SEVENTH FRAMEWORK PROGRAMME
ICT Priority

Deliverable D3.6: Integration of Appearance and Shape Based Object Recognition Methods

Due date of deliverable: 31.05.2014
Actual submission date: 02.06.2014
Updated on: 10.11.2014

Start date of project: 1st February 2011
Duration: 48 months

Coordinator:
Dr. Christian Eitzinger
Profactor GmbH

Lead Beneficiary: CVUT
Contributions by: FORTH

Project co-funded by the European Commission within the 7th Framework Programme

<table>
<thead>
<tr>
<th>Dissemination Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>PU</td>
</tr>
<tr>
<td>PP</td>
</tr>
<tr>
<td>RE</td>
</tr>
<tr>
<td>CO</td>
</tr>
</tbody>
</table>
Contents

1. Introduction ........................................................................................................... 3
2. Texture-less Object Detection ............................................................................. 3
   2.1. Related Work ................................................................................................. 4
   2.2. Basic Decision Cascade ............................................................................... 4
   2.3. Decision Cascade Extended by ICP Procedure ............................................. 6
   2.4. Adding Depth Information ............................................................................ 7
   2.5. Parallel Implementation ............................................................................... 8
3. Fine 3D Pose Estimation ....................................................................................... 9
4. Integrated Vision Module ..................................................................................... 9
   4.1. Architecture of the Vision Module ............................................................... 9
   4.2. Acquisition of Training Data ...................................................................... 11
5. Experimental Evaluation ..................................................................................... 13
6. Conclusions .......................................................................................................... 17
1. Introduction

This deliverable describes an approach for fast detection and pose estimation of texture-less objects. Objects with little or no texture are very common in robotic applications. However, they pose a challenge to existing visual object detection approaches which typically rely on view-invariant features extracted from local textured regions. Such extraction is neither robust nor repeatable when there is little texture and impossible in the case of no texture. We therefore adopt template-based object representation that capture possible object appearance exhaustively. The detection of texture-less objects is addressed by an efficient template matching scheme followed by a fine 3D pose estimation step.

The requirements from Task 3.7 are met by integrating appearance-based and shape-based object representations. Specifically, the current version of the object detector is based not only on 2D features but uses also 3D features to increase robustness and discriminative power. Moreover, the adoption of 3D information enables the vision system to handle texture-rich objects for which the original purely edge-based object detector is not suitable.

The presented approach forms the core of DARWIN vision module which, as the main outcome of WP3, provides the robot with a functionality to detect and localize objects in its workspace. The vision module assumes a stream of RGB-D (color + depth) frames from a Kinect-like sensor. The module has two main parts, the object detection submodule and the pose estimation submodule. The former performs detection and approximate pose estimation, where the objects are represented by a set of templates, each showing one object in one 3D pose. For every scanning window in the input image, the detection submodule efficiently finds the most similar template or asserts there is no such template (background). Initialized with a rough 3D pose, obtained from the detected template, the pose estimation submodule then refines the 3D pose by fitting the 3D model to the input depth map, as was described in detail in Deliverable 3.5.

As is shown experimentally, the proposed approach is comparable with the-state-of-the-art methods. The resulting accuracy follows requirements given by the DARWIN user stories. The industrial relevance is further increased by acquisition of training data from CAD models, which is presented as a viable alternative to real training data captured by an RGB-D sensor.

The deliverable is organized as follows. In Section 2, we describe our progress in object detection since D3.4, leading to better accuracy and speed. In Section 3, we recall our pose estimation method. Section 4 describes the integrated vision module, its architecture, implementation and the process of acquisition of training data. Section 5 then presents an experimental evaluation. In Section 6 we conclude and discuss the achieved results.

2. Texture-less Object Detection

This section summarizes our progress on texture-less object detection. The progress is described relatively to the method presented in Deliverable 3.4 and then published in [1]. Section 2.1 provides a review of a work relevant to our recent progress and to the evaluation presented later in Section 5. Section 2.2 briefly recalls the original edge-based detector and Section 2.3 describes its extension by an ICP procedure. A modification of our method which utilizes depth information is presented in Section 2.4 and implementation details in Section 2.5.
2.1. Related Work

Template matching is one of the earliest techniques applied to object detection in images. Traditional approaches would use one or only a few stored templates per object, perform a sequential scan of an input image, window by window, computing correlation coefficients between a window and the stored templates. A sufficiently high correlation indicates a match. Correlated are intensity images, image gradients or edges. Invariance is achieved only w.r.t. translation, with only a little tolerance for misalignments. See e.g. [10] for a survey.

Because of their low generalization and limited applicability, template-based techniques were for some time out of the mainstream research interest. Research in object recognition rather concentrated on approaches based on viewpoint-invariant local features on texture-rich locally planar objects. 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, capturing varying visual aspects exhaustively. These methods see an increased popularity especially in situations where the local-feature methods are failing, in particular in detection of texture-less objects.

Recent work by Hinterstoisser et al. [7, 8] introduced an efficient implementation for matching a large numbers of templates. Instead of a raw image, object templates are represented by a set of points of interest. These points are selected at locations with significant intensity gradients, in a manner similar to the Harris operator [11]. Image gradient orientations at the points of interest, or surface normals when depth information is available, are discretized into eight bins 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. With data structures optimized for fast memory access and with vectorized implementation using special SSE hardware instructions, the method is capable of real-time matching of thousands of templates, even as each candidate image location is tested against every template. The pose retrieved with the best matching template is used as a starting point for subsequent refinement with the aid of ICP. We adopted this template matching paradigm in our recent object detector (see Section 2.4).

An alternative approach to 3D object detection was presented by Drost et al. [9]. It requires only 3D object models. During training, all possible pairs of 3D points on the 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 often voted pose clusters can be then refined with ICP. Choi and Christensen [12] further augment the point pair feature with color information. Efficiency and performance of these methods depend directly on the complexity of the 3D scene, which makes them difficult to use in real-time applications.

There are also object detection methods which rely only on 2D features (usually on intensity edges). These methods were reviewed in Deliverable 3.4.

2.2. Basic Decision Cascade

Here we briefly review our initial approach to texture-less object detection. It was described in D3.4, and after some improvements published in a conference paper [1]
where it was shown to outperform the state-of-the-art edge-based method by Damen et al. [2] (on a public dataset of texture-less objects provided by the same authors).

Figure 1 shows an overview of our approach. There is a database of training images (templates) representing multiple objects in multiple poses. In the input image, the method searches for occurrences of the templates (up to a certain distortion) in all possible positions and scales. Invariance to rotation is achieved by including rotated images in the training set. The test and training images are greyscale (no color information is used) and are represented by edge maps.

![Figure 1. An example of a training set and detections in a test image.](image)

The method works as follows. A scanning window of a fixed size (the same as the size of templates) is placed in the test image at all different positions and scales. The windows are at first pre-filtered by a very fast and simple saliency check (based on the number of intensity edges) which prunes a large portion of background windows. The detector then consists of two stages, where the first stage generates detection hypotheses for each scanning window and the second stage performs their verification. After the second stage, non-maxima suppression picks final detections.

The first stage is based on hashing of sparse edge-based descriptors. To construct the descriptor, a small set of reference points is chosen in the image (see Figure 2 left). These are typically located on a regular grid which is fixed for all templates and scanning windows. A signature of each reference point is then calculated. It is given by the quantized distance to the nearest edge and its quantized orientation. A vector of these signatures forms the descriptor.

![Figure 2. Left: The sparse edge-based descriptor defined on a regular grid. The reference points (stars) with the same signature in both the template edge map (dots) and the test edge map (white pixels) are in red. Right: Oriented chamfer matching. The template edges for which a matching test edges were found are in red.](image)

The second stage is based on the so-called oriented chamfer matching (Figure 2 right), which counts the number of template edges that are near enough to an equally
oriented edge in the test window. The similarity of a template and a scanning window is given by the number of edges that passed this test. In both stages, the nearest edges are found efficiently via distance transform.

2.3. Decision Cascade Extended by ICP Procedure

To improve detection accuracy (mainly to decrease the number of false positives), we have extended the decision cascade by another stage. This stage computes the matching score between a template and a scanning window by first transforming the template by a similarity transformation (translation, rotation and scale) that maximizes the score. Since finding such a globally optimal transformation is intractable, we find a local optimum by the iterated closest point (ICP) algorithm [3]. This algorithm iterates over three steps: (1) assign each template edge to “the nearest” test edge, (2) prune edge pairs being too far from each other, and, using this assignment, (3) compute the similarity transformation minimizing the sum of squared differences of positions of the corresponding edges.

When the standard distance transform is employed for finding the nearest edge in the first step of ICP, one template edge may be assigned to multiple test edges. This is undesirable because it tends to slide the edge map to a local optimum. To avoid this, we designed a fast algorithm to impose uniqueness in edge matching. It iteratively matches the best edge pair candidates while allowing each edge point to be in at most one pair.

Figure 3: An example alignment of a template edge map (yellow-red) to a test edge map (blue-green) using ICP. The color range of each edge map encodes orientation of edge points.

Figure 4: The extended decision cascade. The typical number of test windows passing the stages are presented together with their visualization.
Typically, 5-15 iterations are needed for ICP to converge. To speed up the algorithm, only 100 of randomly selected edge points are used for each iteration. Even then, the ICP algorithm is notably more time consuming than the oriented chamfer matching. But this is precisely the idea of a decision cascade – later stages are slower but more accurate. Due to the ICP alignment, the matching score turns out to be significantly more precise (see Section 5.1.1). Figure 4 shows the new cascade, along with typical numbers of scanning windows passing each stage.

2.4. Adding Depth Information

Besides the purely edge-based approach to detection of texture-less objects, we modified the detection cascade to utilize also depth information. Depth is a useful cue regarding texture-less surfaces, which are difficult to describe by a 2D image only. The benefit of depth was confirmed experimentally (see Section 5.1.1).

The decision cascade was modified as follows. The hypothesis generation is again based on hashing measurements from a regular grid, but depth differences between reference points and their surface normals are considered in this case (see Section 2.4.1 for details). The generated hypotheses are then compared to the test window by matching points of interest in different modalities (intensity gradients, color, depth and surface normals). For this purpose, we use a template matching approach similar to Hinterstoisser et al. [7] (see Section 2.4.2 for details) which replaces the oriented chamfer matching used in the original edge-based detector. An ICP on point clouds could be done in the next step as a replacement of the edge-based ICP, however, we delegate the 3D pose refinement to the separate, dedicated method described in Section 3. The non-maxima suppression is still performed as the last step.

2.4.1. Hypothesis Generation

To retrieve a set of most similar templates for the given test window, we use a modification of the efficient voting procedure presented by Cai et al. [1]. To each test window we attach a set of m reference points located on a fixed regular grid. We then use n voting k-tuples of reference points (typically k = 3) and describe each tuple by a set of measurements. For each such tuple, there is a hash table whose bins correspond to possible values of the quantized measurements. The hash tables are built and filled by identifiers of templates during the training stage. To increase robustness to background clutter, we insert the template identifier into the hash table only if all reference points from the given tuple fall into an object mask. This mask is binary, distinguishing the foreground and background, and is assumed to be available for each template. Note that the templates have the same fixed size as the test window. To generate hypotheses for the given test window, quantized measurements of each tuple are hashed and used to retrieve identifiers of templates with the same quantized measurements. Identifiers retrieved by all the tuples are accumulated and templates with at least p votes are selected for the following verification (typically p = 3).

Descriptor of a Voting Tuple. The purely edge-based method by Cai et al. [1] described each k-tuple of reference points by the distance of the points to their closest intensity edges and the orientation of these edges. In contrast, one k-tuple of reference points is now described by a set of measurements \( d_{1,2}, \ldots, d_{1,k}, s_1, \ldots, s_k \), where \( d_{1,i} \) is a quantized depth difference between the first and the i-th point from the tuple, and \( s_i \) is a scalar representation of quantized 3D surface normal orientation at the i-th point from the tuple. Borders of quantization bins for the depth difference are learned during the training stage. This is done by collecting depth differences from all reference point pairs from the training templates and setting the bin borders such as
each bin contains the same fraction of entries. To calculate the quantized surface normals, we use the approach proposed by [7]. In our experiments, the number of quantization bins for the depth difference and the surface normal orientation is typically set to 5 and 8 respectively. For example, when \( k = 3 \), the hash table has \( 5^2 \times 8^3 = 12800 \) bins.

**Generation of Voting Tuples.** In order to achieve stable detection time, we want the corresponding hash tables to be filled as uniformly as possible and thus each tuple to generate a comparable number of hypotheses for every test window. Hence, to generate the \( n \) voting \( k \)-tuples, we at first randomly generate \( N \) of them (typically \( N = 5000 \) and \( n = 100 \)). From the \( N \) tuples, we keep \( n \) tuples which have the largest entropy of bin occupancy in the corresponding hash tables. This generation of voting tuples is done offline.

2.4.2. **Multimodal Hypothesis Verification**

Similarly to Hinterstoisser et al. [8], we at first select points of interest for different modalities in the offline training stage. These points are selected independently for each template at locations which are salient for the given modality.

For matching, each template point of interest is compared to pixels from the corresponding local neighborhood in the test window. A detection hypothesis passes the verification for a given modality if there is the same quantized value in the neighborhood for at least \( p\% \) of the points of interest (in our experiments \( p = 70 \) and the size of the checked local neighborhood is \( 5 \times 5 \) pixels).

For the hypothesis verification, we use four modalities: image gradients (\( M_1 \)), 3D surface normals (\( M_2 \)), color (\( M_3 \)) and depth (\( M_4 \)). The verification is done in a cascaded style: hypothesis is verified in modality \( M_i \) only if it passed the verification for all \( M_j \), where \( j < i \). Hypotheses which passed the verification for all modalities proceed to the non-maxima suppression stage.

2.5. **Parallel Implementation**

To speed up the detection, its implementation was parallelized on a multi-core CPU using the OpenMP\(^1\) library. Specifically, scanning windows are now processed in parallel. Each thread processes windows uniformly distributed across the image which guarantees the work load of all threads to be balanced. There is no time-consuming data synchronization between the separate threads. The edge maps and distance transforms on all input image scales are pre-calculated and shared among the threads. The detection candidates from all threads are merged together and the non-maxima suppression is performed by the master thread.

The entire source code was profiled, optimized and internal data structures were re-designed to make them more efficient. Moreover, the exact distance transform (used for the sparse descriptor and the oriented chamfer matching in the edge-based detector) was replaced by a faster approximate version from OpenCV.

Without parallelization, the detection time was 8–10s per VGA frame. With the parallelization, this dropped to 0.59s per VGA frame on average with a 16 core CPU (see Section 5.1.1 for more details).

\(^1\) http://openmp.org
3. Fine 3D Pose Estimation

The method for fine 3D pose estimation is described and evaluated in detail in D3.5 but we briefly recall it here for completeness. The 3D pose of the detected object is estimated by fitting its 3D model (example of such models are shown in Figure 5), which is stored in advance in a database, to the input depth map obtained from the RGB-D sensor. The identity of the 3D object is provided by object detections. This fitting is a hard optimization task, which can be solved only sub-optimally. To reach a good local optimum, an approximate initial pose must be known. This initial pose is provided as a byproduct of object detection.

![Figure 5: 3D models of the fuse and fuse box.](image)

The optimization method uses a hypothesize-and-test approach which forms 3D pose hypotheses and evaluates them with a score, eventually returning the best pose found according to this score. To explore the pose space efficiently, numerical optimization is employed using the Particle Swarm Optimization (PSO) [4]. In each iteration, PSO evaluates multiple hypotheses, each one corresponding to a ‘particle’ in a neighborhood of the pose-space surrounding the initial pose. To evaluate a hypothesis, a particle renders an image of the model at the hypothesized pose. The evaluation score is determined by the degree of similarity of the rendered image to the real sensor data. The rendering of pose hypotheses is accelerated by implementing it in parallel on GPU.

4. Integrated Vision Module

In this section, we describe the integration of the object detection and pose estimation methods into a combined vision module. Section 4.1 introduces the architecture of the module and its implementation. Section 4.2 then describes the acquisition of real and synthetic training data and provides their comparison.

4.1. Architecture of the Vision Module

The architecture of the integrated vision module is shown in Figure 6. The input is a stream of RGB-D frames from a Kinect-like sensor. Depending on its version, the object detector can either receive only a grayscale image (in the case of the purely edge-based detector) or a full RGB-D frame (in the case of the new detector described in Section 2.4). The object detection submodule scans the input image and for each detected template provides its image position, scale and information about the rough 3D pose of the detected object. The rough 3D pose, which is associated with each template during training (see Section 4.2), is then adjusted according to image position and scale of the detected template and eventually refined by the pose estimation submodule.
The 3D object poses which are associated with the training templates cannot be directly used as initial poses for the fine 3D pose estimation. The reason for this is that they correspond to the object aligned at the origin of the reference coordinate system (where it was during acquisition of training data) and needs to be adjusted to reflect the 2D image position and scale of the detected templates. In particular, an extra rotation is applied to the original pose to account for the detection displacement from the image center. The median 3D point distance is calculated from pixels belonging to the object mask in the detected template and the object distance from the camera center is then given by this median distance + median offset of the frontal object surface in the detected pose (the offset is calculated for each template during training and is relative to the plane parallel to the image plane and passing the origin of the reference coordinate system).

As an optional input, the vision module is able to accept prior information about occurrence of objects in the form of a list of expected objects or in the form of a binary mask (the dashed arrow in the diagram in Figure 6). This mask can be also used for verification of object presence in a specified region.

Training data for each object consists of a set of templates, which are used for object detection, and a 3D mesh model, which is used for 3D pose estimation. A template consists of a 2D image (RGB or color – depending on the version of the object detector), a binary object mask, and a depth map (in the case of the depth-based object detector). The 3D models can be shared with other DARWIN modules, such as the grasping module. To facilitate grasping, the 3D models are endowed with grasping affordances by fitting a cylinder around each graspable part.

The architecture of the vision module is similar to that proposed by Muja et al. [5], which also includes separate modules for object detection and pose estimation. Besides, their architecture includes a special attention module which actually corresponds to the pre-filtering stage in our detection cascade. Another similar approach was presented by Holzer [6] or Hinterstoisser et al. [7].

The vision module is fully implemented in C++ and communicates with the other DARWIN modules via the YARP robotic platform. For better efficiency, the two submodules of the vision module communicate with each other via C++ interfaces.

---

2 http://eris.liralab.it/yarp
rather than YARP messages. The module produces a constant stream of YARP messages encoding the identities of recognized objects and their estimated 3D poses. Every message is equipped with a time stamp of the detection used for synchronization within the system.

4.2. Acquisition of Training Data

The integrated vision system requires the training templates to be annotated with information about object pose. For this reason, acquisition of real training data (using an RGB-D sensor) requires a special setup, unlike in the case of raw 2D detection approaches (as described in Deliverable 3.4). The following two subsections provide details about acquisition of both real and synthetic data and compares their use for training.

Figure 7: Our acquisition setup (left) and one of the captured images with visualized 3D coordinates estimated from the calibration markers (right).

4.2.1. Real Training Data

To capture real training data, we use the setup shown in Figure 7. The object is placed on a board (with a black central part and a calibration marker field around) which rests on an uncalibrated turntable. The sensor can observe the object from multiple different elevations. There is a black vertical shield behind the object which does not move with the table and thus ensures that the object is seen by the sensor always against black background. Figure 8 shows the used calibration marker field and samples of captured templates of a fuse box.

Figure 8: The calibration marker field (left). Sample templates of a fuse box (right).
As the internal camera parameters are known (from the specs of the RGB-D sensor), it is possible to compute the sensor's pose from the special marker field visible in each RGB image (Figure 8 left). This computation employs the posest\(^3\) library for robust 6DoF pose estimation from 3D-2D correspondences which was developed earlier in the course of DARWIN.

### 4.2.2. Synthetic Training Data

As an alternative source of training data, we also investigated the use of synthetic templates which were rendered from CAD models. Examples of a real and a synthetic RGB-D template are shown in Figure 9.

![Real and Synthetic RGB-D Templates](image)

**Figure 9**: A real RGB-D template captured by a sensor (left) and a synthetic RGB-D template rendered from a CAD model (right). The reddish is the color in the depth map, the bigger depth of the object surface it represents, while the blue color encodes an undefined depth information.

The obvious advantage of synthetic templates is the simplicity of their acquisition. No special capturing setup with a turntable and calibration markers is needed since the object pose is known when rendering. Another advantage concerns the noise-free depth maps. Possible disadvantages could be artificial lighting (leading to unrealistic intensity gradients) or other artefacts caused by imperfections of 3D models which lack some of the object details. However, none of these was observed to be an issue in our experiments.

To compare detection performance for both types of training data, we ran the detector on the test dataset presented in Section 5.1.1. We ran it once with real templates and once with synthetic ones. The new detector utilizing depth information (Section 2.4), was used for this evaluation.

![Detection Plots](image)

**Figure 10**: “Detection rate / false positives per image” for the weaker (left) and the stricter (right) detection criterion.

When defining an object detection to be a true positive if it has at least 50% overlap with the ground truth, the performance characteristic was very similar for both types of templates, achieving a very high detection rate for already very low number of

\(^3\) [http://users.ics.forth.gr/~lourakis/posest](http://users.ics.forth.gr/~lourakis/posest)
false positives. When a stricter criterion (requiring the overlap of at least 70%) was applied, the use of the synthetic templates was even favourable, when accepting a bit higher number of false positives per image. This suggests that the use of the synthetic templates may lead to a better 2D localization and thus confirms synthetic templates to be a viable alternative to real templates. The evaluation graphs are shown in Figure 10.

5. Experimental Evaluation

This section reports the achieved performance, in both accuracy and speed, of the integrated vision module on two datasets: (1) the CVUT industrial dataset focused mainly on object detection (Section 5.1.1) and (2) the public dataset by Hinterstoisser et al. [7] which is suitable for evaluation of the whole integrated vision module since it provides ground truth for 3D object poses (Section 5.1.2).

5.1.1. Evaluation on CVUT Industrial Dataset

This dataset, designed for evaluation of object detection, consists of 150 RGB-D frames with annotated ground truth in the form of 2D bounding boxes. The dataset contains frames which typically occur in DARWIN user stories, each containing three fuses and one fuse box, and which present varying illumination conditions (see Figure 12).

On this dataset, we evaluated three versions of our object detector:

- DET-ORIG – The original edge-based detector presented in Section 2.2.
- DET-ICP – The original edge-based detector extended by the ICP procedure described in Section 2.3.
- DET-DEPTH – The detector using depth information presented in Section 2.4.

As can be seen in Figure 11, DET-ICP clearly outperforms DET-ORIG. This confirms the benefit of the additional ICP procedure. DET-ICP is further outperformed by DET-DEPTH which proves the added value of depth information for detection of texture-less objects.

![Figure 11](image-url) "Detection rate vs. false positives per image" curves for the three evaluated versions of our object detector. A detection was considered as a true positive if its overlap (intersection / union of 2D bounding boxes) with the ground truth was at least 50%.

Since the ground truth of 3D poses is not available for this dataset, evaluation of the refined 3D poses was done qualitatively, i.e., a 3D pose was considered correct if the projection of the object model visually agreed with the real image (the visual agreement was decided by a human). As a result, 3D poses were correctly estimated for 92% of objects in the dataset.
Average detection and pose estimation time in one VGA frame was 0.59s and 0.4s respectively. This makes the total processing time below 1s per frame (on a single machine with 16 core CPU and NVidia GTX 580 GPU). Let us emphasize that this speed was obtained with the occurrence mask given by the whole robot workspace. Within the DARWIN architecture, the speed is further improved by using prediction and tracking which provide more informative occurrence masks.

![Sample detections on the CVUT’s dataset (top row) and corresponding renderings of object models in the refined 3D poses (bottom row).](image)

Sample detection results of DET-ICP together with the object models superimposed semi-transparently at the corresponding refined 3D poses can be seen in Figure 12. All results of DET-ICP used for the evaluation can be found online: http://goo.gl/rEcYZG.

### 5.1.2. Evaluation on Hinterstoisser’s Dataset

The integrated vision module was evaluated on objects from the publicly available dataset of Hinterstoisser et al. [7] (Figure 13). The training templates were rendered from the available 3D models so that they uniformly covered the upper view hemisphere (with the step of 10° in both azimuth and elevation). To achieve invariance to rotation around the optical axis, we also applied an in-plane rotation to each template (with the step of 10°). There are typically fewer than 3000 templates for an object. For each object, the dataset contains one test sequence of about 1200 RGB-D frames captured from different viewpoints. Each of these frames includes one instance of the object of interest with known ground truth of its 3D pose and contains heavy close range and far range 2D and 3D clutter.

![Objects from Hinterstoisser’s dataset which was used for evaluation (namely: ape, bench vise, can, cat, driller, duck, glue).](image)

We used the same quantification of pose error as in [7]: given the ground truth rotation $R$ and translation $t$ and the estimated rotation $R'$ and translation $t'$, the pose error is the average distance of all points $x$ (from model $M$) from their transformed versions:

$$\text{err} = \text{avg}_{x \in M} \| (Rx + t) - (R'x + t') \|.$$
A model is considered to be correctly detected and its pose correctly estimated if \( k_{err}d \geq err \) where \( k_{err} \) is a chosen coefficient and \( d \) is the diameter of \( M \). As in [7], we used the following modification of the score for the “glue” object whose pose is ambiguous (it is difficult to distinguish between flips around vertical axis due to symmetry):

\[
err = \text{avg}_{x_1 \in M} \min_{x_2 \in M} \| (Rx_1 + t) - (R'x_2 + t') \|.
\]

We compare our method with several state-of-the-art methods: two methods of Hinterstoisser et al. (LINEMOD [8] and LINEMOD++ [7]) and one method of Drost et al. [9]. These methods were already briefly described in Section 2.1, here we provide more details about their relation to our method. LINEMOD performs an exhaustive template matching approach (this matching is adopted in our DET-DEPTH detector as the hypothesis verification stage). LINEMOD++ extends this by several post-processing verification steps. Specifically, a color check and a depth check by a rough but fast 3D ICP is done in order to prune hypotheses and a finer 3D ICP is then performed for the best of the remaining hypotheses. In our method, DET-DEPTH was used for object detection. The essential difference of our method with respect to LINEMOD++ is the added hypothesis generation stage which efficiently generates a subset of template candidates. Only these are then evaluated by the more computationally demanding template matching, thus avoiding the exhaustive approach. Other difference can be found in the template matching for which LINEMOD++ uses just two modalities - image gradients and surface normals. We employ also color and depth for this matching and treat all the modalities equally. LINEMOD++ uses color and depth later in the post-processing stage where it treats each modality differently (the color check is done over all object pixels and depth is used for the 3D ICP).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ape (1235)</td>
<td>92.4</td>
<td>95.8</td>
<td>69.4</td>
<td>86.5</td>
</tr>
<tr>
<td>bench vise (1214)</td>
<td>94.7</td>
<td>98.7</td>
<td>94.0</td>
<td>70.7</td>
</tr>
<tr>
<td>can (1195)</td>
<td>94.5</td>
<td>95.4</td>
<td>79.5</td>
<td>80.2</td>
</tr>
<tr>
<td>cat (1178)</td>
<td>94.1</td>
<td>99.3</td>
<td>88.2</td>
<td>85.4</td>
</tr>
<tr>
<td>driller (1187)</td>
<td>90.0</td>
<td>93.6</td>
<td>81.3</td>
<td>87.3</td>
</tr>
<tr>
<td>duck (1253)</td>
<td>95.3</td>
<td>95.9</td>
<td>75.9</td>
<td>46.0</td>
</tr>
<tr>
<td>glue (1219)</td>
<td>97.8</td>
<td>91.8</td>
<td>64.3</td>
<td>57.2</td>
</tr>
<tr>
<td>average</td>
<td>94.1</td>
<td>95.8</td>
<td>78.9</td>
<td>73.3</td>
</tr>
</tbody>
</table>

Table 1: Recognition rates [%] for \( k_{err} = 0.1 \) (i.e. percentage of frames for which \( 0.1d \geq err \)).

As in the other methods used for comparison, 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 3D pose estimation approach and an error of the resulting pose was calculated as explained above. Table 1 compares recognition rates (for \( k_{err} = 0.1 \)) of our method with several state-of-the-art methods whose rates were adopted from [7]. For all objects, our method achieved a recognition rate \( \geq 90\% \) (94.1\% on average) and outperformed LINEMOD and the method of Drost. LINEMOD++ achieved better recognition rates than our method for the first six objects (by 3\% on average) while we obtained better results for the “glue” object (by 6\%). We attribute the subtly superior results of LINEMOD++ mainly to its fine post processing steps. Recognition rates with respect to different \( k_{err} \) can be seen in Figure 14 (left) and visualization of sample results in Figure 15.

For the robotic manipulation tasks considered in the DARWIN project, the industrial robotic arms used by Profactor are able to tolerate inaccuracies up to 1cm in object
localization with the help of compliant grippers. As can be seen in Figure 14 (right),
the pose error achieved by our method meets this requirement for all the
Hinterstoisser’s objects used in the evaluation. We suppose that the error is similar
also for the fuse and the fuse box which are the objects used in the DARWIN user
stories and for which we get qualitatively comparable results.

![Graph showing recognition rates with respect to different k_err.]

Figure 14: Left: Recognition rates with respect to different k_err.
Right: Mean pose errors of correctly recognized objects for k_err = 0.1. For all objects, the
mean errors of the refined poses are all at least one standard deviation (visualized with cyan)
below 1cm.

In terms of running time, our method needed on average less than 1.5s per VGA
frame. It is slower than the time reported in Section 5.1.1 because there was no
occurrence mask available in this case and thus the whole frame had to be scanned
in the object detection stage. As reported in [7], the method of Drost needed on
average 6.3s and LINEMOD++ only 0.12s. The latter time was achieved with highly
optimized implementation using heavy SSE parallelization. With this level of
optimization, we expect our method (at least the object detection part) to run even
faster thanks to the added hypothesis generation stage which allows to compare only
a fraction of training templates against each test window. The hypothesis generation
stage makes our method to scale well to bigger amounts of training templates.

![Visualization of detected objects in estimated 3D poses.]

Figure 15: Visualization of detected Hinterstoisser’s objects in the estimated 3D poses.

Multiple object detection is a common problem in robotic applications where our
method is expected to clearly outperform the speed of LINEMOD++. Such evaluation
is the subject of future work.
6. Conclusions

We have described an approach for fast detection and pose estimation of texture-less objects integrating appearance-based and shape-based object representation. This approach forms the core of DARWIN vision module, which is the main outcome of WP3.

The progress in object detection since Deliverable 3.4 has been presented, most importantly the modification of the detection cascade to utilize depth information. The new detector has been shown to outperform the purely edge-based object detectors, which confirmed the benefit of depth for detection of texture-less objects.

We have described the architecture of the integrated vision module, the process of acquisition of training data for object detection, and validated the use of synthetic training data rendered from CAD models as a viable alternative to real training data. The simplicity of the synthetic training data acquisition increases industrial relevance of the presented vision module.

The experimental evaluation on the public dataset of Hinterstoisser has shown the proposed method comparable with the state-of-the-art methods, achieving an average recognition rate of 94.1%.

The mean error of the estimated 3D poses was for all the Hinterstoisser’s objects at least one standard deviation below 1cm. Assuming that a similar error is achieved for the objects from the DARWIN user stories, this meets the accuracy required by the used industrial robotic arms with compliant grippers.

The average total time of detection and pose estimation was on the industrial dataset, which consists of isolated frames, below 1s per frame. Within the DARWIN architecture, the response time of the vision module is reduced by tracking over continuous stream of frames.

Currently, we are in the final stage of the preparation of a joint CVUT-FORTH publication describing the integrated vision module. It will be submitted to one of the coming major robotic conferences (RSS 2015 or IROS 2015). Once the publication is finished (by December 2014 the latest), it will be available online through this link: http://cmp.felk.cvut.cz/~hodanto2/darwin/hodan2015detection.pdf
Bibliography


