Detection and Fine 3D Pose Estimation of Texture-less Objects in RGB-D Images

Tomáš Hodaň[♯], **Xenophon Zabulis**[♭], **Manolis Lourakis**[♭], **Štěpán Obdržálek**[♯], **Jiří Matas**[♯]
[♯]*Center for Machine Perception, Czech Technical University in Prague, Czech Republic*
hodantom|xobdrzal|matas@cmp.felk.cvut.cz
[♭]*Institute of Computer Science, Foundation for Research and Technology - Hellas, Heraklion, Greece*
zabulis|lourakis@ics.forth.gr

***Abstract*— Despite their ubiquitous presence, texture-less objects present significant challenges to contemporary visual object detection and localization algorithms. This paper proposes a practical method for the detection and accurate 3D localization of multiple texture-less and rigid objects depicted in RGB-D images. The detection procedure adopts the sliding window paradigm, with an efficient cascade-style evaluation of each window location. A simple pre-filtering is performed first, rapidly rejecting most locations. For each remaining location, a set of candidate templates (*i.e.* trained object views) is identified with a voting procedure based on hashing, which makes the method's computational complexity largely unaffected by the total number of known objects. The candidate templates are then verified by matching feature points in different modalities. Finally, the approximate object pose associated with each detected template is used as a starting point for a stochastic optimization procedure that estimates accurate 3D pose. Experimental evaluation shows that the proposed method yields a recognition rate comparable to the state of the art, while its complexity is sub-linear in the number of templates.**

I. Introduction

Texture-less, smooth and uniformly colored objects occur frequently in robotic applications that range from personal robotics to intelligent manipulation and assembly. Common to such applications is the requirement of identifying and accurately localizing known objects so that they can be acted upon by a robot end effector. Fig. 1 depicts an example of a robotic assembly scenario involving several texture-less objects. An arm with a gripper is assigned the task of picking up electrical fuses, at arbitrary locations in its workspace, and inserting them into the sockets of corresponding fuse boxes.

The method detailed in this paper aims at the reliable simultaneous detection of multiple texture-less objects with low false detection rate, real time performance, and sub-centimeter accuracy in object localization. The input to the method consists of RGB-D images provided by a consumer-grade depth sensor such as Kinect. Such sensors provide aligned color and depth images that concurrently capture both the appearance and geometry of a scene.

This work was supported in part by the EC FP7 programme under grant no. 270138 DARWIN, by CTU student grant SGS15/155/OHK3/2T/13, and by the Technology Agency of the Czech Republic research program TE01020415 (V3C – Visual Computing Competence Center) TE01020415.

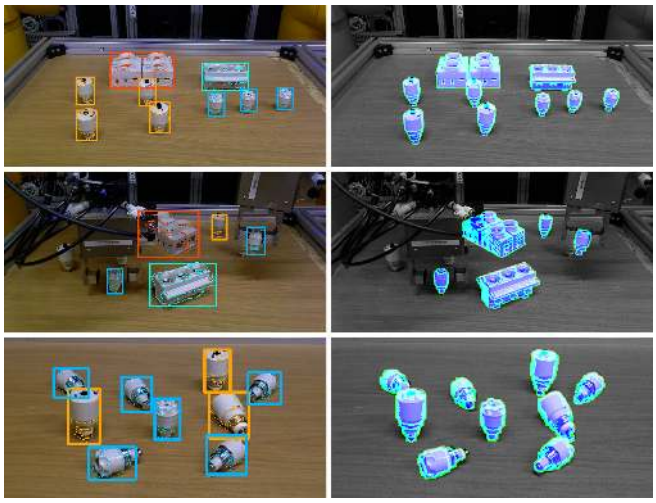

Fig. 1. *Left:* Detection of multiple instances of multiple texture-less objects (fuses and fuse boxes of different types) in a robotic assembly task. *Right:* Superimposed renderings of object models at the estimated 3D poses.

While object recognition is a long-standing and widely studied problem, most attention until recently has been paid to the recognition of textured objects, for which discriminative appearance features, invariant to changes in pose and illumination, can be readily extracted [1]. These objects are often assumed to have piece-wise planar surfaces. Their appearance variations can be therefore modeled by a simple geometric transformation (*e.g.* similarity), which can be reliably determined from the rich textural information. Candidate object locations in the scene are typically determined by identifying so-called interest points or interest regions [2], a strategy which drastically reduces the overall computational cost compared to exhaustive image search. However, when applied to texture-less objects, interest point detectors typically fail to identify corresponding image regions and common local appearance descriptors are no longer discriminative enough to provide reliable correspondences [3].

Recognition and localization of texture-less objects is challenging in several respects. An object's appearance is dominated by its shape, its material properties and by the configuration of light sources. Unless these are known in

Fig. 2. Evaluation cascade of the proposed method. Note the typical numbers of detection candidates advancing through individual stages (for a template size $108 \times 108\,px$, a VGA input image and a scale space with four larger and four smaller scales with a scaling factor 1.2). A detection candidate is a triplet consisting of a template identifier (object and its orientation), a sliding window scale and a sliding window location.

advance and precisely controlled, it is easier to capture possible object appearances exhaustively rather than attempting to describe them with covariant features. In other words, each object can be represented by hundreds or even thousands of images, called templates, which depict it from multiple viewing angles. Being equivalent to matching an image region to one of the templates or asserting there is no suitable such template (*i.e.* the region corresponds to background), the detection task avoids the need of generalization to unknown transformations.

Existing approaches to the detection of texture-less objects usually proceed by sweeping sliding windows of several discrete sizes over the entire image with a small pixel step, searching for a match against all stored object templates. These methods scale poorly to large numbers of objects and special attention has to be paid to implementation details, otherwise they are too slow for real-time operation.

The proposed method addresses the excessive computational complexity of sliding window approaches and achieves high efficiency by employing a cascade-style evaluation of window locations. Fast filtering is performed first, rejecting quickly most of the locations by a simple salience check. For each remaining location, candidate templates are obtained by an efficient voting procedure based on hashing measurements sampled on a regular grid. This makes the complexity of the method sub-linear in the total number of known objects. The candidate templates are then verified by matching feature points in different modalities. Each template is associated with a training-time pose, *i.e.* a 3D rotation and distance to the camera reference frame origin. Therefore, a successful match against a template provides a rough estimate of the object's 3D location and orientation. As a final step, a stochastic, population-based optimization scheme is applied to refine the pose by fitting a 3D model of the detected object to the input depth map. The pipeline of our method is illustrated in Fig. 2 together with the typical numbers of detection candidates advancing through individual stages.

After reviewing relevant related work in Sec. II, the proposed method is detailed in Sec. III and IV. Sec. V presents experimental results and Sec. VI concludes the paper.

II. Related Work

A. Texture-less Object Detection

Template matching is one of the earliest techniques applied to object detection in images. Traditional approaches typically use only a few stored templates per object, perform a sequential scan of the input image and compute correlation coefficients between each window and the stored templates. A sufficiently high correlation score indicates a successful match. Correlation employs intensity images, image gradients or edges. Invariance is achieved only w.r.t. translation, with little tolerance to misalignments. The interested reader is referred to [4] for a survey.

Due to their low generalization and limited applicability, template-based techniques were for some time out of the mainstream research agenda. Instead, research in object recognition concentrated on approaches based on viewpoint-invariant local features obtained from objects rich in texture [5]. Such approaches require only a small number of training images per object and generalize well to a wide range of possible appearances. As computers became faster and equipped with more memory, template-based methods grew in popularity again. Today it is not unusual to maintain thousands of templates per object, thus capturing varying visual aspects exhaustively.

In recent work, Hinterstoisser et al. [6], [7] have introduced an efficient template matching technique. Instead of a raw image, object templates are represented by a set of carefully selected feature points in different modalities (specifically orientation of intensity gradients and orientation of 3D surface normals). Measurements at feature points are quantized and represented as bit vectors, allowing for fast matching with binary operations. Tolerance to misalignments is achieved by comparing the binarized representation with pixels in a small local neighbourhood. The pose retrieved from the best matching template is used as a starting point for subsequent refinement with the Iterative Closest Point (ICP) algorithm [8]. With data structures optimized for fast memory access and a highly vectorized implementation using special SSE hardware instructions, the method is capable of real-time matching of several thousands of templates. However, its performance is expected to degrade noticeably for large object databases, since its time complexity is linear in the number of loaded templates (around 3000 templates are employed per object). The matching procedure of [6] inspired the verification stage of our proposed method.

An alternative approach to 3D object detection that requires only 3D object models was presented by Drost et al. [9]. During training, all possible pairs of 3D points on a model are described and recorded in a hash table. During detection, sampled pairs of 3D points from the test scene are described and used to vote for corresponding object pose hypotheses. The most voted pose clusters can be then refined with ICP. Choi and Christensen [10] further

augmented the point pair feature with color information. The efficiency and performance of these methods depend directly on the complexity of the 3D scene, which might limit their applicability to real-time applications.

Another class of methods relies solely on intensity edges, *e.g.* [11], [12], [3]. Albeit such methods can operate very fast, their recognition capability is inherently lower compared to methods also taking into account depth information. Cai et al. [11] employed a sliding window approach with hypothesis generation based on hashing distances and orientations of the nearest edges from points on a fixed regular grid. We use a similar hashing scheme in the proposed method.

B. 3D Pose Estimation

A common aspect of the approaches mentioned in Sec. II-A is that the pose associated with the detected object is approximate. This is due to the limited resolution of the pose sampling process employed in training or possible mismatches, and necessitates the refinement of the retrieved pose with a geometric optimization step. The ICP algorithm [8] is the most common choice for this purpose. ICP represents the gold standard method for geometrically aligning two sets of points whose relative pose is approximately known. However, when the two point sets are relatively far apart or have a small overlap, ICP's strategy of matching closest points generates large numbers of incorrect correspondences. The situation is aggravated by the inevitable presence of noise and outliers. As a result, ICP can easily get stuck in local minima and its performance largely depends on the quality of initialization. To counter this, numerous enhancements to the basic ICP have been proposed that aim to improve the speed of convergence or increase robustness to local minima, outlying points and noise [13]. These enhancements often require considerable trial and error for tuning their parameters to a particular application. Here we take a different approach and refine 3D pose with an optimization scheme based on Particle Swarm Optimization (PSO) [14]. PSO has proven to be an effective framework for dealing with other flavors of pose estimation, *e.g.* [15], [16].

III. Detection of Texture-less Objects

Detection of objects in an input RGB-D image is based on a sliding window approach, operating on a scale pyramid built from the image. Let $\mathcal{L}$ denote the set of all tested locations. The number of locations $|\mathcal{L}|$ is a function of image resolution, spatial image sampling by the sliding window (*e.g.* every 5 pixels), scale range (*e.g.* two or four octaves), and scale space discretisation. The known objects are represented with a set $\mathcal{T}$ of template images – RGB-D images of a fixed size. There are several thousands of templates per object, capturing its appearance from all possible viewing angles, but from a fixed distance. The training distance, which then affects the depth channel of an RGB-D template, is object-specific but fixed for all templates of a certain object. This distance is chosen so that the object would optimally fill the template image if observed with a camera with identical intrinsic parameters (focal length and resolution) as the camera observing later the test scene. Each template is associated with the object ID, the training distance Z_t and the object orientation $\mathbf{R}_0$ it represents.

In general, every window $w_\mathbf{l}$, $\mathbf{l} = (x, y, s)$, $\mathbf{l} \in \mathcal{L}$, needs to be tested against every template, which makes the asymptotic complexity $\mathcal{O}(|\mathcal{L}||\mathcal{T}|)$. This is computationally very demanding even for moderate numbers of known objects. We therefore propose a cascaded evaluation, where the set of candidate locations $\mathcal{L}$ is quickly reduced (Sec. III-A) and the candidate templates $\mathcal{T}$ are pruned (Sec. III-B) before the template matching itself (Sec. III-C) is performed.

A. Pre-filtering of Window Locations

To reduce the number of image locations, an image window is first assessed with a simple *objectness* measure [17], [18], *i.e.* its likelihood that it contains any of the objects. This corresponds to a two-class classifier distinguishing between background and object classes, with the object class encompassing all the templates in $\mathcal{T}$.

Our objectness measure is based on the number of depth-discontinuity edgels within the window, and is computed with the aid of an integral image for efficiency. Depth-discontinuity edgels arise at pixels where the response of the Sobel operator, computed over the depth image, is above a threshold θ_e, which is set to 30% of the physical diameter of the smallest object in the database. The window is classified as containing an object if its number of depth edgels is at least 30% of the number of depth edgels in the template containing the least amount of them. This setting is tolerant to partial occlusions but still strong enough to prune most of the window locations – roughly 90% to 99% of them in images of our robot workspace (Fig. 1), depending on the scene clutter. Only image windows that pass the objectness test are processed further.

B. Hypothesis Generation

In this phase, a small subset of candidate templates is quickly identified for each image window that passed the objectness test. Up to N templates with the highest probabilities $p_t(t|w_\mathbf{l})$, $t \in \mathcal{T}$ are retrieved. This can be seen as a multi-class classification problem where there is one class for each training template, but none for the background.

The procedure retrieves candidate templates from multiple (for robustness) trained hash tables, which is a constant complexity $\mathcal{O}(1)$ operation in the number of stored templates. Each hash table $h \in \mathcal{H}$ is indexed by a trained set $\mathcal{M}_h$ of measurements taken on the window $w_\mathbf{l}$ or template t, discretized into a hash key. $\mathcal{M}_h$ is different for each table. The table cells contain lists of templates with the same key, the lists are then used to vote for the templates. A template can receive up to $|\mathcal{H}|$ votes, in which case all the template's measurement sets (for all the tables) would be discretised to the same hash keys as measurements on the window $w_\mathbf{l}$. Up to N templates with the highest number of votes, and with at least v votes, are passed onward to the next step of the detection cascade. The voting is still an $\mathcal{O}(|\mathcal{T}|)$ operation for each $w_\mathbf{l}$.

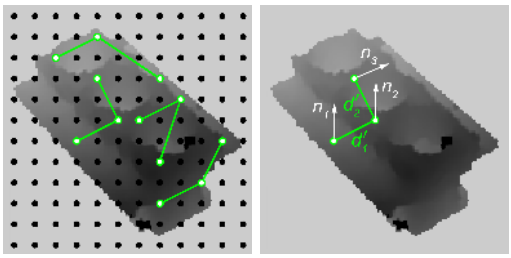

Fig. 3. Templates and test windows are hashed using measurements from trained triplets of grid points. *Left:* Sample triplets which are valid for the shown template, *i.e.* their points lie in the object mask. *Right:* A triplet is described by depth differences $\{d'_1, d'_2\}$ and normal vectors $\{\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3\}$.

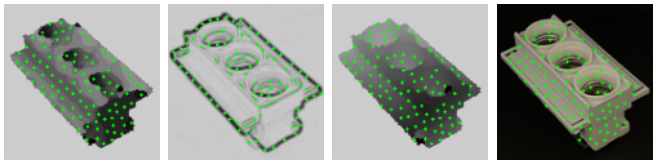

Fig. 4. Feature points in different modalities whose consistency is evaluated in the hypothesis verification. The points are trained independently for each template. *Left to right:* surface normals, image gradients, depth, color.

Measurement sets $\mathcal{M}_h$ and their quantization. The hashing/voting procedure was inspired by the work of Cai et al. [11]. A regular grid of 12×12 reference points is placed over the training template or sliding window. This yields 144 locations from which k-tuples are sampled. We use triplets in our setup, *i.e.* $k = 3$ (Fig. 3). Each location is assigned with a depth d and a surface normal $\mathbf{n}$. A measurement set $\mathcal{M}_h$ is a vector consisting of $k-1$ relative depth values and k normals, $\mathcal{M}_h = (d_{2_h} - d_{1_h}, d_{3_h} - d_{1_h}, \ldots, d_{k_h} - d_{1_h}, \mathbf{n}_{1_h}, \ldots, \mathbf{n}_{k_h})$. The relative depths $d_{i_h} - d_{1_h}$ are quantized into 5 bins each, with the quantization boundaries learned from all the training templates to provide equal frequency binning, *i.e.* each bin contains the same number of templates. To quantize surface normals we use the approach proposed in [7], where the normals are quantized to 8 discrete values based on their orientation. For triplets of reference points we have two relative depths and three normals, leading to a hash table size of $5^2 8^3 = 12800$ bins.

Training-time selection of measurement sets. To provide robustness to occlusion and noise, multiple hash tables are built, which differ in the selection of the k-tuples drawn from the 144 reference points. The k-tuples can be chosen randomly. Alternatively, they can be optimally selected to (a) cover maximally independent measurements (for robustness), and (b) to fill the tables as uniformly as possible (for stable detection time). The optimal selection is unfortunately NP-complete, therefore we employ a hybrid heuristic strategy. We first randomly generate a set of m, $m \gg |\mathcal{H}|$, k-tuples and then retain the subset with the largest joint entropy of the quantized measurements.

For the results reported below, $|\mathcal{H}| = 100$ hash tables were employed, chosen from $m = 5000$. The minimal number of votes per template was $v = 3$ and the maximum number of candidates passed to the verification stage was $N = 100$.

C. Hypothesis Verification

The verification stage corresponds to the traditional template matching. Thanks to the template selection in the previous step, only up to N templates are considered for each image window $w_\mathbf{I}$ that passed the initial objectness test. This makes the complexity of this stage constant in the number of stored templates. Since the templates were already identified in the previous step, the verification can be seen as a set of up to N separate two-class classification problems discriminating between the object represented by a template and the background class, *i.e.* $p(\text{obj}|t_i, w_\mathbf{I}) \lessgtr p(\text{bkg}|t_i, w_\mathbf{I})$.

The verification proceeds in a sequence of tests evaluating the following: **I** object size in relation to distance, **II** sampled surface normals, **III** sampled image gradients, **IV** sampled depth map, and **V** sampled color. The tests are ordered according to increasing computational cost. Any failed test classifies the window as non-object – corresponding to either the background, or an object not represented by template t_i – and subsequent tests are not evaluated.

Test **I** verifies that the observed object size (*i.e.* the level of the scale pyramid) corresponds to its distance measured in the depth map. The object is expected at distance Z_e calculated as $Z_e = Z_t s$, where s is the scale factor of the pyramid level and Z_t is the template's training distance. If the measured depth Z_w is within the interval $|Z_e/\sqrt{f}, Z_e \cdot \sqrt{f}|$, where f is the discretization factor of the scale space, the depth Z_w is considered to be feasible, otherwise the test fails.

Tests **II** and **III** verify orientation of surface normals and intensity gradients at several feature points. Following [19], the point locations are greedily extracted during the training stage, independently for each template (Fig. 4). The feature points for the surface normal orientation test are extracted at locations with locally stable orientation of normals (*i.e.* further away from depth discontinuities). For the intensity gradient orientation test, the feature points are extracted at locations with large gradient magnitude (*i.e.* typically on the object contour). We extract 100 points in both cases. The orientations are quantized and compared template-against-sliding window, which can be done very fast by bitwise operations using response maps described in [6].

The depth map test **IV** and the color test **V** reuse the locations of feature points extracted for the surface normal test **II**. In the depth test, difference d between the depth in the template and the depth in the window is calculated for each feature point. A feature point is matched if $|d - d_m| < kD$, where d_m is the median value of ds over all feature points, D is the physical object diameter, and k is a coefficient (set to 0.05 in our experiments). Finally, pixel colors in test **V** are compared in the HSV space, as done in [19].

A template passes the tests **II** to **V** if at least θ_c of the feature points have a matching value within a small neighbourhood. In our experiments $\theta_c = 60\%$ to tolerate partial occlusions, and the extent of the local neighborhood is 5×5 pixels to compensate for the sliding window step, and for the discretization of orientations during training. A verified template that passes all the tests is assigned a final

score computed as $m = \sum_{i \in \{\mathbf{II} \ldots \mathbf{V}\}} c_i$, where c_i is the fraction of matching feature points in tests **II** to **V**.

D. Non-maxima Suppression

The verified templates are accumulated from all different locations and scales. Since different views of one object are often alike, and since multiple objects may by rather similar, unique detections are identified by repeatedly retaining the candidate with the highest score r, and removing all detections that have a large overlap with it. The score is calculated as $r = m(a/s)$, where m is the verification score defined above, s is the detection scale, and a is the area of the object in the considered template. Weighting the score by the object area favours detections which explain more of the scene (*e.g.* when a cup is seen from a side, with the handle visible, we prefer a template depicting the handle over other templates where the handle is occluded, but which would otherwise yield the same matching score). The retained detections are passed to the 3D pose estimation stage, together with the approximate 3D poses which have been associated with the training templates.

IV. Fine 3D Pose Estimation

Fine 3D pose estimation refers to the accurate computation of translation and rotation parameters that define an object's position and orientation in space, assuming that approximate initial values for these parameters are provided. This process receives as inputs a mesh model of the object, an initial object pose $\{\mathbf{R}_0,\ \mathbf{t}_0\}$, a depth image and the sensor's intrinsics and outputs a refined pose $\{\mathbf{R},\ \mathbf{t}\}$. Objects are represented with arbitrary 3D mesh models, which can originate from CAD drawings or from digital scans. A mesh $\mathcal{M}$ is comprised of an ordered set of 3D vertex points V and an ordered set G of triplet indices upon V that define the mesh triangles. The 3D oriented bounding box B of each model is precomputed using the eigenvectors of V's covariance matrix.

Candidate poses $\{\mathbf{R}_i, \mathbf{t}_i\}$ are generated and then evaluated by using them to synthesize renderings of $\mathcal{M}$, producing depth images S_i (see Sec. IV-B). A scoring function yields score $o(i)$, which quantifies the similarity between each image S_i and the input using depth, edge and orientation cues (Sec. IV-C). PSO is used to optimize the scoring function and find the pose whose rendered depth image is the most similar to the input one (Sec. IV-D). An overview of the approach is provided in Fig. 5 whereas its components are briefly described in the following subsections. A detailed presentation and evaluation of our pose estimation pipeline can be found in [20].

A. Initialization

The initial pose used to bootstrap pose estimation can be quite crude. The projection on the sensor of the model at the initial pose determines a 2D, axis-aligned bounding box b. This box is inflated proportionally to the distance of the initial pose to mitigate any pose inaccuracies and is assumed to enclose most of the projection of the target object on the input image.

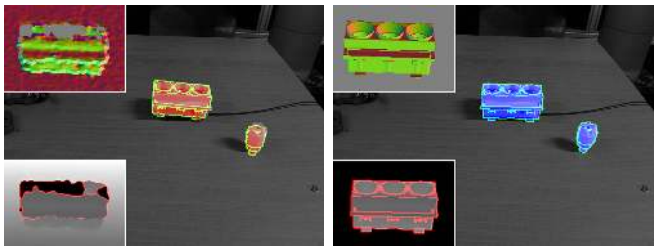

Fig. 5. Pose estimation for an electrical fuse and fuse box. *Left:* Initial poses superimposed on a captured image (their misalignment can be seen by zooming in). Thumbnails show in magnification captured depths and detected edges (bottom), along with color-coded surface normals (top) in the region of the fuse box. *Right:* Refined poses. Thumbnails show corresponding fuse box depths and surface normals obtained by rendering.

To suppress sensor noise, the acquired depth image is median filtered with a 5×5 kernel and the result is retained as image D. Depth shadows and other shortcomings of consumer depth sensors manifest themselves as invalid pixels in D, not contributing with 3D points. Surface normals for valid depth pixels are estimated by local least-squares plane fitting and stored in N [21]. Binary image E is computed from D, by thresholding the output of the Sobel operator applied to D. The distance transform T of E is computed for later use. The above computations are parallelized on a GPU at the pixel level, while the computation of T uses the parallel formulation of [22]. To evaluate an input frame, only the depth image D is uploaded to the GPU which then uses it to compute N and T. No other input data exchange with the GPU occurs during pose estimation.

B. Pose Hypotheses Rendering

A rendering process simulates depth images of the target object at a hypothesized pose against a blank background. Pose rendering is formulated as follows. Transform $\{\mathbf{R}_0, \mathbf{t}_0\}$ brings the model in an approximate location and orientation, in the depth sensor's reference frame. Candidate poses are parametrized relative to this initial pose, using a relative translation $\mathbf{t}_i$ and an "in place" rotation $\mathbf{R}_i$ about the centroid $\mathbf{c}$ of points in V. Specifically, the model is first translated by $-\mathbf{c}$, rotated by $\mathbf{R}_i$, and translated back to place by $\mathbf{c}$. Rotation $\mathbf{R}_i$ is the product of primitive rotations about the 3 axes: $\mathbf{R}_i = \mathbf{R}_x(\theta_i) \cdot \mathbf{R}_y(\phi_i) \cdot \mathbf{R}_z(\omega_i)$. The transformation model point $\mathbf{x}$ undergoes is thus $\mathbf{R}_i \cdot (\mathbf{x} - \mathbf{c}) + \mathbf{c} + \mathbf{t}_i$. To avoid repeated transformations, the initial and candidate poses are combined into the following:

$$\mathbf{R}_i \cdot \mathbf{R}_0 \cdot \mathbf{x} + \mathbf{R}_i \cdot (\mathbf{t}_0 - \mathbf{c}) + \mathbf{c} + \mathbf{t}_i. \quad (1)$$

The rotational component of candidate poses is parameterized using Euler angles whereas their translation is parameterized with Euclidean coordinates. The model transformed according to Eq. (1), is rendered in depth image S_i. Depth edges and surface normals of S_i are computed and stored in binary image E_i and data structure N_i, respectively.

Computation and storage of S_i, E_i, and N_i is delegated to the GPU. The process employs Z-buffering to respect visibility and realistically render self-occlusions. Parallelization

is performed at two levels of granularity. At a fine level, rendering is parallelized upon the triangles of the rendered mesh. At a coarser level, multiple hypotheses are rendered simultaneously, with a composite image gathering all renderings. In this manner, multiple hypotheses are evaluated in a single batch, resulting in better utilization of GPU resources and reduced communication. Edge detection is applied once, directly upon the composite image.

C. Pose Hypotheses Evaluation

A candidate pose is evaluated with respect to the extent to which it explains the input depth image. Objective function $o(\cdot)$ avails a score $o(i)$ and considers the similarity of depth values, surface normals, as well as depth edges between D and S_i. Two range images are corresponded in terms of their coordinates and are compared as follows.

Depth values are directly compared between D and S_i for pairs of pixels. For n pixel pairs, depth differences δ_k, are computed and the cumulative depth cost term is defined as:

$$d_i = \sum_{k=1}^{n} 1/(|\delta_k| + 1), \tag{2}$$

where $|\delta_k|$ is set to ∞ if greater than threshold d_T ($20\,mm$ in our implementation) to avoid comparing with background surfaces. For the same n pairs of pixels, the cost due to surface normal differences is quantified as:

$$u_i = \sum_{k=1}^{n} 1/(|\gamma_k| + 1), \tag{3}$$

where γ_k is the angle between the two surface normals, provided by their dot product. Edge differences are aggregated in an edge cost using E and E_i. Let m be the number of edgels of E_i within b. For each such edgel j, let ϵ_j denote the distance from its closest edgel in D which is looked up from T. The corresponding edge cost term is then:

$$e_i = \sum_{j=1}^{m} 1/(\epsilon_j + 1). \tag{4}$$

Each of the cost terms in Eqs. (2), (3) and (4) involves two ordered pixel sets, one from each image D and S_i, that contain the pixel locations to be compared. As d_i, u_i, and e_i have different numeric ranges, the combined cost is defined by their product $o(i) = -d_i \cdot e_i \cdot u_i$, where the minus sign is used to ensure that optimal values correspond to minima, since d_i, e_i and u_i are non-negative. Summing the reciprocals of partial differences $|\delta_k|$, $|\gamma_k|$ and ϵ_j rewards poses that maximize the support (*i.e.* spatial overlap) between the compared regions of D and S_i. The objective function improves when more pixels in the rendered depth map closely overlap with the imaged surfaces in the input image.

As no segmentation is employed, inaccurate pose hypotheses might cause the rendered object to be compared against pixels imaging background or occluding surfaces. To counter this, only pixels located within b are considered. Hypotheses that correspond to renderings partially outside b obtain a poor similarity score and the solution does not drift towards an irrelevant surface. Also, during the evaluation of each hypothesis, the oriented bounding box B_i that corresponds to hypothesis i is computed by transforming B according to Eq. (1). By so doing, depth pixels from D that correspond to 3D points outside B_i are not considered in the comparison, as they are irrelevant to the hypothesis being evaluated.

D. Pose Estimation

The search space for the pose estimation is constrained in a 6D neighborhood of the initial pose estimate. Each dimension of the pose search space is bounded, defining a search hyperrectangle centered on the initial pose estimate. As the cost of an exhaustive search in $\mathcal{R}^6$ is prohibitive, a numerical optimization approach is adopted to minimize objective function $o(\cdot)$. This minimization is performed with PSO, which stochastically evolves a population of candidate solutions dubbed particles, that explore the parameter space in runs called generations. PSO does not require knowledge of the derivatives of the objective function, depends on very few parameters and requires a relatively small number of objective function evaluations until convergence. Compared to gradient-based optimization methods, PSO has a wider basin of convergence, exhibiting better robustness to local minima. Furthermore, as particles evolve independently at each generation, it is amenable to an efficient parallel implementation [20].

V. Experiments and Evaluation

A. Object Localization

The presented method was evaluated quantitatively with the aid of the publicly available dataset by Hinterstoisser et al. [19]. This dataset includes 15 texture-less objects and provides for each a 3D mesh model and a test sequence consisting of approximately 1200 RGB-D frames in VGA resolution. The test sequences feature heavy 2D and 3D clutter, mild occlusions and large viewpoint variations and are accompanied by the ground truth object pose for each frame. The task is to localize the given object in each frame, *i.e.* to detect it and estimate its 3D pose.

Training templates were rendered from the provided 3D models so that they uniformly covered the upper view hemisphere (with a step of 10° in both azimuth and elevation). To achieve invariance to rotation around the optical axis, an in-plane rotation to each template (from -40° to 40° with a step of 10°) was also applied. In total, each object was represented by 2916 templates of size $108 \times 108\,px$. Each test image was scanned at 9 scales (4 larger and 4 smaller scales with scaling factor 1.2) with a scanning step of $5\,px$.

We compare our method to the LINEMOD [6] and LINEMOD++ [7] methods of Hinterstoisser et al. and the method of Drost et al. [9]. These methods were already briefly described in Sec. II-A; here we provide more details regarding their relation to our method. LINEMOD follows an exhaustive template matching approach. LINEMOD++ extends it by two post-processing verification steps. Specifically, a color check and a depth check (by a rough but fast ICP) are performed in order to prune hypotheses. A finer ICP is then carried out for the best of the remaining hypotheses. The essential difference of our method is the addition of the

Sequence	Our method	LINEMOD++	LINEMOD	Drost et al.
1. Ape	93.9	**95.8**	69.4	86.5
2. Benchvise	**99.8**	98.7	94.0	70.7
3. Bowl	98.8	**99.9**	99.5	95.7
4. Box	**100.0**	99.8	99.1	97.0
5. Cam	95.5	**97.5**	79.5	78.6
6. Can	**95.9**	95.4	79.5	80.2
7. Cat	98.2	**99.3**	88.2	85.4
8. Cup	**99.5**	97.1	80.7	68.4
9. Driller	**94.1**	93.6	81.3	87.3
10. Duck	94.3	**95.9**	75.9	46.0
11. Glue	**98.0**	91.8	64.3	57.2
12. Hole punch	88.0	**95.9**	78.4	77.4
13. Iron	97.0	**97.5**	88.8	84.9
14. Lamp	88.8	**97.7**	89.8	93.3
15. Phone	89.4	**93.3**	77.8	80.7
Average	95.4	**96.6**	83.0	79.3

TABLE I

RECOGNITION RATES [%] FOR THE DATASET OF [19] AND $k_m = 0.1$ (*i.e.* THE PERCENTAGE OF OBJECTS LOCALIZED WITH AN ERROR SMALLER THAN 10% OF THEIR DIAMETER).

pre-filtering and the hypothesis generation stage, avoiding the exhaustive search.

We used the same quantification of pose error as in [19]. That is, for the ground truth pose $\{\mathbf{R}_g, \mathbf{t}_g\}$ and the estimated pose $\{\mathbf{R}_e, \mathbf{t}_e\}$, the error is $e = 1/\nu \sum_i |\mathbf{g}_i - \mathbf{e}_i|$, where $\mathbf{g}_i = \mathbf{R}_g x_i + \mathbf{t}_g$, $\mathbf{e}_i = \mathbf{R}_e x_i + \mathbf{t}_e$, and i enumerates the ν vertices of V. For objects with ambiguous pose due to their symmetry (namely "Cup", "Bowl", "Box" and "Glue"), the error is computed as $e = 1/\nu \sum_i min_j |\mathbf{g}_i - \mathbf{e}_j|$. An object is considered to be correctly localized if $e \leq k_m d$, where k_m is a fixed coefficient and d is the diameter of the model, *i.e.* the maximum distance between any of its vertices.

As in the methods being compared, the best detection of the object of interest was selected and evaluated in each frame. For the detected template with the highest matching score, the corresponding initial 3D pose was refined by our pose estimation method and the error of the resulting pose was calculated as explained above. Table I compares the recognition rates (for $k_m = 0.1$) of our method with the rates of the other methods which were published in [19]. Our method achieved an average recognition rate of 95.4% (*i.e.* the percentage of correctly localized objects) and outperformed LINEMOD and the method of Drost et al. The average recognition rate achieved by LINEMOD++ is better by 1.2%. Recognition rates of our method with respect to different values of k_m can be found in Fig. 6 (top). The benefit of the pose estimation stage can be seen in Fig. 6 (middle), where the average pose errors of the initial and the refined poses are compared. Pose refinement employed PSO with parallelized hypotheses rendering (cf. Sec. IV-B). With 100 particles and 100 generations it required around $0.19\,s$ per frame to estimate the pose of a single model with $\approx 7K$ triangles. For comparison, when hypotheses were evaluated sequentially on the GPU, PSO required $4.8\,s$. In [20], we show that our PSO-based pose estimation delivers superior results compared to the commonly used ICP. Visualizations

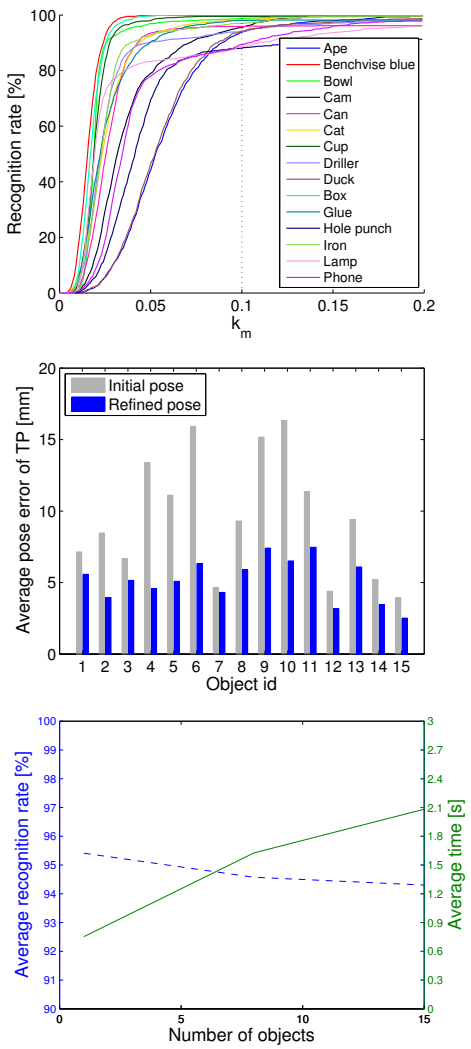

Fig. 6. *Top:* Recognition rates for various k_m. *Middle:* Average errors of initial and refined poses of true positives for $k_m = 0.1$ (object names corresponding to the numbers can be found in Table I). *Bottom:* Average recognition rate (dashed line) and average recognition time w.r.t. the number of loaded object templates (there were 2916 templates per object).

of sample results are in Fig. 7.

To evaluate scalability, we also run our method when templates of 8 and 15 objects were loaded for detection. As can be seen in Fig. 6 (bottom), the complexity of our method was proven sub-linear in the number of templates ($0.75\,s$ vs. $2.08\,s$ when templates of 1 and 15 objects were loaded) while the recognition rate drops only slightly (by only 1% when the 43740 templates of all 15 objects were loaded w.r.t. the case when only 2916 templates of a single object were loaded). The sub-linearity is achieved by the hypothesis generation stage which allows the comparison of only a small set of templates for each window location.

In terms of running time, the whole method needed on average $0.75\,s$ per VGA frame. This time was achieved by our parallelized C++ implementation on a modern desktop PC equipped with a 16 core CPU and an NVIDIA GTX 780 GPU (the GPU was only employed during the pose

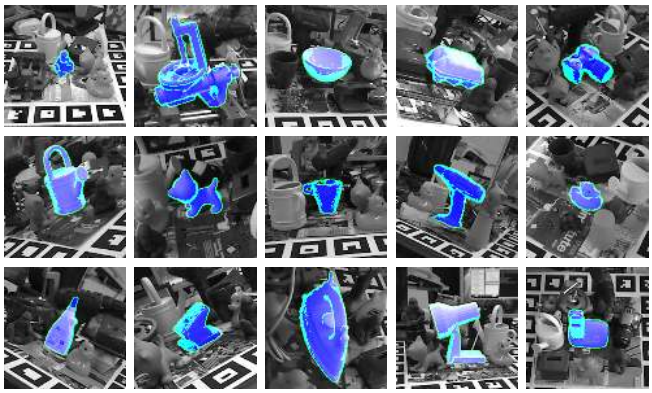

Fig. 7. Sample 3D pose estimations on the dataset of [19] (cropped).

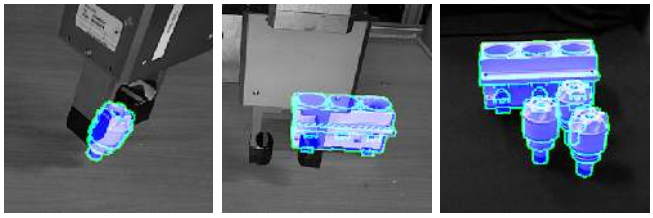

Fig. 8. Cropped close-ups of 3D pose estimations in a manipulation task.

estimation). As reported in [19], the method of Drost needed on average $6.3\,s$ and LINEMOD++ only $0.12\,s$. The latter time was achieved with a highly optimized implementation using heavy SSE parallelization and a limited scale space (for each object, only a limited set of scales were considered). With a similar level of optimization, we expect our method to run even faster since it evaluates only a small set of templates for each window. In the case of multiple object detection and localization that arises often in robotic applications, our method is expected to outperform LINEMOD++ in terms of running time.

B. Robotic Application

The intended application of the proposed method relates to robotic manipulation and assembly of objects. Figs. 1 and 8 demonstrate its suitability for a manipulation task with electrical parts along with its tolerance to mild occlusions. When a robotic gripper was present in the scene, its posture was accurately provided by motor encoders and used to mask out the corresponding pixels in the image, preventing them from contaminating pose estimation. As shown in Fig. 6 (middle), the poses estimated with the presented method achieve a sub-centimeter average accuracy in the estimated pose (for the recognition of 95.4%). This meets the requirements imposed by the compliant grippers of the industrial robotic arms used in our application (Stäubli RX130 and RX90).

VI. Conclusion

An approach for texture-less object detection and 3D pose estimation in RGB-D images has been presented. It is based on a sliding window approach with a cascade-style evaluation of each window location. Sub-linearity in the number of training templates is achieved by an efficient voting procedure based on hashing which generates a small set of candidate templates for each window location. The templates are verified in several modalities and approximate object poses associated with the matched templates are refined by a stochastic optimization scheme. Experiments on a public dataset have demonstrated that the proposed method detects objects with a recognition rate comparable to the state of the art and achieves sub-centimeter accuracy in localization. The method is therefore well-suited to multiple object detection, a commonly required task in robotic applications.

References

[1] A. Collet, M. Martinez, and S. Srinivasa, "The MOPED framework: Object Recognition and Pose Estimation for Manipulation," *I. J. Robotic Res.*, vol. 30, no. 10, pp. 1284–1306, 2011.

[2] T. Tuytelaars and K. Mikolajczyk, "Local invariant feature detectors: A survey," *Found. Trends. Comput. Graph. Vis.*, vol. 3, no. 3, pp. 177–280, July 2008.

[3] F. Tombari, A. Franchi, and L. Di, "BOLD features to detect texture-less objects," in *ICCV*, 2013, pp. 1265–1272.

[4] R. Brunelli, *Template Matching Techniques in Computer Vision: Theory and Practice*. Wiley Publishing, 2009.

[5] D. Lowe, "Object recognition from local scale-invariant features," in *ICCV*, vol. 2, 1999, pp. 1150–1157.

[6] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit, "Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes," in *ICCV*, 2011.

[7] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit, "Gradient response maps for real-time detection of texture-less objects," *IEEE PAMI*, 2012.

[8] P. Besl and N. McKay, "A Method for Registration of 3-D Shapes," *PAMI*, vol. 14, no. 2, pp. 239–256, 1992.

[9] B. Drost, M. Ulrich, N. Navab, and S. Ilic, "Model globally, match locally: Efficient and robust 3d object recognition," in *CVPR*, 2010, pp. 998–1005.

[10] C. Choi and H. Christensen, "3D Pose Estimation of Daily Objects Using an RGB-D Camera," in *IROS*, 2012, pp. 3342–3349.

[11] H. Cai, T. Werner, and J. Matas, "Fast detection of multiple textureless 3-D objects," in *ICVS*, ser. LNCS, 2013, vol. 7963, pp. 103–112.

[12] D. Damen, P. Bunnun, A. Calway, and W. Mayol-Cuevas, "Real-time Learning and Detection of 3D Texture-less Objects: A Scalable Approach," in *BMVC*, Sep 2012, pp. 1–12.

[13] S. Rusinkiewicz and M. Levoy, "Efficient Variants of the ICP Algorithm," in *3DIM*, 2001, pp. 145–152.

[14] R. Poli, J. Kennedy, and T. Blackwell, "Particle Swarm Optimization," *Swarm Intelligence*, vol. 1, no. 1, pp. 33–57, 2007.

[15] I. Oikonomidis, N. Kyriazis, and A. A. Argyros, "Efficient model-based 3D tracking of hand articulations using Kinect," in *BMVC*, 2011, pp. 1–11.

[16] S. Ivekovič, E. Trucco, and Y. Petillot, "Human Body Pose Estimation with Particle Swarm Optimisation," *Evolutionary Computation*, vol. 16, no. 4, pp. 509–528, 2008.

[17] B. Alexe, T. Deselaers, and V. Ferrari, "Measuring the objectness of image windows," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 11, pp. 2189–2202, Nov 2012.

[18] M.-M. Cheng, Z. Zhang, W.-Y. Lin, and P. Torr, "BING: Binarized normed gradients for objectness estimation at 300fps," in *CVPR*, 2014, pp. 3286–3293.

[19] 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 *ACCV*, 2012.

[20] X. Zabulis, M. Lourakis, and P. Koutlemanis, "3D object pose refinement in range images," in *ICVS*, ser. LNCS, 2015, vol. 9163, pp. 263–274.

[21] H. Badino, D. Huber, Y. Park, and T. Kanade, "Fast and accurate computation of surface normals from range images," in *ICRA*, 2011, pp. 3084–3091.

[22] T.-T. Cao, K. Tang, A. Mohamed, and T.-S. Tan, "Parallel Banding Algorithm to Compute Exact Distance Transform with the GPU," in *I3D*, 2010, pp. 83–90.