accuracy. Implemented into the ViSP open-source library, the user can easily choose which features to use, depending on the nature of the object to be tracked. In particular, a novel depth feature has been integrated into the tracking framework. Thanks to the versatility of the tracking method, visual features extracted from different viewpoints can be fused as soon as the change of frames between each sensor is known from an a priori calibration step. Experiments have shown that adding depth information to the classical edge or keypoint features improve the tracking robustness. Indeed, the dense nature of the depth map allows having more reliable measurements. Moreover, the model is registered in 3D (point-to-plane distance) rather than in the 2D image space. Even with objects whose full pose is not observable with depth features only, combining depth with edges and/or points of interest reduces tracking jitters.

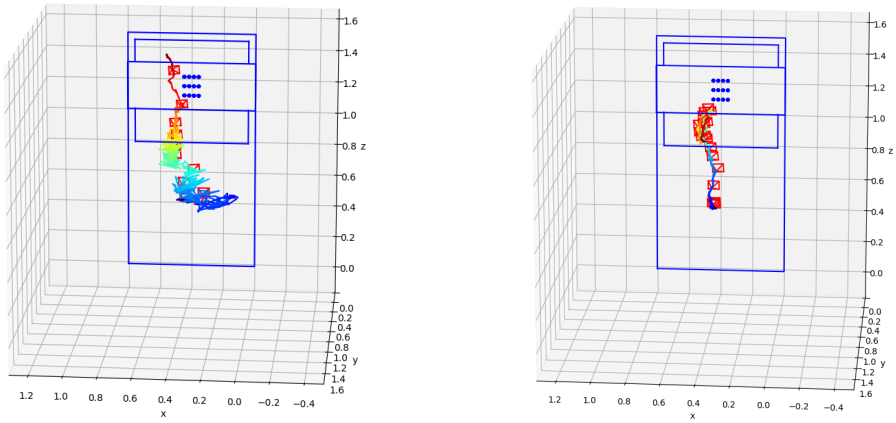

Fig. 11. Left: camera trajectory estimated with edge + keypoint tracking. Right: camera trajectory estimated with edge + keypoint + depth tracking.

## REFERENCES


[1] F. Chaumette and S. Hutchinson, "Visual servo control, Part I: Basic approaches," *IEEE Robotics and Automation Magazine*, vol. 13, no. 4, pp. 82–90, 2006.

[2] E. Marchand, H. Uchiyama, and F. Spindler, "Pose estimation for augmented reality: a hands-on survey," *IEEE Trans. on Visualization and Computer Graphics*, vol. 22, no. 12, pp. 2633–2651, December 2016.

[3] A. Paolillo, A. Faragasso, G. Oriolo, and M. Vendittelli, "Vision-based maze navigation for humanoid robots," *Autonomous Robots*, vol. 41, no. 2, pp. 293–309, 2017.

[4] E. Marchand, F. Spindler, and F. Chaumette, "ViSP for visual servoing: a generic software platform with a wide class of robot control skills," *IEEE Robotics and Automation Magazine*, vol. 12, no. 4, pp. 40–52, 2005, special issue on Software Packages for Vision-Based Control of Motion, P. Oh, D. Burschka (Eds.).

[5] S. Baker and I. Matthews, "Lucas-kanade 20 years on: A unifying framework," *Int. Journal of Computer Vision*, vol. 56, no. 3, pp. 221–255, 2004.

[6] S. Benhimane and E. Malis, "Real-time image-based tracking of planes using efficient second-order minimization," in *IEEE/RSJ Int. Conf. on Intelligent Robots Systems*, Sendai, Japan, Oct. 2004, pp. 943–948.

[7] A. Comport, E. Marchand, M. Pressigout, and F. Chaumette, "Real-time markerless tracking for augmented reality: the virtual visual servoing framework," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 4, pp. 615–628, 2006.

[8] D. Lowe, "Fitting parameterized three-dimensional models to images," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 13, no. 5, pp. 441–450, May 1991.

[9] T. Drummond and R. Cipolla, "Real-time visual tracking of complex structures," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 932–946, July 2002.

[10] C. Choi and H. Christensen, "Robust 3d visual tracking using particle filtering on the special euclidean group: A combined approach of keypoint and edge features," *Int. Journal of Robotics Research*, vol. 31, no. 4, pp. 498–519, Apr. 2012.

[11] M. Pressigout and E. Marchand, "Real-time hybrid tracking using edge and texture information," *International Journal of Robotics Research*, vol. 26, no. 7, pp. 689–713, 2007.

[12] A. Petit, E. Marchand, and K. Kanani, "Combining complementary edge, point and color cues in model-based tracking for highly dynamic scenes," in *IEEE Int. Conf. on Robotics and Automation, ICRA'14*, Hong-Kong, Hong Kong SAR China, June 2014.

[13] F. Ireta and A. I. Comport, "Point-to-hyperplane RGB-D Pose Estimation: Fusing Photometric and Geometric Measurements," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2016)*, Daejeon, South Korea, Oct. 2016.

[14] P. Bouthemy, "A maximum likelihood framework for determining moving edges," *IEEE Trans. on Pattern Analysis and Machine intelligence*, vol. 11, no. 5, pp. 499–511, May 1989.

[15] T. Drummond and R. Cipolla, "Real-time visual tracking of complex structures," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 932–946, July 2002.

[16] F. Dionnet and E. Marchand, "Stereo tracking and servoing for space applications," *Advanced Robotics*, vol. 23, no. 5, pp. 579–599, 2009.

[17] C. Choi and H. I. Christensen, "Real-time 3d model-based tracking using edge and keypoint features for robotic manipulation," in *2010 IEEE International Conference on Robotics and Automation*, May 2010, pp. 4048–4055.

[18] P. Azad, D. Münch, T. Asfour, and R. Dillmann, "6-dof model-based tracking of arbitrarily shaped 3d objects," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 5204–5209.

[19] B.-K. Seo, H. Park, J.-I. Park, S. Hinterstoisser, and S. Ilic, "Optimal local searching for fast and robust textureless 3d object tracking in highly cluttered backgrounds," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 1, pp. 99–110, Jan. 2014.

[20] J. Issac, M. Wüthrich, C. G. Cifuentes, J. Bohg, S. Trimpe, and S. Schaal, "Depth-based object tracking using a robust gaussian filter," *CoRR*, vol. abs/1602.06157, 2016.

[21] A. Krull, F. Michel, E. Brachmann, S. Gumhold, S. Ihrke, and C. Rother, "6-dof model based tracking via object coordinate regression," in *ACCV*, 2014.

[22] C. Y. Ren, V. Prisacariu, O. Kaehler, I. Reid, and D. Murray, "3d tracking of multiple objects with identical appearance using rgb-d input," in *2014 2nd International Conference on 3D Vision*, vol. 1, Dec 2014, pp. 47–54.

[23] H. Tjaden, U. Schwanecke, and E. Schmer, "Real-time monocular pose estimation of 3d objects using temporally consistent local color histograms," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017, pp. 124–132.

[24] K. Pauwels and D. Kragic, "Simtrack: A simulation-based framework for scalable real-time object pose detection and tracking," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2015, pp. 1300–1307.

[25] G. Panin, E. Roth, and A. Knoll, "Robust contour-based object tracking integrating color and edge likelihoods," in *Proc. of the Vision, Modeling, and Visualization Conference 2008, VMV 2008*, Konstanz, Germany, Oct. 2008, pp. 227–234.

[26] H. Wuest and D. Stricker, "Tracking of industrial objects by using cad models," *Journal of Virtual Reality and Broadcasting*, vol. 4, no. 1, Apr. 2007.

[27] P.-J. Huber, *Robust Statistics*. Wiler, New York, 1981.

[28] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother, "Learning 6d object pose estimation using 3d object coordinates," in *European conference on computer vision*. Springer, 2014, pp. 536–551.

[29] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab, "Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes," in *Proceedings of the 11th Asian Conference on Computer Vision - Volume Part I*, ser. ACCV'12. Berlin, Heidelberg: Springer-Verlag, 2013, pp. 548–562.

[30] B. Tekin, S. N. Sinha, and P. Fua, "Real-time seamless single shot 6d object pose prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 292–301.