Real-time scalable 6DOF pose estimation for textureless objects

Zhe Cao[1], Yaser Sheikh[1], and Natasha Kholgade Banerjee[2]

***Abstract*— Real-time recognition of the 6DOF pose of textureless objects is a fundamental and challenging problem in robotics. We present a novel approach to perform real-time estimation of the viewpoint, scale, and translation of an object in RGB and RGB-D image captures. In this work, we use a 3D model to render example poses of a textureless object, and find the nearest match to the input image using a GPU implementation. To achieve invariance to illumination and appearance across an object, we transform images to the Laplacian of Gaussian space. To perform real-time matching, we introduce a novel reshaping of the template set and the image, and we restructure the traditional normalized cross-correlation operation to leverage the GPU for fast matrix-matrix multiplication. We provide further speed up of large-scale template matching by contributing a dimensionality reduction approach using principal component analysis, and a candidate elimination method. Our method achieves state-of-the-art performance as shown by qualitative results and quantitative comparisons to pre-existing methods.**

I. Introduction

In robotics and computer vision, real-time recognition of the 6DOF pose of an object is an essential function for many applications such as robotic manipulation [1], [2] and automated visual inspection [3]. Approaches to 3D pose estimation can be divided into feature-based methods and template-based methods. Feature-based methods such as those of Collet et al. [1] and Xie et al. [4] show high accuracy on textured objects. However, they are unable to deal with textureless objects due to their reliance on discriminating features. In contrast, template-based methods, such as the approach in this paper, perform accurate pose estimation on textureless objects by matching object shape to templates from several viewpoints, scales, and translations of the object. Additionally, those approaches can efficiently transfer pre-computed data in the templates to test images, e.g., they may transfer grasping points from templates to detected objects for robotic manipulation.

Current approaches focus on improving template-based methods in three directions. The first is to make the method robust to environment illumination, object shape, and image appearance by using robust feature descriptors such as HOG [6] and DOT [7]. However, in providing invariance, feature descriptors such as HOG and DOT lose pixel-level detail. In this work, we propose a similarity measurement based on Laplacian of Gaussian filtering to gain invariance to illumination and object appearance while retaining pixel information pertaining to object shape. The second improvement direction is to speed up matching the image to a library of several thousand templates. Approaches to solve the speed up problem include matching cascades [8] or a hierarchical tree structure of templates [9], [10]. Unlike hierarchical approaches where candidates are eliminated early in the algorithm by the hierarchy, we provide an approach that uses a graphics processing unit (GPU) to provide match scores for all templates. The contributions of our approach are (a) novel GPU-based vectorized normalized cross-correlation algorithm that vectorized templates and image patches into matrices, (b) dimensionality reduction on the template matrix, and (c) GPU-based elimination of unlikely candidates in the final stage of the approach. The third improvement direction of template-based methods is to enhance accuracy in heavily cluttered environment. With the availability of RGB-D sensors, several approaches leverage the depth image to improve recognition accuracy in cluttered environments. Linemod [11] uses depth to calculate the normals as part of the feature descriptor. The approach in this paper focuses on performing GPU-based template matching in intensity space, and hence works on both RGB and RGB-D images. If the depth channel is available, we use the depth value at each pixel location to estimate an approximate object scale so as to limit the search space.

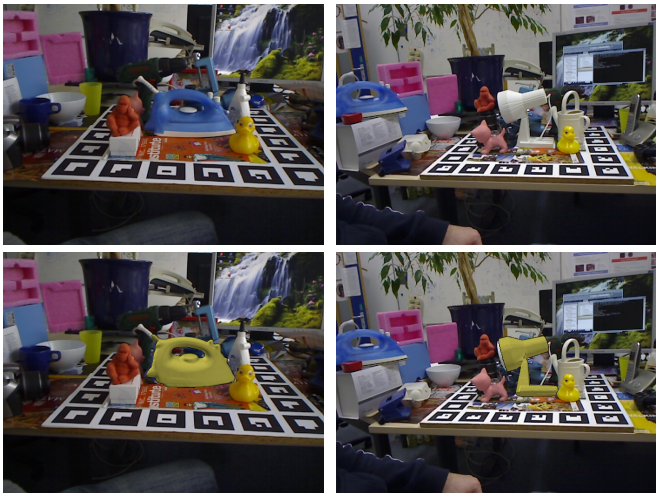

Fig. 1: Object pose estimation results of heavily clustered images. The first row: images of the iron and lamp model from ACCV12 dataset [5]. The second row: model alignment results using the estimated object pose. Two models are rendered with yellow texture for distinction.

[1]Zhe Cao, Yaser Sheikh are with Robotics Institute, Carnegie Mellon University, Forbes Avenue 5000, Pittsburgh, PA 15213, USA zhecao@andrew.cmu.edu, yaser@cs.cmu.edu

[2]Natasha Kholgade Banerjee is with the Department of Computer Science, Clarkson University, 8 Clarkson Ave, Potsdam, NY USA nbanerje@clarkson.edu

The contribution of this paper is a GPU-based approach to perform real-time textureless object pose estimation by framing normalized cross-correlation based template matching as matrix-matrix multiplications on the GPU, and exhaustively searching over the viewpoints, scales, and translations of the object in an image. The motivation behind using normalized cross-correlation is that it can be written as a simple dot product between a normalized template and image patch, an operation which inherently provides load balancing in contrast to feature descriptors or distance metrics based on searching within a space such as chamfer distance [9] or DOT [7]. We convert the images to the Laplacian of Gaussian space because normalized cross-correlation on images with high frequency information such as quantity of edge content captured by the Laplacian of Gaussian provides sharper peaks in the match result as opposed to normalized cross-correlation on the original intensity images themselves.

Our approach is a purely detection-based approach, and does not perform tracking. We achieve state-of-the-art speed and scalability amongst detection-based approaches for pose estimation for multiple objects from RGB-D input. As an example, our method estimates the pose for 15 objects in an image of size 320×240 at the rate of 23 frames per second, where each object has templates sampled with 600 viewpoints and continuous scales. Our method is scalable in the number of objects as it shows sub-linear run-time increase with multiple objects. This is a crucial step toward the problem of large-scale RGB-D object pose estimation.

Fig. 2: Object pose estimation results on our dataset. The first row: images of the bottle with different appearance and lighting conditions. The second row: model alignment results using the estimated object pose. Our method is robust to varying illumination and object appearance.

II. Related Work

A popular approach to the 6DOF pose estimation problem is to find correspondences from points [1], [12], [13] and parts [14], [15]. Felzenszwalb et al. [14] use the deformable parts model (DPM) to do category-level object detection. Their approach provides a bounding box and a corresponding label as output. Recent approaches provide extensions to DPM to perform viewpoint estimation. Gu et al. [16] perform joint category detection and viewpoint classification by learning a mixture of templates for each model. Lim et al. [17] leverage appearance cues from real images and geometric cues from 3D models to accurately estimate the fine pose of furniture. Several approaches provide view-specific classifiers using Exemplar-SVMs [18], [19]. Malisiewicz et al. [18] define a large range of object poses as an exemplar, and train an independent SVM classifier for each pose exemplar in the training dataset. Aubry et al. [19] leverage the exemplar based method [18] to train mid-level visual elements for estimation of identies and poses of chairs in images.

Several approaches incorporate depth from RGB-D data to improve the performance of object detection and pose estimation. Drost et al. [20] provide a voting approach to use oriented point pair features for object pose estimation. Saenko et al. [21] use depth information to infer the segmentation of objects, and to impose size priors on objects. Brachmann et al. [22] predict 3D object coordinates and object class labelling in a single RGB-D images using random forest. Kim et al. [23] localize an object in a 3D scene by generating object segmentation hypotheses from depth images and using a structural SVM to provide the best location. Tejani et al. [24] use latent-class Hough forests to combine Linemod features into a scale-invariant patch descriptor. Song et al. [25] train an Exemplar SVM classifier per rendered depth image, and run the classifier as a sliding window to detect the object. The computational complexity of these approaches limits their ability to perform real-time object pose estimation.

Real-time approaches to 3D pose estimation can be divided into tracking-based methods and detection-based methods. Tracking-based methods [26], [27], [28], achieve real time rates for 3D pose estimation by leveraging neighboring frames. Detection-based methods address the acceleration of exhaustive search in the space of all possible object poses. Gavrila et al. [9] and Ulrich et al. [10] organize templates in a hierarchical structure to reduce computation cost. Rios-Cabrera et al. [8] use discriminatively trained cascades of templates to improve the speed of pose estimation. To perform real-time 3D pose estimation for textureless objects, Hinterstoisser et al. [7] provide dominant orientation templates (DOT), where the template is represented as a grid of dominant orientations in several image patches. Linemod [11] achieves robust 3D object detection and pose estimation by combining DOT features from color images and normal orientations from depth images. Hinterstoisser et al. [5] improve Linemod by adding a post-processing step which begins with color and depth check to remove outliers, and follows with a fine ICP adjustment.

The approach provided in this paper differs from prior approaches in that it restructures template matching as a large-scale matrix-matrix multiplication. By doing so, the approach provides speed up in two ways: (1) it leverages hardware units dedicated to multiplication and addition on GPUs, and the inherent ease of load balancing matrix-matrix multiplications on GPUs, and (2) it leverages the ability to decompose matrices into factors of reduced dimensions.

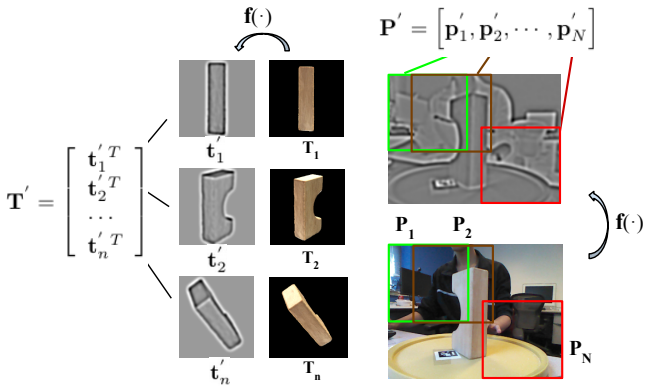

Fig. 3: Image matrix $\mathbf{P}'$ and template matrix $\mathbf{T}'$. T_i represents the original template generated from rendering the 3D model, where $i \in \{1, 2, \ldots, n\}$. After being transformed into $\mathbf{t}'_i$ using Equation (1), each template is vectorized into one row in the template matrix $\mathbf{T}'$. Similarly, P_j is the image patch with the same size as the template at all possible location in the image, where $j \in \{1, 2, \ldots, N\}$. After being transformed into $\mathbf{p}'_j$ using Equation (1), each image patch is vectorized into one column in the image matrix $\mathbf{P}'$.

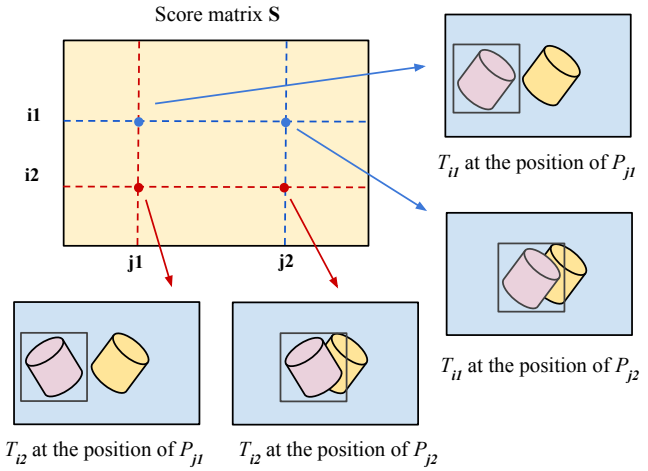

Fig. 4: Illustration of the score matrix $\mathbf{S}$. $\mathbf{S}(i, j)$ represents the correlation value of the i^{th} template T_i with the j^{th} image patch P_j.

III. Vectorized Similarity Measure

Our approach takes the 3D model of an object and a single RGB image or single RGB-D image as input. Our aim is to achieve exhaustive coverage of the viewpoints, scales, and translations of the object in the image, while achieving invariance to illumination. To exhaustively span viewpoint, we render the 3D model of the object at several rotations to get n templates. To exhaustively span translation, we simultaneously match all templates to all possible patches in an image. To exhaustively span scale, we extract image patches over a variety of scales and resize them accordingly to the same size of the template. For RGB images, we resize the image over five scales and extract patches, while for RGB-D images we use the depth value at each image patch center location to estimate an approximate object scale and then extract the resized patch.

We convert the color image into an intensity image $I \in \mathbb{R}^{W \times H}$ and each color template into an intensity template $T_i \in \mathbb{R}^{W' \times H'}$. Here, W is the width of the image I, H is the height of the image, $i \in \{1, 2, \ldots, n\}$, n is the number of templates, and W' and H' are respectively the width and height of the template T_i. We vectorize each template T_i by concatenating all pixels in the template row-wise into a column vector $\mathbf{t}_i \in \mathbb{R}^N$, where $N = W'H'$ is the number of pixels in the template. To match each template T_i to the image, we develop a similarity metric between T_i and each patch $P_j \in \mathbb{R}^{W' \times H'}$ at location j in the image I. Here $j \in \{1, 2, \ldots, m\}$, and m represents the number of image patches in I. As in the case of the template, we vectorize the image patch P_j into a vector $\mathbf{p}_j \in \mathbb{R}^N$.

To provide invariance to illumination, we transform each vectorized template $\mathbf{t}_i$ and patch $\mathbf{p}_j$ to $\mathbf{t}'_i$ and $\mathbf{p}'_j$ using function $\mathbf{f}(\cdot) : R^m \leftarrow R^m$, i.e., to compute $\mathbf{t}'_i = \mathbf{f}(\mathbf{t}_i)$, and $\mathbf{p}'_j = \mathbf{f}(\mathbf{p}_j)$. On transferring the patches using $\mathbf{f}$, we use cross-correlation to measure the similarity of the resulting two vectors. We obtain quasi-invariance to illumination, object appearance, and even slight deformation change if the transformation function $\mathbf{f}(\cdot)$ performs mean-variance normalization of the Laplacian of Gaussian of the image patch, i.e., if

$$\mathbf{f}(\cdot) = (\mathbf{f}_{mvnorm} \circ \mathbf{f}_{LoG})(\cdot), \text{ where} \tag{1}$$

$$\mathbf{f}_{mvnorm}(\mathbf{v}) = \frac{(\mathbf{v} - \mu_{\mathbf{v}})}{\sigma_{\mathbf{v}}}, \tag{2}$$

$\mu_{\mathbf{v}}$ and σ_v are the mean and standard deviation of the intensity values of a vectorized patch $\mathbf{v} \in R^N$, and

$$\mathbf{f}_{LoG}(\mathbf{u}) = (\Delta \mathbf{G}) * \mathbf{u}, \tag{3}$$

where $\mathbf{u}$ is a patch from the original image, $\mathbf{G}$ represents a 2D Gaussian, and Δ represents the Laplace operator.

We then represent the similarity between the transformed patches $\mathbf{t}'_i$ and $\mathbf{p}'_j$ as:

$$s = \mathbf{t}_i'^T \mathbf{p}'_j. \tag{4}$$

The value s represents the normalized cross-correlation between the Laplacians of Gaussian (LoG) of the original patches T_i and P_j.

By vectorizing the templates and patches, we leverage fast matrix-matrix multiplication on a GPU to speed up the computation. We set up a matrix $\mathbf{T}' = \begin{bmatrix} \mathbf{t}'_1 & \mathbf{t}'_2 & \cdots & \mathbf{t}'_n \end{bmatrix}^T$, which consists of concatenated vectorized templates transposed into rows, as shown in Figure 3. We similarly set up a matrix $\mathbf{P}' = \begin{bmatrix} \mathbf{p}'_1 & \mathbf{p}'_2 & \cdots & \mathbf{p}'_m \end{bmatrix}$, which consists concatenated vectorized image patches, as shown in Figure 3. Using the GPU, we compute a score matrix, $\mathbf{S} \in \mathbb{R}^{n \times m}$ where

$$\mathbf{S} = \mathbf{T}'\mathbf{P}'. \tag{5}$$

As shown in Figure 4, $\mathbf{S}(i,j)$ represents the normalized cross-correlation between the i^{th} template T_i and the j^{th} image patch P_j.

IV. Dimensionality Reduction in Large Matrix

To further speed up calculation of Equation (5), we precompute a dimensionality reduction on the matrix $\mathbf{T}'$, i.e., we decompose $\mathbf{T}'$ as

$$\mathbf{T}' = \mathbf{AZ}, \tag{6}$$

where $\mathbf{A} \in \mathbb{R}^{n\times k}$, $\mathbf{Z} \in \mathbb{R}^{k\times m}$, and $k << \min(n,m)$. We require

$$\mathbf{A}^T\mathbf{A} = \mathbf{I}, \tag{7}$$

where $\mathbf{I}$ is the $k \times k$ identity matrix. To obtain $\mathbf{A}$ and $\mathbf{Z}$, we use principal component analysis (PCA) to perform dimensionality reduction. In particular, we use singular value decomposition to express $\mathbf{T}'$ as

$$\mathbf{T}' = \mathbf{UDV}^T, \tag{8}$$

where $\mathbf{D} \in \mathbb{R}^{n\times m}$ represents a diagonal matrix of the same dimension as $\mathbf{T}'$, with non-negative diagonal elements in decreasing order, and $\mathbf{U} \in \mathbb{R}^{n\times n}$ and $\mathbf{V} \in \mathbb{R}^{m\times m}$ are orthogonal matrices. Reducing the dimensionality of $\mathbf{T}'$ is akin to expressing the templates as linear combinations of a reduced set of column vectors in $\mathbf{V}$, which turn out to be the principal components of ${\mathbf{T}'}^T\mathbf{T}$. We retain the top k principal components, such that

$$\frac{\sum_{i=1}^{k}\mathbf{D}_{ii}}{\sum_{i=1}^{n}\mathbf{D}_{ii}} \geq \alpha, \tag{9}$$

i.e., a variance of α is retained. In our experiments, we select α = 90%, and obtain an accuracy drop of no more than 5%.

We set up the matrix $\mathbf{A}$ by selecting the first k columns $\mathbf{U}_1,\ \mathbf{U}_2 \cdots \mathbf{U}_k$ from $\mathbf{U}$, i.e.,

$$\mathbf{A} = [\mathbf{U}_1, \mathbf{U}_2 \ldots \mathbf{U}_k]. \tag{10}$$

We define $\mathbf{Z}$ as the product of a diagonal matrix consisting of the first k diagonal elements $\mathbf{D}_{11},\ \mathbf{D}_{22}, \cdots, \mathbf{D}_{kk}$ of $\mathbf{D}$, and a matrix consisting of the first k column vectors $\mathbf{V}_1,\ \mathbf{V}_2 \cdots \mathbf{V}_k$ from $\mathbf{U}$ of $\mathbf{V}$ transposed, i.e.,

$$\mathbf{Z} = \text{diag}\left(\begin{bmatrix}\mathbf{D}_{11} & \mathbf{D}_{22} & \cdots & \mathbf{D}_{kk}\end{bmatrix}\right)\begin{bmatrix}\mathbf{V}_1 & \mathbf{V}_2 & \cdots & \mathbf{V}_k\end{bmatrix}^T. \tag{11}$$

By combining Equations (5) and (6), we get

$$\mathbf{S} = \mathbf{AZP}'. \tag{12}$$

We perform the multiplication in Equation (12) on the GPU in two steps. We first compute

$$\mathbf{Q} = \mathbf{ZP}'. \tag{13}$$

Next, we compute

$$\mathbf{S} = \mathbf{AQ}. \tag{14}$$

Section V provides an additional speed up on the computation of the score matrix in Equation (14) by eliminating unlikely image locations in $\mathbf{Q}$.

V. Elimination of Unlikely Image Locations

Instead of calculating every element in the score matrix $\mathbf{S}$ in Equation (14), we speed up the calculation of $\mathbf{S}$ by first eliminating image locations where a high score is unlikely. Specifically, we eliminate columns in $\mathbf{Q}$ and get $\mathbf{Q}' = [\cdots \mathbf{Q}_j \cdots]$, where $j \in \mathcal{J}$, and $\mathcal{J}$ is a reduced subset of indices into the image matrix columns corresponding to image locations with higher match scores. We obtain the final score matrix $\mathbf{S}'$ as for likely pixel locations in the image as

$$\mathbf{S}' = \mathbf{AQ}'. \tag{15}$$

Theorem 5.1 proves that if the L_2 norm of the j^{th} column of $\mathbf{Q}$, i.e., $\|\mathbf{Q}_j\|$ is bounded above by λ, then every element $\mathbf{S}(i,j)$ in the corresponding column $\mathbf{S}(j)$ in the large score matrix is bounded above by λ, that is, the match score of every template at the i^{th} pixel in the image is below λ. This observation allows us to eliminate the j^{th} image patch P_j as unlikely if $\mathbf{Q}_j \leq \lambda$.

In experiments on the ACCV12 RGB-D dataset, we choose λ to be 0.8. The number of columns in $\mathbf{Q}'$ are reduced to 70% of the number of columns in $\mathbf{Q}$. We find a 50% reduction in computation time using the PCA and candidate elimination methods.

Theorem 5.1: if $\|\mathbf{Q}_j\| < \lambda$, then $\forall i,\ \mathbf{S}(i,j) < \lambda$.

Proof: From Equation (15),

$$\begin{aligned}\|\mathbf{S}(j)\| = \|\mathbf{AQ}_j\| &= \sqrt{\|\mathbf{AQ}_j\|^2} = \sqrt{\mathbf{Q}_j^T\mathbf{A}^T\mathbf{AQ}_j} \\ &= \sqrt{\mathbf{Q}_j^T\mathbf{Q}_j} \text{ from Equation (7)} \\ &= \|\mathbf{Q}_j\|\end{aligned} \tag{16}$$

If $\|\mathbf{Q}_j\| < \lambda$, then $\|\mathbf{S}(j)\| \leq \lambda$ or $\sum_{i=1}^{m}\mathbf{S}(i,j)^2 \leq \lambda^2$.

Since $\forall i,\ \mathbf{S}(i,j) \geq 0$, hence $\forall i,\ \mathbf{S}(i,j) \leq \lambda$. ■

We generate the template matrix $\mathbf{T}'$ and perform PCA on it offline, and we generate the image matrix $\mathbf{P}'$, obtain the score matrix $\mathbf{S}'$, and perform the matrix-multiplication and elimination of unlikely locations online. We divide the steps discussed in Sections III to V as offline steps detailed in Algorithm 1 and online steps detailed in Algorithm 2. Both algorithms leverage the GPU to perform operations such as vectorization, mean-normalization, and cross-correlation. We use the cuBLAS API for matrix-matrix multiplications.

Algorithm 1 Offline Template Matrix Generation and PCA

Input: All templates T_1 ...T_n, PCA variance α.
Output: Coefficient matrix A, basis matrix Z

1: **for all** templates T_i **do**
2: Perform LoG(T_i) using one thread per pixel in the LoG.
3: Mean normalize T_i using one thread per pixel in the mean-normalized template.
4: Reshape T_i to one column in the template matrix $\mathbf{T}'_i$.
5: **end for**
6: $\mathbf{A}, \mathbf{Z} \Leftarrow PCA(\mathbf{T}'_i)$: performed on the CPU.

Algorithm 2 Online Large-Scale Template Matching

Input: Matrices $\mathbf{A}$and $\mathbf{Z}$, threshold λ.
Output: Score matrix $\mathbf{S}$ for each frame.

for each new frame I **do**
  Perform LoG(I) using one thread per pixel in the LoG.
  Call one parallel thread for each image patch P_j.
  for all threads **do**
    Mean normalize P_j.
    Reshape P_j to one column in the image matrix $\mathbf{P}'_j$.
  end for
  Compute $\mathbf{Q} \Leftarrow \mathbf{ZP}'$ using cuBLAS.
  $\mathbf{L} \Leftarrow$ empty vector.
  Call one parallel thread for each column j in $\mathbf{Q}$.
  for all threads **do**
    $\mathbf{L}_j \Leftarrow \|\mathbf{Q}_j\|$
  end for
  $m \Leftarrow 1$, $\mathbf{Q}' \Leftarrow$ empty matrix.
  for $j = 1$ to N **do**
    if $\mathbf{L}_j > \lambda$ **then**
      $\mathbf{Q}'_m \Leftarrow \mathbf{Q}_j$, $m \Leftarrow m + 1$;
    end if
  end for
  Compute $\mathbf{S}' \Leftarrow \mathbf{AQ}'$ using cuBLAS.
end for

Fig. 5: 3D models in our dataset: cereal box, oil can, sugar bag, plate, bowl, soy milk container, IKEA mug, toy wooden car, plush toy, cup, toy wooden pumpkin, coffee creamer container, round-bottomed mug, wooden toy bridge, teapot, and book (left to right from top row down).

VI. Experiments

In this section, we compare our method to both RGB and RGB-D approaches on two datasets: (1) Our indoor objects dataset consisting of sixteen 3D objects, and RGB images with ground-truth annotation, and (2) the ACCV12 dataset [5] consisting of fifteen 3D models and RGB-D images with ground-truth object pose. All results were obtained using a standard desktop computer with an NVIDIA Quadro K6000 GPU consisting of 2880 cores and 12 GB memory. We show qualitative results in Figures 1 and 7.

Indoor objects RGB dataset. Our indoor objects dataset consists of 3D models for six textured objects (a cereal box, a sugar bag, a soy milk container, a textbook, a plush toy, and a toy wooden pumpkin), and ten non-textured objects (an oil can, a plate, a bowl, an IKEA mug, a toy wooden car, a cup, a coffee creamer container, a round-bottomed mug, a wooden toy bridge, and a teapot). The dataset also contains 248 test RGB images of the objects together with ground truth pose by manual annotation. Figure 5 shows a detailed view of our 3D models. For each object, we captured RGB images of size 640×480 using a Microsoft Kinect v1 sensor from a variety of viewpoints. We obtained 3D models for the IKEA mug, the bowl and the plate from 3D Warehouse. We used Autodesk 123D Catch to create 3D models for the rest of objects from multiple high resolution images. For the teapot and cup, we applied fiducials to the objects using tape and markers. In the training stage, we rendered the 3D model to get a set of 600 template images, with 24 samples in yaw, 5 samples in pitch, and 5 samples in roll spaced 15 degrees apart. We create a single sample for the angle around the axis of rotational symmetry for objects such as the bowl and the bottle in Figure 5.

We evaluate the accuracy of our objects for each angle and translational axis by computing

$$err_{\text{angle}} = |\theta_{\text{angle}} - \alpha_{\text{angle}}|,$$
$$err_{\text{ax}} = |P_{\text{ax}} - R_{\text{ax}}|, \quad (17)$$

where angle $\in$ {pitch, yaw, roll} represents one of the angles of pitch, yaw, and roll, err_{angle} represents the error in the particular angle, θ_{angle} represents the ground truth angle value, α_{angle} represents the estimated angle value, ax $\in \{x, y\}$ represents one of the $x-$ and $y-$axes, err_{ax} represents the error along the particular axis, P_{ax} represents the coordinate along the particular axis for the ground truth position vector, and R_{ax} represents the estimated coordinate along that axis for the position vector. If the object is rotationally symmetric about an axis, the error for the angle corresponding to the axis is set to 0. For instance, the yaw error of the bowl is set to 0. In the case of RGB images, we do not evaluate the error in depth, i.e., in z.

Table I shows the comparison between our method (VNCC), Line2D [29] and our method with PCA training (VNCC-PCA). We evaluate both methods using two strategies, top one hypothesis and the best from top five hypotheses. On average, the angular error of VNCC is 12% smaller than Line2D and the translational error of VNCC is 35% smaller than Line2D. The main reason is that Line2D uses DOT as feature descriptor, which loses pixel-level detail and gets only approximated detected position. In contrast,

our method uses Laplacian of Gaussian feature and performs pixel-wise comparison with the original image. Table I also indicates that VNCC-PCA can decrease the run-time by 25% while obtaining an accuracy 95% of that of the VNCC approach.

	pitch	roll	yaw	x	y	run-time
VNCC-1	5.13	6.51	12.48	10.26	8.17	159 ms
Line2D-1	5.67	7.38	13.92	15.62	13.43	283 ms
VNCC-5	**2.71**	**5.43**	**6.35**	**5.27**	**4.28**	162 ms
Line2D-5	3.05	6.12	7.88	9.35	7.24	288 ms
VNCC-PCA-5	2.92	5.56	6.42	5.43	4.47	**119 ms**

TABLE I Comparison between our method (VNCC) and Line2D. The first five columns show the average errors of each angle (in degrees), each axis (in pixels). The sixth column gives the average run-time per image.

ACCV12 RGB-D dataset. Hinterstoisser et al. [5] provide 15 textureless objects, $1000+$ images of each object, and ground truth poses. They achieve 96.6% accuracy on average on this dataset. The post-processing step is important for achieving the reported accuracy. As part of the post-processing, they perform a rough color check, a fast voxel-based iterative closest point (ICP) estimation [30] to remove outliers, and a finer ICP adjustment to refine the pose estimation result. Rios-Cabrera et al. [8] also perform coarse and fine ICP to remove outliers. We use the same color check step as [5], however, instead of a depth check and fine ICP process, we use multiple hypotheses for the purpose of selecting the best matching template.

Table III shows the accuracy comparison between our method (VNCC), DDT-3d [8], Hinterstoisser et al. [5], Linemod [11] and Drost et al. [20]. We use the same viewpoint sampling method and accuracy evaluation metric as Hinterstoisser et al. [5]. Since Rios-Cabrera et al. [8] and Hinterstoisser et al. [5] evaluate their methods with an additional depth check and fine ICP adjustment, for a fair comparison, we evaluated our method using two strategies: (1) the top one detected template or hypotheses, and (2) the best detected template (compared with the ground-truth) out of the top ten hypotheses. Our one hypothesis approach (VNCC-1) outperforms Linemod [11] and Drost et al. [20]. Our top ten hypotheses approach (VNCC-10) achieves 96% accuracy on average which is slightly worse than Hinterstoisser et al. [5] and DDT-3d. However, our method still has accuracy improvement space after applying ICP adjustment.

Table II shows the average run-time comparison between our method (VNCC), our method with PCA training (VNCC-PCA), DDT-3d [8], Hinterstoisser et al. [5]. To measure the scalability of each method, we also compare the run-time for processing different numbers of objects. Our method can be easily extended to the case of multiple objects detection. For multiple objects, we combine the template matrix of each object into the whole template matrix, and thus we can obtain each object score matrix from the corresponding region in the complete score matrix. We obtain multiple hypotheses for each object simultaneously without sacrificing accuracy. By using the NVIDIA Quadro 6000 GPU with 12GB memory,

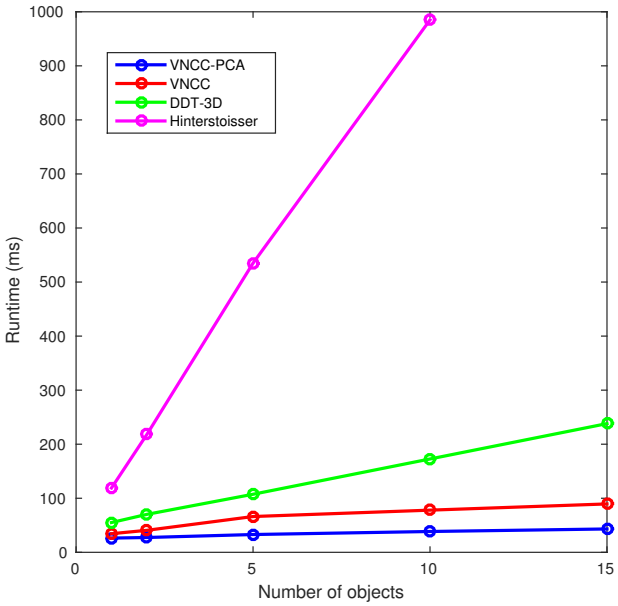

Fig. 6: Average run-time comparison of one image with increasing number of objects.

	VNCC-PCA	VNCC	DTT-3d [8]	Hinterstoisser [5]
one	26.3 ms	34.4 ms	55.1 ms	119 ms
two	27.4 ms	40.7 ms	70.2 ms	218 ms
five	32.9 ms	66.2 ms	107.4 ms	535 ms
ten	38.5 ms	78.0 ms	172.7 ms	985 ms
fifteen	43.2 ms	89.5 ms	238.1 ms	1388.2 ms

TABLE II Run-time comparison on the ACCV12 dataset: the first column shows the number of objects and the following columns show the run-time for each method. We outperform the state-of-the-art in speed and scalability.

single precision floating point arithmetic, and templates of size 90×90, we can store a score matrix of maximum dimension $27,000 \times 37,500$, together with corresponding image matrix and template matrix in memory, i.e., perform pair-wise matching for $27,000$ templates and $37,500$ image patches of size 90×90.

Figure 6 shows that the run-time of DDT-3d and Hinterstoisser et al. [5] increases linearly with increase in the number of objects. Due to the ability of the dimensionality reduction approach in Section IV to represent similarities across diverse objects within a few templates, our VNCC-PCA approach shows sub-linear increase in run-time with increase in number of objects. The PCA-based dimensionality reduction not only speeds up the method but also makes the method more memory-efficient.

Similar to DDT-3d and Hinterstoisser et al. [5], we perform an offline templates training once per object. DDT-3d requires 217 seconds and Hinterstoisser et al. [5] requires 54 seconds for traning one object. Our offline training step takes 25 seconds per object. For 10 objects, our method requires 8 minutes for the offline training. Most time is spent on the

	DDT-3d [8]	Hinterstoisser [5]	VNCC-PCA-10	VNCC-10	VNCC-1	Linemod [11]	Drost et al. [20]
Ape	95.0%	95.8%	95.6%	96.0%	74.8%	69.4%	86.5%
Bench Vise	98.9%	98.7%	97.4%	97.5%	89.8%	94.0%	70.7%
Driller	94.3%	93.6%	95.4%	95.8%	84.2%	81.3%	87.3%
Cam	98.2%	97.5%	95.5%	95.7%	80.4%	79.5%	78.6%
Can	96.3%	95.4%	96.1%	96.7%	83.9%	79.5%	80.2%
Iron	98.4%	97.5%	96.5%	96.8%	86.7%	88.8%	84.9%
Lamp	97.9%	97.7%	97.2%	97.3%	89.3%	89.8%	93.3%
Phone	95.3%	93.3%	93.3%	93.6%	79.6%	77.8%	80.7%
Cat	99.1%	99.3%	96.8%	97.0%	86.9%	88.2%	85.4%
Hole punch	97.5%	95.9%	94.3%	94.4%	80.2%	78.4%	77.4%
Duck	94.2%	95.9%	93.5%	93.7%	78.6%	75.9%	46.0%
Cup	97.5%	97.1%	95.1%	95.4%	83.2%	80.7%	68.4%
Bowl	99.7%	99.9%	99.1%	99.3%	94.2%	99.5%	95.7%
Box	99.8%	99.8%	99.2%	99.5%	95.6%	99.1%	97.0%
Glue	96.3%	91.8%	94.3%	94.2%	75.8%	64.3%	57.2%
Average	97.2%	96.6%	96.0%	96.2%	84.2%	83.0%	79.3%

TABLE III Accuracy comparison on the ACCV12 dataset using same evaluation metric in [5]: The first column shows the name of each image sequence. The second and third column show the accuracy of DDT-3d and Hinterstoisser et al. [5] respectively. In the fourth and fifth column, we evaluate the accuracy of VNCC-PCA and VNCC using the best out of the top ten matched templates compared to ground truth. The sixth column shows the accuracy of our method using the top one matching template. The seventh and eighth column show the accuracy of Linemod [11] and Drost et al. [20] respectively.

PCA dimensionality reduction on the template matrix.

VII. Discussion

In this paper, we provide a novel solution for the problem of real-time textureless object detection and pose estimation. Our method leverages GPUs to perform normalized cross-correlation in the Laplacian of Gaussian space re-formulated as large matrix-matrix multiplications. In Section VI, we validate that our method provides high accuracy of pose estimation on textureless objects under varying illumination, handles RGB and RGB-D images, provides a speed up of an order of magnitude in contrast to existing techniques for multiple objects, and shows scalability due to sub-linear increase of run-time with number of objects.

The main limitation of our approach is that while it provides accurate global view alignment, it does not account for local match precision in rigid 6DOF pose. Additionally, while it shows accuracy on slight deformations, it does not capture significant deformations such as those that may arise in aligning generic 3D models of natural objects such as fruit, vegetables, or animals from online repositories to specific instances of such objects. To address these limitations, future work includes using GPUs to accelerate the matching of local patches in the image, and using iterative closest-point algorithms and surface deformation algorithms to perform rigid and non-rigid alignments to several 2D-3D correspondences.

References

[1] A. Collet, D. Berenson, S. S. Srinivasa, and D. Ferguson, "Object recognition and full pose registration from a single image for robotic manipulation," in *ICRA 2009*. IEEE, 2009, pp. 48–55.

[2] M. Zhu, K. G. Derpanis, Y. Yang, S. Brahmbhatt, M. Zhang, C. Phillips, M. Lecce, and K. Daniilidis, "Single image 3d object detection and pose estimation for grasping," in *ICRA 2014*. IEEE, 2014, pp. 3936–3943.

[3] C. Zang and K. Hashimoto, "A flexible visual inspection system combining pose estimation and visual servo approaches," in *ICRA 2012*. IEEE, 2012, pp. 1304–1309.

[4] Z. Xie, A. Singh, J. Uang, K. S. Narayan, and P. Abbeel, "Multimodal blending for high-accuracy instance recognition," in *IROS 2013*. IEEE, 2013, pp. 2214–2221.

[5] 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*. Springer, 2013, pp. 548–562.

[6] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR 2005.*, vol. 1. IEEE, 2005, pp. 886–893.

[7] S. Hinterstoisser, V. Lepetit, S. Ilic, P. Fua, and N. Navab, "Dominant orientation templates for real-time detection of texture-less objects," in *CVPR 2010*. IEEE, 2010, pp. 2257–2264.

[8] R. Rios-Cabrera and T. Tuytelaars, "Discriminatively trained templates for 3d object detection: A real time scalable approach," in *ICCV 2013*. IEEE, 2013, pp. 2048–2055.

[9] D. M. Gavrila, "A bayesian, exemplar-based approach to hierarchical shape matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 8, pp. 1408–1421, 2007.

[10] M. Ulrich, C. Wiedemann, and C. Steger, "Cad-based recognition of 3d objects in monocular images." in *ICRA*, vol. 9, 2009, pp. 1191–1198.

[11] 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*. IEEE, 2011, pp. 858–865.

[12] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Computer vision and image understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[13] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[14] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *CVPR 2008*. IEEE, 2008, pp. 1–8.

[15] S. Savarese and L. Fei-Fei, "3d generic object categorization, localization and pose estimation," in *ICCV 2007*. IEEE, 2007, pp. 1–8.

[16] C. Gu and X. Ren, "Discriminative mixture-of-templates for viewpoint classification," in *ECCV 2010*. Springer, 2010, pp. 408–421.

[17] J. J. Lim, A. Khosla, and A. Torralba, "Fpm: Fine pose parts-based model with 3d cad models," in *ECCV 2014*. Springer, 2014, pp. 478–493.

[18] T. Malisiewicz, A. Gupta, and A. A. Efros, "Ensemble of exemplar-svms for object detection and beyond," in *ICCV 2011*. IEEE, 2011, pp. 89–96.

[19] M. Aubry, D. Maturana, A. A. Efros, B. C. Russell, and J. Sivic, "Seeing 3d chairs: exemplar part-based 2d-3d alignment using a large dataset of cad models," in *CVPR 2014*. IEEE, 2014, pp. 3762–3769.

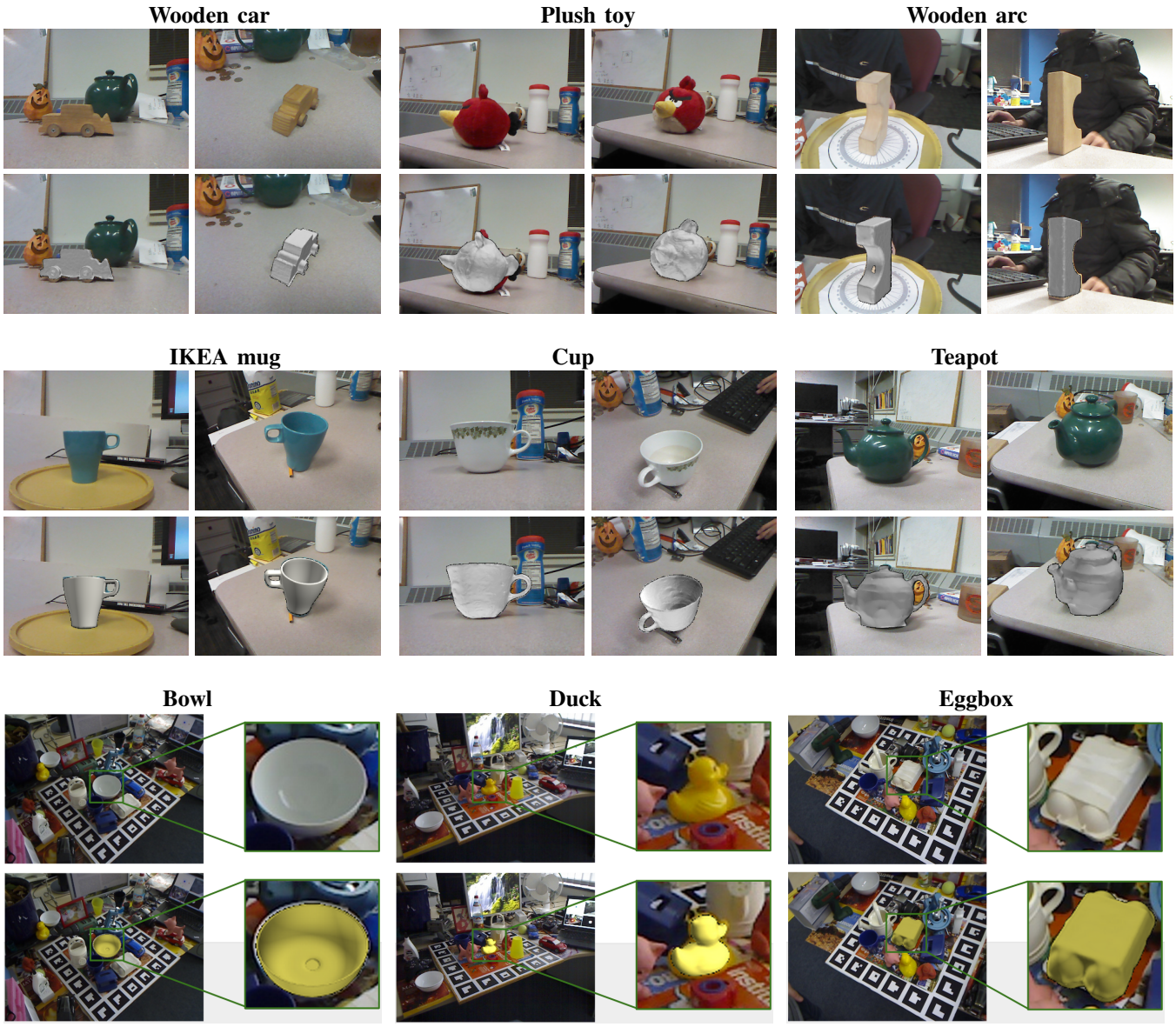

Fig. 7: Qualitative results on the wooden car, plush toy, wooden arc, IKEA mug, cup, and teapot from our dataset, and the bowl, duck and eggbox from ACCV12 dataset. For each object, we show the original image, and model alignment results using the estimated object pose.

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

[21] K. Saenko, S. Karayev, Y. Jia, A. Shyr, A. Janoch, J. Long, M. Fritz, and T. Darrell, "Practical 3-d object detection using category and instance-level appearance models," in *IROS 2011*. IEEE, 2011, pp. 793–800.

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

[23] B.-s. Kim, S. Xu, and S. Savarese, "Accurate localization of 3d objects from rgb-d data using segmentation hypotheses," in *CVPR 2013*. IEEE, 2013, pp. 3182–3189.

[24] A. Tejani, D. Tang, R. Kouskouridas, and T.-K. Kim, "Latent-class hough forests for 3d object detection and pose estimation," in *ECCV 2014*. Springer, 2014, pp. 462–477.

[25] S. Song and J. Xiao, "Sliding shapes for 3d object detection in depth images," in *ECCV 2014*. Springer, 2014, pp. 634–651.

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

[27] C. Choi and H. I. Christensen, "3d textureless object detection and tracking: An edge-based approach," in *IROS, 2012*. IEEE, 2012, pp. 3877–3884.

[28] K. Pauwels, L. Rubio, J. Diaz, and E. Ros, "Real-time model-based rigid object pose estimation and tracking combining dense and sparse visual cues," in *CVPR 2013*. IEEE, 2013, pp. 2347–2354.

[29] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit, "Gradient response maps for real-time detection of textureless objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 5, pp. 876–888, 2012.

[30] A. W. Fitzgibbon, "Robust registration of 2d and 3d point sets," *Image and Vision Computing*, vol. 21, no. 13, pp. 1145–1153, 2003.