6D pose estimation of textureless shiny objects using random ferns for bin-picking

José Jeronimo Rodrigues, Jun-Sik Kim, Makoto Furukawa, João Xavier, Pedro Aguiar, Takeo Kanade

***Abstract*— We address the problem of 6D pose estimation of a textureless and shiny object from single-view 2D images, for a bin-picking task. For a textureless object like a mechanical part, conventional visual feature matching usually fails due to the absence of rich texture features. Hierarchical template matching assumes that few templates can cover all object appearances. However, the appearance of a shiny object largely depends on its pose and illumination. Furthermore, in a bin-picking task, we must cope with partial occlusions, shadows, and inter-reflections.**

In this paper, we propose a purely data-driven method to tackle the pose estimation problem. Motivated by photometric stereo, we build an imaging system with multiple lights where each image channel is obtained under different lightning conditions. In an offline stage, we capture images of an object in several poses. Then, we train random ferns to map the appearance of small image patches into votes on the pose space. At runtime, each patch of the input image votes on possible pose hypotheses. We further show how to increase the accuracy of the object poses from our discretized pose hypotheses.

Our experiments show that the proposed method can detect and estimate poses of textureless and shiny objects accurately and robustly within half a second.

I. INTRODUCTION

Detecting and localizing objects in three-dimensional space is essential for robotic manipulation. One practical task is known as "bin-picking", where a robot manipulator picks objects from a bin of parts without any assistance of an operator. For such a task, vision-based object detection and localization can be a cost-effective solution.

In practice, however, vision-based methods encounter some technical challenges. Industrial parts are usually made of metal and their surfaces are highly reflective. Due to this reflection property, the object appearance becomes highly dependent on the distribution of light, surface material, camera viewing direction and pose of the object. Fig. 1 shows the drastic appearance changes of an object in a bin-of-parts image. Because of the strong specular reflection in a narrow range of viewing directions, the dynamic range of the irradiance is large, and the intensity greatly changes with a small pose change. In addition, the inter-reflection from the nearby objects or even the object itself can not be neglected.

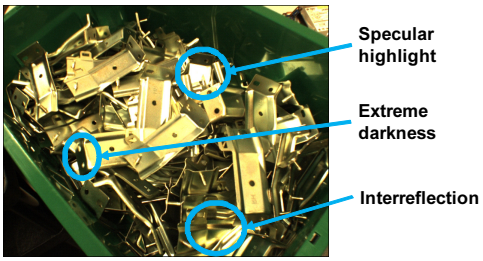


Fig. 1. Image of shiny objects with their typical appearance variations

A. Related work

Sparse 2D-3D feature matching. A very popular approach to the pose estimation problem is to establish visual feature matches between an input 2D image and a 3D object model [1], [2], [3]. However, the drastic appearance change of a shiny object makes it intractable to find the corresponding visual features between images of the same object in distinct poses. Additionally, visual feature matching only works well for objects containing locally planar textures [3], [4], [5], [6], which industrial parts rarely have.

Template matching. Another conventional way to estimate an object pose is to match whole templates to an input image [7], [8]. In this approach, object images in many poses are captured in a database. The object pose can be estimated by correlating image templates in the database with the input image, and finding the best match. Though this approach is simple, it has two major drawbacks: long computation time and sensitivity to the local appearance change by occlusion, inter-reflection, shadows or small pose changes.

Recent improvements on template matching focus on solving these two issues. The computation time can be significantly reduced by using image pyramids and adopting divide-and-conquer strategy methods on the structured image database [7]. The local appearance changes can be tackled by developing more robust appearance descriptors [10], by representing an image as a grid of patches [8], or by using systems that produce more stable or rich features [18], [9]. While this approach is suitable for pose estimation of textureless objects, the usage of image pyramids to speed up these methods assumes that few templates can cover all object appearances. However, shiny objects are highly view-dependent. Thus, many templates are necessary to represent

This work is supported by Honda Engineering Co., Ltd. and by Portuguese Foundation for Science and Technology (FCT), under ISR/IST plurianual funding, through the PIDDAC Program funds, and Grants CMU-PT/SIA/0026/2009 and SFRH/BD/33521/2008, through the Carnegie Mellon|Portugal Program managed by ICTI.

J.-S. Kim and T. Kanade are with the Robotics Institute, Carnegie Mellon University. {kimjs, tk}@cs.cmu.edu

J. J. Rodrigues is with the Robotics Institute, Carnegie Mellon University, and with the Institute for Systems and Robotics, Instituto Superior Técnico, Technical University of Lisbon, Lisbon, Portugal. jmoreira@cs.cmu.edu

M. Furukawa is with Honda Engineering Co., Ltd., Japan.

J. Xavier and P. Aguiar are with the Institute for Systems and Robotics, Instituto Superior Técnico, Technical University of Lisbon, Lisbon, Portugal. {jxavier,aguiar}@isr.ist.utl.pt

all the variations of a shiny object and guarantee detection across all poses.

Vote-based pose estimation. Several methods attempt to use Hough transform for detecting objects in higher dimensional pose space [11], [12]. As in a conventional generalized Hough transform, each edge point votes for all possible poses. This is inefficient because the number of possible poses for each edge point is too large. In addition, it depends on the binarization, which is not so stable for a high-dynamic range image of a shiny object. Recently, similar voting approaches based on image patches, rather than on edge images, have been proposed using random forests for detecting and classifying objects [13], [14] and for matching interest points [4], [5] in 2D images. We borrow some of these ideas to our framework in 6D pose estimation.

B. Proposed approach and contributions

In this paper, we propose a practical pose estimation system designed for the bin-picking of a textureless and shiny object. Motivated by photometric stereo, we build a multi-light imaging system where each image channel is obtained under different lightning conditions. To solve the problem of the appearance changes without resorting to the complicated modeling of the imaging process, we use a fully data-driven approach. Images of an object in many different poses are captured in advance and, at runtime, the algorithm generates pose hypotheses by selecting images in the database. Instead of matching the database images directly to the input image, we propose a voting approach to generate *bottom-up* hypotheses from image patches. Our multi-light imaging system provides object images with rich clues about the object pose, enabling the generation of reliable pose hypotheses from an image patch.

The key contributions of this work are:

- We build a multi-light imaging system where the image color changes with surface normal, enabling efficient pose estimation from patches.
- We develop a data-driven method for 6D pose estimation, using random ferns to map the patch appearance into pose hypotheses votes.
- Our 6D pose estimation system handles various textureless shiny objects without a need for object-specific tuning of system parameters.
- We make an intense evaluation on sets of 100 sequential picking tests involving the realistic effects of occlusions, shadows, and inter-reflections.

II. Imaging system using multiple lights

To have enough discrimination between images of objects in different poses, we created an imaging system using multiple lights motivated by the photometric stereo method [15]. In this section, we briefly review the photometric stereo method and discuss about its limitations in our case.

A. Photometric stereo and its limitations

Photometric stereo [15] is a method to reconstruct a surface from its orientation estimated from its responses to multiple lights. Assuming that the surface is Lambertian, and there is no ambient light, the intensity I of a point is related to its surface normal $\mathbf{n}$ and the light direction $\mathbf{L}$ as $I = \rho \mathbf{n}^\top \mathbf{L}$, where ρ is an albedo of the point. For multiple light sources at known positions, the equation can be stacked

$$\begin{bmatrix} I_1 & I_2 & I_3 \end{bmatrix} = \rho \mathbf{n}^\top \begin{bmatrix} \mathbf{L}_1 & \mathbf{L}_2 & \mathbf{L}_3 \end{bmatrix}, \tag{1}$$

and the surface normal $\mathbf{n}$ and its albedo ρ can be estimated by solving the linear system. Once the surface normal for every single point is estimated, the surface shape can be reconstructed by integration.

The above algorithm is based on some assumptions which are not appropriate for our case. First, the surface of shiny objects is not Lambertian, and the imaging process can not be described as simply as Eq. (1). To model the imaging process for a shiny surface, an accurate bidirectional reflectance distribution function (BRDF) is required, which is not trivial to obtain. In addition, the above algorithm assumes that all the light directions are known and, more importantly, each intensity value is affected by only one light ray. In practice, these assumptions are useful only when using point light sources or parallel lights with accurate system calibration, which are hard to achieve in a factory site.

Rather than trying to estimate the surface orientation by modeling the imaging process accurately, we just collect the images by switching lights one by one. We assume that neither the surface reflectance property like a BRDF nor the light distribution of multiple lights is known. Objects can be complex, being composed by different materials or having different finishings along the surface. The collection of images implicitly encodes the surface orientation, and discriminates between poses of the object. Our data-driven classifier utilizes this discrimination without complex imaging models and their calibration process.

B. System implementation and multi-light image

Fig. 2 shows the implemented imaging system. It has three incandescent light bulbs, which are easily available from a retail store. The lights are about 1 meter high from the object and roughly located at vertices of a regular triangle. A B/W camera is about 1.75 meters high from the object and aims to the center of the bin. To minimize the effects of the ambient light from fluorescent lights on a ceiling, a 720nm IR filter is attached in front of the lens.

Because our system has three light sources, we collect the images in three channels of a RGB image as shown in Fig. 3. We call this a *multi-light image*, and each color *encodes* the orientation of the surface. Because the light sources are not far enough to be parallel, the color depends not only on the surface orientation but also on the location of the surface, being not possible to have a one-to-one mapping between color and surface orientation.

C. Database collection

The proposed pose estimation method is data-driven. Therefore, we need to collect images of an object in many different poses, to make a database. In order to automate the data collection, we built the rotation stage with three

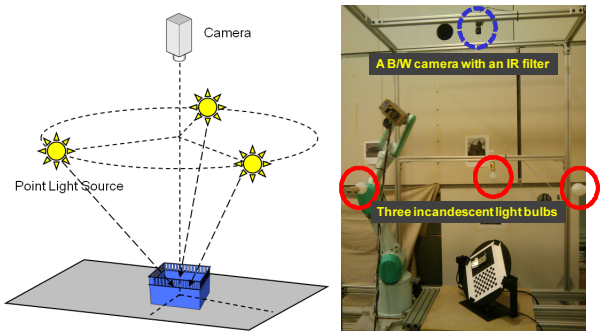


Fig. 2. Three light source system. Left: Conceptual diagram. Right: Real implementation of the multi-light source imaging system with a rotation stage for database collection.

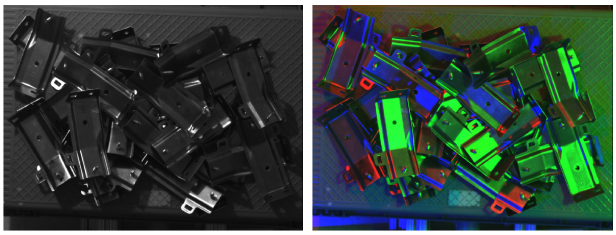


Fig. 3. Multi-light image. Left: a gray image under the natural illumination. Right: a multi-light image captured by the three light sources in Fig. 2. (Best viewed in color)

rotation axes, shown in Fig. 2. We estimate the object pose with respect to the camera using the checkerboard on the rotation stage.

The number of images in the database determines the resolution of the pose hypothesis. We uniformly sample the 3D rotation angles. If the object is two-sided as shown in Fig. 2, both sides should be collected independently. Typically, we capture about 16600 images of an object for both sides, and in this case, the orientation resolution is approximately 4-degree.

III. Data-driven Pose Estimation

Three-dimensional pose estimation includes detecting an object and estimating its pose in the 3D space. We propose a data-driven method based on random ferns to solve both tasks simultaneously. Utilizing the distinctiveness of images in various object poses, pose hypotheses can be generated effectively. Each patch generates *bottom-up* proposals of object poses, and all the information from image patches are aggregated to create valid pose hypotheses.

A. Pose Estimation by Patch Voting

Given an input image I, our problem is to find a 6D object pose $p = (t_x, t_y, t_z, \rho, \theta, \phi)$ with position (t_x, t_y, t_z) and rotation angles (ρ, θ, ϕ), which can be formulated as

$$\arg\max_p P(p|I). \tag{2}$$

Since in our context it is hard to write a realistic expression for $P(p|I)$, we propose a series of approximations.

We consider a discrete version of the problem by performing the search on a set of interest poses $p_i = (t_{xi}, t_{yi}, t_{zi}, \rho_i, \theta_i, \phi_i)$, $i = 1, 2, \ldots, n$:

$$\arg\max_i P(p_i|I). \tag{3}$$

Defining the object position in the image plane as $(x_i, y_i) = (t_{xi}/t_{zi}, t_{yi}/t_{zi})$ and the remaining pose information as $c_i = (t_{zi}, \rho_i, \theta_i, \phi_i)$, problem (3) becomes

$$\arg\max_i P(x_i, y_i, c_i|I). \tag{4}$$

Inspired on a recent work in 2D object detection [13], our subsequent idea is to use many small image patches to vote for object poses. The votes for a pose (x_i, y_i, c_i), from a set of small image patches $M_j(x_j, y_j)$, centered at (x_j, y_j), are accumulated in a non-probabilistic way as

$$A(x_i, y_i, c_i) = \sum_j P(x_i, y_i, c_i|M_j(x_j, y_j)), \tag{5}$$

In order to learn the patch votes $P(x_i, y_i, c_i|M_j(x_j, y_j))$ at the training stage, one would need to collect images of the object across all possible poses (x_i, y_i, c_i). However, under the assumption of parallel light sources and orthographic camera, the appearance of the object, as well as a patch centered at a fixed point on the object, remains the same while varying (x_i, y_i), with c_i fixed. Under this assumption, the patch votes in our system depend on the relative position between the object and the patch, rather than their absolute positions. As a result, the patch votes can be rewritten as $P(x_c, y_c, c_i|M_j)$, where $(x_c, y_c) = (x_i - x_j, y_i - y_j)$ is the position of the object in the image, relative to the patch center.

At the training stage, each patch $M_j(x_j, y_j)$ of a database image with object pose (x_i, y_i, c_i) contributes to the probability $P(x_c, y_c, c_i|M_j)$ of having the object at position (x_c, y_c) from its center and at depth and orientation c_i. Then, we use this probability in the online stage to cast votes for the pose of a part given the local appearance M_j of a patch. From now on, we will refer to c_i not as $(t_{zi}, \rho_i, \theta_i, \phi_i)$ but as the corresponding database image index, for convenience.

Two practical issues are 1) obtaining the object position (x_c, y_c) in the database images, and 2) describing the local appearance of a patch M_j.

1) Position of the object in the image: We define the position of the object in each database image as the center of the image region that the object occupies. In practice, each database image has its 6D object pose data (x_i, y_i, c_i), estimated by the checkerboard pattern, as shown in Fig. 2. Given the object pose and CAD model, the imaged region of the object is obtained by projection and its mass center is computed as a 2D centroid.

2) Describing the patch appearance: Describing the appearance of a patch is complex, given its gargantuan number of possible appearances. To tackle this problem, we construct a codebook of appearances using a large set of patches from our image database, and associate each patch appearance M_j with a cluster label. Then, the probability $P(x_c, y_c, c_i|M_j)$ is computed for each patch label L in the codebook and not for all possible patch appearances M_j. In the sequel, we explain how we construct this codebook.

B. Clustering the local appearance using random ferns

In order to use the voting scheme in Eq. (5), we need the voting information of database patches that are similar to the

ones in the input image. One possible approach is to search exhaustively for the input patch in the entire database, and then vote with the pose information of the database patch found. However, the number of patches in the database is huge, being more than 100 million for our experiments. The dimension of $n \times n$ patches is $3n^2$ when using three light sources, which is also very large for our patch size $n = 17$.

Alternatively, we can use fast approximate nearest neighbor (NN) search methods to query large databases. These methods usually use KD-trees, hierarchical k-means trees or ferns, where the querying time grows logarithmically with the database size. Using trees, a basic search for a NN candidate corresponds to traverse the tree, and upon reaching a leaf node, perform exhaustive search on the data points in that leaf. We point out that NN search requires all the data to be in memory, which is intractable for our database size. Additionally, the NN candidate might be the correct match with low probability, due to the image noise, and we might have multiple similar data points with useful information for the voting step.

We construct a discriminative codebook where we store just the data statistics at the leaf nodes, i.e., $P(x_c, y_c, c_i|L)$, not raw data points M_j, avoiding the need of a colossal amount of memory for the high-dimensional appearance data M_j. At query time, we find the leaf node of each patch by traversing the tree, and obtain the statistics for the leaf.

To construct the codebook, we have to choose the tree questions at the training step. Given the large size of our database, we need a tree that is easy to train, consumes low memory, and has short retrieval time. Optimally designing a hierarchical k-means tree or a KD-tree requires solving large-sized optimization problems. In addition, for a tree with $m-$levels, storing 2^m questions consumes a large amount of memory. Instead, we use a *random fern*, which is a binary tree with one question per level. Its questions are designed via an easy random process, described below. Having only one question in each level of the $m-$level tree, independent of the ancestors, a fern becomes easily parallelizable. Moreover, there are only m questions to store in memory.

C. Random fern with simple binary questions

Now, the problem is to design the m questions that constitute a fern. We use simple binary questions

$$q_i(M) = \begin{cases} 1 & \text{if } M(p_{i1}) - M(p_{i2}) < \tau \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

at each level i of the fern, which compares two intensity values in the patch M at p_{i1} and p_{i2}. In our case, the image has multiple channels, so $p_{ij} = (x_{ij}, y_{ij}, c_{ij})$ includes the channel c as well.

The number of possible binary questions for $n \times n$ three-light image patches is $3n^2 \times 3n^2$, which makes the design of the m questions a large optimization problem. We randomly choose two points in each channel. Though the resulting fern is not well balanced, similar patches are clustered successfully.

In the training stage, each leaf of the fern collects the centroid locations and pose indices of the patches which have

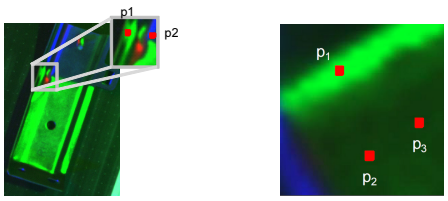


Fig. 4. Binary questions in a fern. For small τ, the binary question defined in (6), between p_1 and p_2 is stable for the given patch, but the one between p_2 and p_3 is highly dependent on the camera noise. (Best viewed in color)

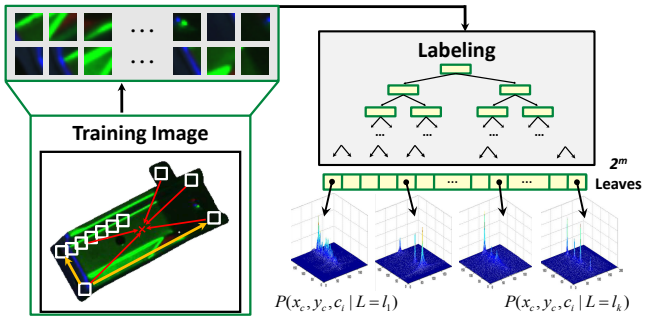


Fig. 5. Training of the probability $P(x_c, y_c, c_i|L)$

the same label. The probability $P(x_c, y_c, c_i|M_j)$ is computed for all the possible labels as $P(x_c, y_c, c_i|L) = 1/q$ where q is the number of database patches M_j assigned with the label L. Fig. 5 shows the training procedure.

A few remarks are in order.

1) Level of the fern: As discussed before, each patch in the database is assigned an $m-$bit binary sequence label. For about 130 million patches in our database, we still use 27 levels of the fern so that we have 2^{27} possible label. Definitely, this over-segments the patch space. In practice, only about 10% out of 2^{27} labels are used, and based on this statistics, we can reduce the number of levels by 4.

However, the oversegmentation of the patch space is useful. The first reason is that the fern is not balanced. Some of the labels are assigned to too many patches, and the probability $P(x_c, y_c, c_i|L)$ becomes too small. By oversegmentation, the number of useful labels becomes larger. On the other hand, some similar patches happen to be scattered into multiple leaves. This is not a problem because similar patches in the input images should be one of the leaves and vote for the correct pose hypothesis.

Some input image patches in the online stage may have wrong labels due to image noise. In this case, by oversegmentation, wrong votes are most likely to be scattered randomly because the wrongly assigned label is randomly selected by the image noise. Though the intentional oversegmentation works well, too much oversegmentation requires a lot of memory. Thus, the available memory should be considered when choosing the depth of the fern.

2) Dealing with multiple channels: The multi-light image captured by our system in Fig. 2 implicitly encodes the surface normal information in the form of color. However, as we discussed, we define each question within a single channel, but not across channels. This is mainly because the

light sources in our system are not far enough, which creates different colors for the same oriented surface at different locations. Therefore, the color can not be mapped into a surface orientation. However, the local change in surface orientation still produces local color change, and thus, the question in the comparison form in Eq. (6) is meaningful in each channel. To use all three channels equally, we use the same number of questions for each channel.

If the light sources are far enough to be assumed parallel, then the color can be mapped directly to the surface orientation. In this case, the questions across channels can be chosen at random.

3) Homogeneity and Threshold τ in questions: The questions in Eq. (6) compare two intensity values. Thus, the $m-$bit binary label of a fully homogeneous image region is selected randomly by the image noise. We tackle this problem in two different ways.

First, we choose only the patches which have large gradient for the voting process, reducing the chance of asking questions in homogeneous regions. This preprocessing rejects a large amount of patches that have low information about pose, and greatly reduces the voting time since repeated homogeneous patches appear frequently in the database of a textureless object.

Second, we should set the threshold τ in Eq. (6). Even in the patches containing large gradients as in Fig. 4, some patch questions are on the homogeneous region. By setting τ properly, the patch clusters become stable. The threshold τ is the noise level of the pixel intensity, which is not a fixed number for every intensity level [16].

We tested two different strategies: using a fixed threshold τ =0, 5, or 10 for all questions, and randomly selecting τ for each question within a range $[0,20]$. In almost all the cases there is no significant performance variation, with the exception of $\tau \approx 0$, where the overall performance degrades a little. This is mainly because the intentional oversegmentation properly handles the wrong labeling by image noise.

D. Online algorithm

The online algorithm is straightforward. For each patch M_j in the input image, we obtain its label L by asking the m questions, and retrieve the list of votes (x_c, y_c, c_i) in the database. By accumulating all the votes from all the patches, the best pose hypotheses are generated. As in the training stage, homogeneous patches do not participate in this online voting process. We discuss now the strategies for speed and robustness of our algotithm.

1) Speeding-up: pose marginalization: Though the online algorithm is simple, it requires a huge 3D accumulator of (x_i, y_i, c_i). For example, if the image size is 1024×768 and the number of database images, i.e. possible poses, is 16600, the number of the accumulator bins is more than 12 billion. Searching for best hypotheses in this huge three-dimensional accumulator takes a long time.

To accelerate the search, we propose a two-step search method by marginalization of pose indices c_i. In the initial voting stage, a two-dimensional voting accumulator $A(x_i, y_i)$ for object centroids (x_i, y_i) is considered by evaluating

$$\begin{aligned} A(x_i, y_i) &= \sum_i P(x_i, y_i, c_i) \\ &= \sum_i \sum_{x_c, y_c} P(x_c, y_c, c_i | L(M(x_i - x_c, y_i - y_c))) \end{aligned} \tag{7}$$

where $L(M(x_i - x_c, y_i - y_c))$ is a label of the patch M at $(x_i - x_c, y_i - y_c)$. Peaks in the $A(x_i, y_i)$ are most likely to have the best hypotheses in 3D (x_i, y_i, c_i) if wrong votes are randomly scattered. This random scattering is achieved by oversegmentation, as discussed in Sec. III-C.1. Eq. (7) is simply achievable by labeling each patch and accumulating its votes in the accumulator independently, as shown in Algorithm 1.

Algorithm 1: Pose hypothesizing algorithm

Given a set of votes (x_c, y_c, c_i) for each label L, and m questions designed in the training
1. Compute the image gradient in the input image
2. Choose pixels (x_j, y_j) on large image gradients
3. Allocate memory for the voting accumulator $A(x_i, y_i)$ and the sparse accumulator for poses $A_{p_i}(x_i, y_i)$
4. **for** *each pixel* (x_j, y_j) **do**
   - 4.1 Label (L) the image patch at (x_j, y_j) by asking the m questions
   - 4.2 Retrieve the set S of votes (x_{cl}, y_{cl}, c_l) of the label L, with cardinality $|S|$
   - 4.3 Compute $P(x_{cl}, y_{cl}, c_l | L) = 1/|S|$
   - 4.4 **for** *each vote* (x_{cl}, y_{cl}, c_l) **do**
     - Add $P(x_{cl}, y_{cl}, c_l | L)$ to $A(x_j + x_{cl}, y_j + y_{cl})$
     - Concatenate to $A_{p_i}(x_i, y_i)$ the pose index and its vote $\{c_l,\ P(x_{cl}, y_{cl}, c_l | L)\}$
5. Search for peaks (x_p, y_p) in $A(x_i, y_i)$
6. Retrieve the votes for all poses in the peaks (x_p, y_p) from A_{p_i}
7. Accumulate votes $P(x_c, y_c, c_i | L)$ for pose index c_i using neighbor poses
8. Search for the best pose hypotheses among (x_p, y_p, c_i)

Once the peaks in the 2D accumulator $A(x_i, y_i)$ are picked up, we search for the best pose hypotheses in the 3D distribution $P(x_i, y_i, c_i)$ only in the selected peaks (x_p, y_p), which is a simple one-dimensional search. For this search, we have another accumulator $A_{p_i}(x_i, y_i)$ that contains the votes for each pose at each pixel (x_i, y_i). Because only a few pose indices are voted for each pixel, $A_{p_i}(x_i, y_i)$ is an array of lists containing pose indices and votes for (x_i, y_i, c_i) for efficient memory usage and faster search.

Since the number of bins visited in the two-step search process greatly reduces compared to the original 3-dimensional search, that would have more than 12 million bins in our setup, generating pose hypotheses becomes much faster.

2) Neighboring poses: Similar poses tend to have very similar patches in the object database. However after reformulating the original 6D pose estimation into a 3D search

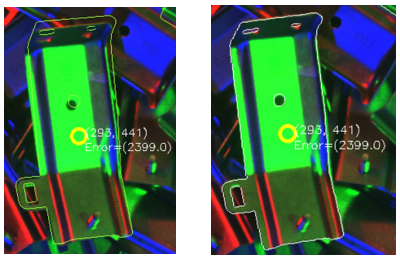


Fig. 6. Pose hypothesis by the proposed algorithm and refinement. Left: A pose hypothesis in (x_i, y_i, c_i) is misaligned. Right: After the pose refinement, the model is aligned accurately. (Best viewed in color)

for (x_i, y_i, c_i), the discretized pose index c_i does not contain information about the pose similarity anymore.

To alleviate this problem, we adopt another information aggregation after marginalization. For each peak (x_p, y_p) in $A(x_i, y_i)$, patches contribute to the similar pose hypotheses (x_i, y_i, c_i). All the votes within a local neighborhood at (x_p, y_p) accumulate their voting score to pose index c_i as well as their neighboring pose indices with a weight factor representing the pose difference. The neighboring poses are precomputed in the training stage. Each pose index has a fixed number of neighboring poses.

This additional information aggregation improves the robustness to errors by pose quantization. The online pose hypothesis generating algorithm is shown in Algorithm 1.

E. Pose Refinement

The generated pose hypothesis (x_i, y_i, c_i) is in discretized space. As the object pose of the input image in continuous 6D pose space may not be exactly the same as the discretized pose of the database image, the pose hypothesis is not accurate, as shown in Fig. 6. Denser sampling of the database poses may alleviate this problem, but can not eventually solve it. We describe how to upgrade the discretized pose into 6D continuous space, and our criteria for rejection of wrong pose hypotheses.

1) Procedure: To estimate a more accurate 6D pose, we refine the object pose starting from the discrete pose hypothesis. Assuming that the pose hypothesis is close to the object pose in 6D, an incremental pose update is made by using a visual servoing method. First, an object boundary of the pose hypothesis is extracted by projecting the CAD model in the image, and the corresponding 3D coordinates are collected from the CAD model. Once the computed object boundary is overlaid on the image, we search for the correspondence of each boundary pixel in the image. At each boundary pixel, we first compute the direction of the projected boundary and then choose the strongest gradient point along its perpendicular, within a small range, as the correspondence. We validate this correspondence by checking if the boundary pixel direction and the gradient directions at the corresponding image point are similar. After establishing all the correspondences between the input image and the CAD model boundary, the 6D part pose is updated by calculating the image Jacobians. This is a conventional visual servoing based object pose refinement procedure[17].

TABLE I

DETECTION PERFORMANCE OF 100-PART PICKING TEST

Rank	number of detections	false alarm	inaccurate pose
1	498 (99.6%)	1 (0.2%)	11 (2.2%)
2	447	4 (0.89%)	20 (4.47%)
3	341	1 (0.29%)	13 (3.81%)
4	218	4 (1.83%)	11 (5.04%)
5	96	1 (1.04%)	6 (6.25%)
Total	1600	11 (0.7%)	61 (3.8%)

2) Speeding-up: boundary precomputation: In practice, extracting 3D boundary points by rendering the object in a given pose takes a long time. To make it faster, we precompute all the 3D boundary points in each database image in advance. In the refinement process, only the precomputed 3D points are projected. This makes us to avoid the time-consuming boundary point computation in the iteration loop.

3) Rejecting hypotheses: Pose estimation using random ferns sometimes proposes wrong pose hypotheses, especially when a small number of objects exist in the image. We use the matching score of the pose refinement as an evidence of the existence. In searching for the boundary correspondences, we measure the ratio of valid matches out of all the points. If the ratio is less than a certain threshold, we simply reject the pose hypothesis.

IV. EXPERIMENTS

In this section, we show how the proposed method works step by step, and analyze its performance in accuracy, robustness, and computation time.

A. Pose estimation examples

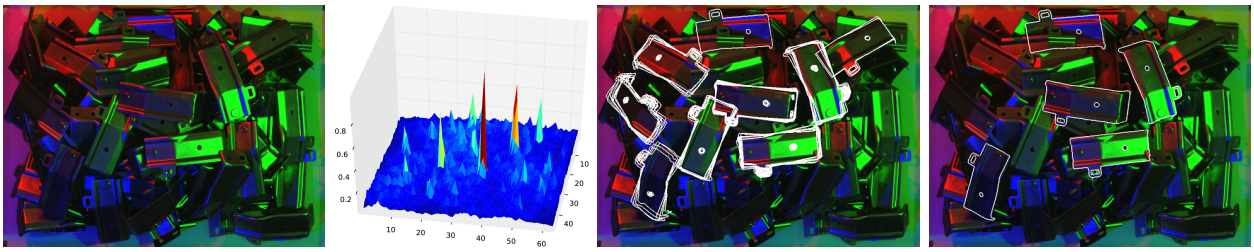
Fig. 7 shows the intermediate results of the proposed method searching for "bracket" objects. The multi-light image shown in Fig. 7(a) is captured by the proposed imaging system in Fig. 2. After voting from each image patch, there exist a few peaks in the marginalized 2D voting image $A(x_i, y_i)$ as shown in Fig. 7(b). Fig. 7(c) shows the pose hypotheses selected at the highest peak points, which are fairly accurate. Note that several pose hypotheses can be generated at a single (x_i, y_i) peak. The pose refinement and rejection provided accurate pose estimation in 6D space as shown in Fig. 7(d).

In Fig. 8, we show that the flexibility of the proposed method. Fig. 8(a) is an input multi-light image which contains two kinds of objects. Because the image has only four "bracket" objects, some of the pose hypotheses from 10 peaks are wrong, as shown in Fig. 8(b). Fig. 8(c) shows that the pose refinement successfully rejects all the wrong hypotheses.

On the other hand, detecting the other objects is successfully done by just changing the object-specific database as shown in Fig. 8(d). This flexibility is appreciated for handling many different parts in a same system setup.

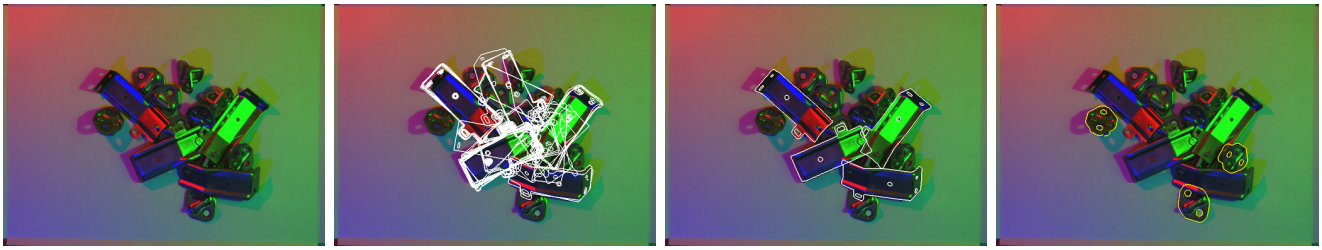
B. Detection performance

To test the detection performance of the proposed method, we designed a "100-part picking" test. At first, we randomly stacked 100 parts in the image field of view, and tried



(a) Input multi-light image (b) Voting in 2D (c) 50 pose hypotheses (d) Top 5 detections

Fig. 7. Step-by-step pose estimation procedure. We obtained the multi-light image (a) using the proposed imaging system with three lights in Fig. 2. After voting, the candidate object locations are collected from the marginalized votes (b). Then, pose hypotheses (c) are at the candidate object locations, which are close to the actual object poses. The final detection (d) is obtained through pose refinement. (Best viewed in color)



(a) Input multi-light Image (b) 50 pose hypotheses (c) Detection results (d) Detection of another object

Fig. 8. Object specific detection. The scene has two kinds of objects (a), and some of the pose hypotheses (b) are incorrect. The pose refinement successfully rejects the incorrect hypotheses. The exactly same binary code can be used for other object (d) by changing the object-specific database. (Best viewed in color)

detecting object poses using the proposed method. Once the proposed algorithm detected object poses, we picked up one object, and repeated the detection again. We conducted the test five times, and processed 500 images in total. The proposed method reports the matching score of the detected poses, and Table I shows the statistics of the best 5 pose hypotheses. The method failed to detect object poses only in two images out of 500. In total, 1600 poses were detected and the overall false alarm rate is 0.7%. For the best pose, only one false alarm happened out of 500. The pose estimation never failed for an image with less than 5 objects.

Sometimes, pose refinement is trapped by the nearby strong image gradient. It happened 3.8% in total and 2.2% for the best pose detected.

C. Accuracy

As we discussed in Section II-B, the light sources in the system are not far enough to be parallel, and surfaces in the same orientation at different position may have different colors. Because of the position dependency, the database image is not identical to the input object image, even though the object orientation is the same.

To test the accuracy depending on the position, we conducted experiments by changing the orientation and position of the object. We located an object right under the camera at first, and moved it using a rotation stage and a linear guide in each direction. At each position, we collected 100 multi-light images to check the repeatability. Fig. 9 shows the result statistics.

Because the camera is located at 1750 mm high, which is much longer than the object size, rotation in X and Y axes and translation in Z direction is less accurate than the others. We noticed that the X, Y location and the rotation in Z axis is very accurate even though the multi-light images are position-dependent. This is because our method does not use the absolute intensity, but only rely on the intensity difference in each channel, as discussed in Section III-C.2.

TABLE II

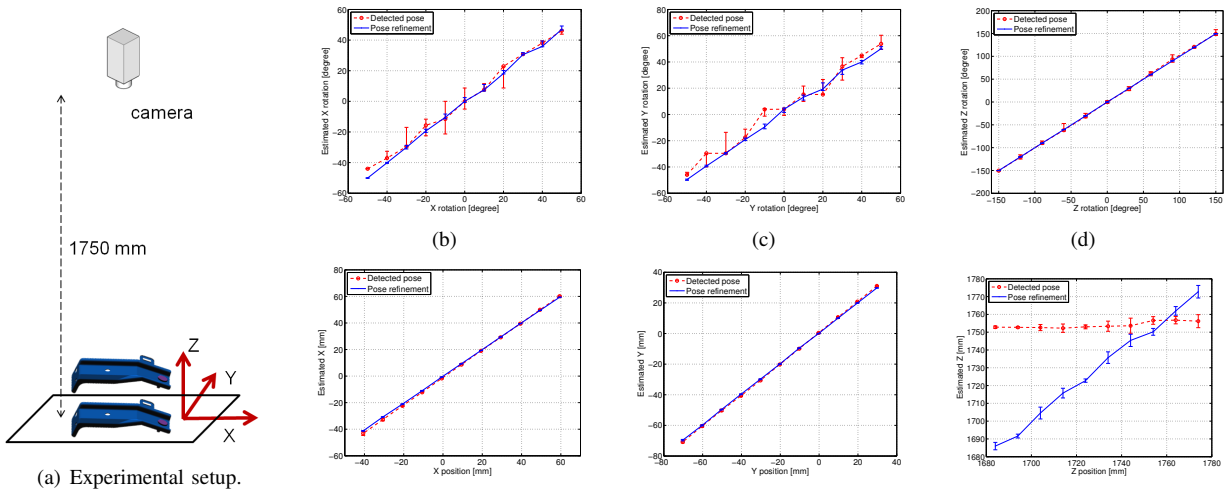
PROCESSING TIME FOR DETECTING UP TO 5 POSES WITH 50 POSE HYPOTHESES

Process	Average [ms]	St. Dev. [ms]
Choosing voting points	69.1	6.7
Labeling and Voting	136	41.7
Generate pose hypotheses	75.4	21.6
Testing pose hypotheses	164	35.6
Total	445	76.5
HALCON [-30°,30°]	839	101
HALCON [-50°,50°]	2446	243

In addition, the pose refinement improves the performance significantly. A pose hypothesis is one of the discretized poses in the object database. Fig. 9 shows that the pose refinement corrects the error by the discretization. Rotation estimations in X and Y axes are also stabilized well. In estimating a depth (Z position), one can note that all the pose hypotheses have the same depth, because the database images were collected with the same depth (around 1750 mm). The pose refinement successfully estimates the depth.

D. Speed

Table II shows the computation time in each online process for detecting up to 5 poses. We used a 3.2GHz Intel QuadCore processor with 3GB memory for this test. In this case, at most 50 pose hypotheses were tested after picking up 10 peaks in the marginalized 2D voting image $A(x_i, y_i)$. We



(a) Experimental setup. (b) (c) (d) (e) (f) (g)

Fig. 9. Accuracy analysis in various poses. The camera was located about 1750mm high from the object and the axis is set as shown in (a). Under rotations in each axis, the proposed method accurately estimate the rotations (b,c,d). Translations in each axis were also well-estimated after the pose refinement (e,f,g). For each pose, 100 trials were made.

used a set of 100-picking test data for this analysis. Speed of the labeling process depends on the number of voting points, which are determined by the complexity of the input image. In average, the whole process is done in about 500 ms per image. Testing hypotheses by pose refinement takes a longest time, and it is linearly proportional to the number of pose hypotheses. If a user wants to detect just one pose, the number of hypotheses can be reduced. Compared to a commercial implementation HALCON by MVTec [7], the proposed method runs faster. The processing time of the HALCON system using template matching depends on the pose coverage. For similar pose coverage of [-50°,50°], the proposed method runs about 6 times faster.

V. CONCLUSIONS

We proposed a practical method for detecting and localizing 3D shiny objects. The appearance of a metal surface greatly changes with the illumination direction, the viewpoint and its orientation, making it hard to detect.

By using our data-driven method for pose estimation, the large appearance variation becomes useful, since patches are very informative about pose. We build a inexpensive multi-light imaging system where the image color changes with the surface normal, making the patches even more distinctive. The detection and localization problem is reformulated as a database search problem, and the large diversity of the appearance helps the search. The database search is achieved by aggregating observations of image patches. Specifically, each input image patch corresponds to one cluster of patches in the database, and votes for possible locations and poses. The corresponding cluster is indexed by m binary questions which are selected randomly. Pose hypotheses with most votes are tested using the subsequent pose refinement.

Experiments show that the proposed detection and localization is successfully done in about 500 milliseconds. Additionally, applying it to another object is done by just changing the object database, without the need for any code or parameter modification.

REFERENCES

[1] P. David and D. DeMenthon, "Object recognition in high clutter images using line features," in *ICCV'05*, vol. 2, pp. 1581–1588.

[2] D. G. Lowe, "Three-dimensional object recognition from single two-dimensional images," *Artificial Intelligence*, vol. 31, pp. 355–395, 1987.

[3] D. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, vol. 60, no. 2, pp. 91–110, 2004.

[4] V. Lepetit and P. Fua, "Keypoint recognition using randomized trees," *IEEE T-PAMI*, pp. 1465–1479, 2006.

[5] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua, "Fast keypoint recognition using random ferns," *IEEE T-PAMI*, vol. 32, no. 3, pp. 448–461, 2010.

[6] E. Tola, V. Lepetit, and P. Fua, "A Fast Local Descriptor for Dense Matching," in *CVPR'08*.

[7] M. Ulrich, C. Wiedemann, and C. Steger, "CAD-based recognition of 3d objects in monocular images," in *ICRA'09*, pp. 1191–1198.

[8] S. Hinterstoisser, V. Lepetit, S. Ilic, P. Fua, and N. Navab, "Dominant orientation templates for real-time detection of texture-less objects," in *CVPR'10*.

[9] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, V. Lepetit, "Gradient Response Maps for Real-Time Detection of Texture-Less Objects," IEEE T-PAMI, vol.PP, no.99, pp.1, 2011.

[10] C. Steger, "Occlusion clutter, and illumination invariant object recognition," in *IAPRS'02*.

[11] R. Strzodka, I. Ihrke, and M. Magnor, "A graphics hardware implementation of the generalized hough transform for fast object recognition, scale, and 3D pose detection," in *ICIAP'03*, pp. 188–193.

[12] V. Ferrari, F. Jurie, and C. Schmid, "From images to shape models for object detection," *IJCV*, vol. 87, no. 3, pp. 284–303, 2010.

[13] J. Gall and V. Lempitsky, "Class-specific Hough forests for object detection," in *CVPR'09*. pp.1022–1029.

[14] O. Barinova, V. Lempitsky, and P. Kohli, "On detection of multiple object instances using Hough transforms," in *CVPR'10*.

[15] D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*, Prentice Hall, Aug. 2002.

[16] Y. Hwang, J. Kim, and I. Kweon, "Sensor noise modeling using the Skellam distribution: Application to the color edge detection," in *CVPR'07*.

[17] T. Drummond and R. Cipolla, "Real-time visual tracking of complex structures," *IEEE T-PAMI*, pp. 932–946, 2002.

[18] M. Liu, O. Tuzel, A. Veeraraghavan, R. Chellappa, A. Agrawal, H. Okuda, "Pose estimation in heavy clutter using a multi-flash camera," in *ICRA'10*, pp.2028-2035.

[19] P. F. Felzenszwalb and J. D. Schwartz, "Hierarchical Matching of Deformable Shapes," in *CVPR'07*.