Fast Object Localization and Pose Estimation in Heavy Clutter for Robotic Bin Picking

Liu, M-Y.; Tuzel, O.; Veeraraghavan, A.; Taguchi, Y.; Marks, T.K.; Chellappa, R.

TR2012-007 May 2012

Abstract

We present a practical vision-based robotic bin-picking system that performs detection and 3D pose estimation of objects in an unstructured bin using a novel camera design, picks up parts from the bin, and performs error detection and pose correction while the part is in the gripper. Two main innovations enable our system to achieve real-time robust and accurate operation. First, we use a multi-flash camera that extracts robust depth edges. Second, we introduce an efficient shape-matching algorithm called fast directional chamfer matching (FDCM), which is used to reliably detect objects and estimate their poses. FDCM improves the accuracy of chamfer matching by including edge orientation. It also achieves massive improvements in matching speed using line-segment approximations of edges, a 3D distance transform, and directional integral images. We empirically show that these speedups, combined with the use of bounds in the spatial and hypothesis domains, give the algorithm sublinear computational complexity. We also apply our FDCM method to other applications in the context of deformable and articulated shape matching. In addition to significantly improving upon the accuracy of previous chamfer matching methods in all of the evaluated applications, FDCM is up to two orders of magnitude faster than the previous methods.

The International Journal of Robotics Research

Fast Object Localization and Pose Estimation in Heavy Clutter for Robotic Bin Picking

Ming-Yu Liu[†*] Oncel Tuzel[†] Ashok Veeraraghavan[†‡]
Yuichi Taguchi[†] Tim K. Marks[†] Rama Chellappa[*]

[†]Mitsubishi Electric Research Laboratories (MERL)
[*]University of Maryland [‡]Rice University

Abstract

We present a practical vision-based robotic bin-picking system that performs detection and 3D pose estimation of objects in an unstructured bin using a novel camera design, picks up parts from the bin, and performs error detection and pose correction while the part is in the gripper. Two main innovations enable our system to achieve real-time robust and accurate operation. First, we use a multi-flash camera that extracts robust depth edges. Second, we introduce an efficient shape-matching algorithm called fast directional chamfer matching (FDCM), which is used to reliably detect objects and estimate their poses. FDCM improves the accuracy of chamfer matching by including edge orientation. It also achieves massive improvements in matching speed using line-segment approximations of edges, a 3D distance transform, and directional integral images. We empirically show that these speedups, combined with the use of bounds in the spatial and hypothesis domains, give the algorithm sublinear computational complexity. We also apply our FDCM method to other applications in the context of deformable and articulated shape matching. In addition to significantly improving upon the accuracy of previous chamfer matching methods in all of the evaluated applications, FDCM is up to two orders of magnitude faster than the previous methods.

1 Introduction

Building smarter, more flexible, and independent robots that can interact with the surrounding environment is a fundamental goal of robotics research. Potential applications are wide-ranging, including automated manufacturing, entertainment, in-home assistance, and disaster rescue. One of the long-standing challenges in realizing this vision is the difficulty of "perception" and "cognition"—i.e., providing the robot with the ability to understand its environment and make inferences that allow appropriate actions to be taken. Perception through inexpensive contact-free sensors such as cameras are essential for continuous and fast robot operation. In this paper, we address the challenge of robot perception in the context of industrial robotics.

1.1 Visual Perception in Industrial Robotics

Computer vision has made rapid progress in the last decade, moving closer to definitive solutions for longstanding problems in visual perception such as object detection [DT05, VJ01, TPM08], object recognition [FFFP06, OT01, DBdFF02], and pose estimation [AT06, SB06b, MREM04]. While the huge strides made in these fields lead to important lessons, most of these methods cannot be readily adapted to industrial robotics because many of the common assumptions are either violated or invalid in such settings.

Material properties: One of the most common assumptions in traditional vision algorithms relates to the characterization of the reflectance of materials in the scene. Most vision algorithms characterize materials as Lambertian [BJ03], i.e., the appearance (radiance) of a single surface point is invariant to the location of the observer (camera). While this is a reasonable assumption in many scenarios, this is less applicable to industrial vision tasks. Several common materials handled in industrial settings such as metal, glass, ceramics, and some plastics are not close to Lambertian. Hence, using the Lambertian assumption for such objects generally results in poor performance. This necessitates industrial robots to possess the ability to understand and make inferences about objects that have complex reflectance characteristics.

Environmental challenges: The types of errors that afflict vision-based systems in industrial settings are also very different from those in natural environments. Several industrial assembly and manufacturing tasks must be accomplished in dark or dimly lit environments with dust, dirt, grime, and grease. It is essential for vision-based techniques to be able to cope with such sources of error in order to be successful in such environments.

Variable appearance: The most popular methods for object detection, recognition, and pose estimation are based on the idea of feature descriptors such as Scale Invariant Feature Transform (SIFT) [Low04], Histogram of Gradients (HOG) [DT05], and SURF [BETG08]. The basic idea is to detect several keypoint locations on the surface of each object and compute these feature descriptors at these keypoint locations. The features of each object are then stored in a database. When a test image is acquired, the keypoint locations and the feature descriptors for the test image are computed and then matched to the features stored in the database. An appropriately computed matching score is used to detect and recognize objects, and the geometric relationship between the matched keypoint locations in the test image and the database are used to make inferences about the poses of the objects. This general principle is quite popular and is used in several object recognition methods [STFW05, LSP06, NFF07]. Unfortunately, this successful paradigm cannot be easily adapted to industrial robotics because visual appearance features are unreliable in industrial settings. Variable material properties, as well as uncontrolled illumination and environmental conditions, make appearance-based descriptors unreliable and preclude the use of such techniques in most industrial applications.

Background Clutter: The objects in a factory environment are usually stacked in part containers, which produces additional challenges such as overlapping parts, occlusions, cast shadows, and complex backgrounds. Therefore, most commercial vision systems assume that parts are separated in a kitting stage before operation. Machine vision systems that are capable of handling clutter, occlusions, and complex backgrounds would eliminate the need for kitting stages, thereby allowing such systems to handle a complex bin of parts.

Model-based estimation: While industrial settings are challenging because of the above-mentioned factors, they are also in some ways more structured and allow opportunities to exploit this structure. For example, 3D CAD models of most of the industrial parts are readily available. Even if some of them are not, the fact that most industrial assembly lines repeatedly handle a finite set of discrete parts many times (order of millions) makes CAD model acquisition cost-effective. The 3D CAD models provide a reliable source of information, potentially overcoming the challenging reflectance and environmental conditions.

1.2 A Practical Vision-Based Robotic Bin-Picking System

In this paper, we present a practical vision-based robotic bin-picking system that overcomes the challenges described in Section 1.1. The system performs detection and estimation of the 3D poses of objects that are stacked in a part container, picks up the parts from the container using an industrial robot arm, performs pose verification and refinement while the part is in the gripper, and inserts the picked part at a designated position. We have introduced two novel ideas that allow us to achieve reliable, fast, and accurate operation: (1) Novel imaging hardware that provides reliable geometric features regardless of the object's material and surface characteristics; (2) Fast, robust, and accurate 3D pose estimation based on the Fast Directional Chamfer Matching (FDCM) algorithm.

The fundamental challenges that arise due to non-Lambertian materials (e.g., metal, glass, ceramic), textureless parts (e.g., uniformly painted parts), and greasy and dirty environments lead to the fact that photometric features such as color and appearance descriptors are not robust enough in industrial settings. This motivates the need to develop features that are dependent on the geometry of the part rather than its photometry. We use an inexpensive camera design, the multi-flash camera (MFC) [RTF$^+$04], that provides reliable geometric features: depth edges. The location of the depth edges on an object are dependent only on the pose of the object with respect to the observer and the object geometry. Therefore, depth edges can be used to determine the pose of the object uniquely. In addition, these geometric descriptors allow easy and efficient incorporation of the 3D CAD models into our system. Given the 3D CAD model of an object, we retrieve the pose of the object that provides the best match between the observed features and the depth edges from the CAD model. This allows us to bypass a time-consuming training phase for each of the objects that would otherwise be necessary. A new part can be integrated into our system in less than 10 minutes.

Although many shape matching algorithms have been proposed over the decades, chamfer matching (CM) [BTBW77] remains among the fastest and most robust approaches in the presence of clutter. We adapt traditional chamfer matching with a host of techniques to improve reliability, accuracy, and speed. First, we exploit the geometric redundancies in the 3D structure of industrial parts by approximating the edge features using line segments. This, along with a 3D integral distance transform representation, allows us to both reduce the memory footprint and speed up the matching algorithm by orders of magnitude. Second, we incorporate a directional error term in the distance transform definition which significantly improves the reliability, robustness, and accuracy of matching. Further, we improve the pose estimation accuracy using a continuous optimization procedure. The result is a system that is capable of real-time operation for several industrial assembly applications.

While the primary goal of this research is to develop a robust and reliable vision system for industrial robotics, the FDCM algorithm also achieves state-of-the-art performance in shape matching. We present two additional application domains that benefit from FDCM: deformable object detection using a hand-drawn shape, and human pose estimation.

The paper is organized as follows. We briefly overview the related literature in Section 2. We present the shape matching algorithm and its optimization in Section 3. Pose estimation and the robotic bin-picking system are described in Section 4. We report on our extensive experimental validation of the proposed system and compare it to the state of the art in Section 5. The paper is concluded in Section 6. We note that this paper builds upon our previous work [LTV$^+$10, LTVC10].

2 Related Work

In recent decades, there has been a considerable amount of work on automating the process of part assembly using vision systems [SI73, NPK96, ASBR10]. Though vision systems are successful in identifying, inspecting, and locating parts in carefully engineered manufacturing settings, it remains a great challenge to extend their applicability to more general, unconstrained settings. Also, they often make use of simple geometric features such as lines, circles, or ellipses and their spatial organization [Vis]. Changing a target object in the assembly process would require significant manual adjustments or algorithmic modifications.

Model-based vision systems exploit 3D CAD models of objects, along with either acquired 2D images or range sensor data, for image interpretation. Such methods provide means for efficient detection, recognition, and pose estimation of objects in cluttered environments [Low87, Low91, SM92, JH99, BN10, DUNI10]. Methods such as [Low87, Low91, DD92] rely on establishing correspondences between 2D image features and points in the 3D models in order to obtain an initial estimate of object pose. The estimate is later refined using iterative algorithms. These correspondences are, however, difficult to obtain reliably.

Since establishing 3D-to-2D correspondences using images is a hard task, several systems rely either directly or indirectly on 3D information. Such methods greatly simplify the correspondence problem at the cost of increased hardware requirements. The most common approach is the use of 3D range sensors either based on structured light [SS03] or on time of flight [CSD$^+$10]. This provides the ability to establish 3D-to-3D point correspondences by matching 3D point descriptors from the CAD model to those in the acquired point cloud data. Several 3D point descriptors [SM92, JH99, BN10] have been proposed for matching the 3D scene points to the model points. To remove false matches, an interpretation tree procedure [GLP84] can be applied to find mutually consistent pairs. With these consistent pairs, one can use Horn's method [Hor87] to estimate the object's 3D pose. Unfortunately, these descriptors are less reliable for pose estimation of industrial parts because these objects are mostly made of planar surfaces, which leads to very few and uninformative features.

Recently, the use of the multi-flash camera [RTF$^+$04] for object pose estimation was proposed in [ASBR10]. The MFC, which was originally developed in the context of non-photorealistic rendering, provides depth edge features that can be used for pose estimation tasks. In this paper and in [LTV$^+$10], we significantly expand the scope and impact of MFC for industrial robotics. While [ASBR10] presented a system capable of handling isolated parts, here we present a system that can handle multitudes of parts randomly placed in a cluttered bin. Further, we also present a novel shape-matching algorithm that results in better accuracy and orders-of-magnitude improvement in matching speed, allowing real-time system performance.

One of the main technical contributions of this work is the development of the fast directional chamfer matching algorithm, which is widely applicable to several applications that currently use shape matching. Below, we briefly discuss related approaches in shape matching.

Shape matching has been an active area in robotic vision research. Several authors have proposed shape representations and similarity measures that aim to be invariant to object deformations [BMP02, LJ07]. These methods actively handle intra-class shape variations and achieve good performance in object recognition. However, they require a clean segmentation of the target object. This renders them less suitable for dealing with unstructured scenes due to the difficulty in foreground-background separation.

Recent studies focus on the recognition and localization of object shapes in cluttered images. In [BBM05], the shape matching problem is posed as finding the optimal correspondences between feature points, which leads to an integer quadratic programming problem. In [FTG06], a contour segment network framework is proposed in which shape matching is formulated as finding paths on the network that are similar to model outlines. In [FFJS08], Ferrari et al. propose a family of scale-invariant local shape descriptors (pair-of-adjacent-segment features) formed by k-connected nearly straight contour fragments in the edge map. These descriptors are later utilized in a shape matching framework [FJS10] through a voting scheme on a Hough space.

Zhu et al. [ZWWS08] formulate shape detection as a subset selection problem on a set of salient contours. Due to the NP-hardness of the selection problem, they compute an approximate solution using a two-stage linear programming procedure. In [FS07], a hierarchical object contour representation is proposed to model shape variation, and the matching is performed using dynamic programming. In [RJM08], a multi-stage approach is employed in which coarse detections, which are established by matching subsets of contour segments, are pruned by building the entire contour using dynamic programming.

These algorithms yield impressive results for matching shapes in cluttered images. However, they share a common drawback, high computational complexity, which makes them unsuitable for time-critical applications. Although proposed decades ago, chamfer matching [BTBW77] remains the preferred method when speed and accuracy are required, as discussed in [TSTC03]. In this paper, we propose an improved version of chamfer matching and demonstrate its superiority with respect to other variants [Gav98, SBC08]. Our approach improves the accuracy of chamfer matching while greatly reducing its time complexity, leading to a speedup of up to two orders of magnitude in several application scenarios.

3 Fast Directional Chamfer Matching

In this section, we introduce our *fast directional chamfer matching* algorithm, which we use for object detection and pose estimation in industrial robotics and other application areas.

3.1 Chamfer Matching

First we briefly explain standard chamfer matching (CM) [BTBW77], which is a popular technique for finding the best alignment between a template edge map and a query edge map. Let $U = \{\mathbf{u}_i\}$, where $i = 1, 2, ..., |U|$, be the set of edge pixels from a template edge map, and let $V = \{\mathbf{v}_j\}$, where $j = 1, 2, ..., |V|$, be the set of edge pixels from a query image edge map. The *chamfer distance* between U and V is defined as the average over all pixels $\mathbf{u}_i \in U$ of the distance between $\mathbf{u}_i$ and its nearest pixel in V:

$$d_{\mathrm{CM}}(U, V) = \frac{1}{n} \sum_{\mathbf{u}_i \in U} \min_{\mathbf{v}_j \in V} \|\mathbf{u}_i - \mathbf{v}_j\|. \tag{1}$$

where n is the number of template edge pixels, $n = |U|$.

Let W be a warping function defined on the image plane that is parameterized by $\mathbf{s}$. For instance, if W is a 2D Euclidean transformation, then $\mathbf{s} \in \mathrm{SE}(2)$ can be written as $\mathbf{s} = (\theta,\ \bar{t}_x,\ \bar{t}_y)$, where $\bar{t}_x$ and $\bar{t}_y$ are translations parallel to the x and y axes, respectively, and θ is the in-plane rotation angle. Its action on each image point $\mathbf{x} \in \mathbb{R}^2$ is given via the transformation

$$W(\mathbf{x}; \mathbf{s}) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \mathbf{x} + \begin{pmatrix} \bar{t}_x \\ \bar{t}_y \end{pmatrix}. \tag{2}$$

The best alignment parameter $\hat{\mathbf{s}}$ between the two edge maps is then given by

$$\hat{\mathbf{s}} = \arg \min_{\mathbf{s} \in \mathrm{SE}(2)} d_{\mathrm{CM}}(W(U; \mathbf{s}), V) \tag{3}$$

where $W(U; \mathbf{s}) = \{W(\mathbf{u}_i, \mathbf{s})\}$, $i = 1, 2, ..., |U|$.

The chamfer matching cost can be computed efficiently using the distance transform image

$$\mathrm{DT}_V(\mathbf{x}) = \min_{\mathbf{v}_j \in V} \|\mathbf{x} - \mathbf{v}_j\|, \tag{4}$$

which specifies the distance from each pixel $\mathbf{x}$ in the distance transform image to the nearest edge pixel in V. The distance transform can be computed in two passes over the image using dynamic programming [FH04]. Using the distance transform, the cost function (1) can be evaluated in linear time $O(n)$ via

$$d_{\mathrm{CM}}(U,V) = \frac{1}{n} \sum_{\mathbf{u}_i \in U} \mathrm{DT}_V(\mathbf{u}_i). \tag{5}$$

Chamfer matching provides a fairly smooth measure of fitness and can tolerate small rotations, misalignments, occlusions, and deformations. However, it becomes less reliable in the presence of background clutter due to an increase in the proportion of false correspondences. To improve its robustness, several variants of chamfer matching have been introduced that exploit edge orientation information. In [Gav98, DCS09], the template and query image edges are quantized into discrete orientation channels, and individual matching scores across channels are summed. Although these methods improve performance in cluttered scenes, the cost function is sensitive to the number of orientation channels and becomes discontinuous across channel boundaries. In [SBC08], the chamfer distance is augmented with an additional cost for orientation mismatch, which is given by the average difference in orientations between template edges and their nearest edge points in the query image. The method is known as oriented chamfer matching (OCM).

3.2 Directional Chamfer Matching

Instead of an explicit formulation of the orientation mismatch, we generalize the chamfer distance to points in $\mathbb{R}^3$ in order to match directional edge pixels. Each edge pixel $\mathbf{x}$ is augmented with a direction term, $\phi(\mathbf{x})$, and the directional chamfer matching (DCM) score is given by

$$d_{\mathrm{DCM}}(U,V) = \frac{1}{n} \sum_{\mathbf{u}_i \in U} \min_{\mathbf{v}_j \in V} \left(\|\mathbf{u}_i - \mathbf{v}_j\| + \lambda \|\phi(\mathbf{u}_i) - \phi(\mathbf{v}_j)\|_\pi \right) \tag{6}$$

where the parameter λ is a weighting factor between the location and orientation terms. To compute the direction terms, we fit line segments to the edge points (as explained in Section 3.3), and $\phi(\mathbf{x})$ is the orientation of the line segment associated with point $\mathbf{x}$. Note that the directions are written modulo π: $0 \leq \phi(\mathbf{x}) < \pi$, and the orientation error is defined as the minimum circular difference between the two directions:

$$\|\phi(\mathbf{x}_1) - \phi(\mathbf{x}_2)\|_\pi = \min\left\{ |\phi(\mathbf{x}_1) - \phi(\mathbf{x}_2)|, \big|\, |\phi(\mathbf{x}_1) - \phi(\mathbf{x}_2)| - \pi \big| \right\}. \tag{7}$$

In Figure 1, we illustrate the differences between DCM and OCM [SBC08]. The proposed matching cost, DCM, is a piecewise smooth function of both the

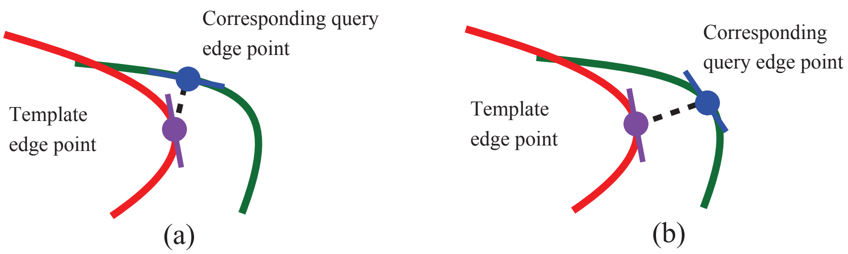
Corresponding query edge point
Template edge point
(a)
Corresponding query edge point
Template edge point
(b)

Figure 1: Matching costs for an edge point. (a) Oriented chamfer matching (OCM) [SBC08]. (b) Directional chamfer matching (DCM, proposed in this paper). Whereas in OCM the location error is augmented with the orientation difference from the nearest edge point, DCM jointly minimizes location and orientation errors.

translation (t_x, t_y) and the rotation (θ) of the template pose. It is more robust to clutter, missing edges, and small misalignments.

The computational complexity of existing chamfer matching algorithms is linear in the number of template edge points. Even though DCM includes an additional direction term, our algorithm (derived in this section) computes the *exact* DCM score with *sublinear* complexity.

3.3 Line-Based Representation

The edge map of a scene is not an unstructured binary pattern. On the contrary, the object contours comply with certain continuity constraints that can be retained by combining line segments of various lengths, orientations, and translations. Based on this observation, we represent an edge image as a collection of m line segments. Compared with a set of points which has cardinality n, its line-based representation is more concise. Encoding an edge map using the line-based representation requires only $O(m)$ memory size, where $m << n$, and is particularly suitable when the storage space for templates is limited. When the object exhibits a curved contour, more segments are required for good approximation, but the line-based representation is still more concise than the set of edge pixels.

We use a variant of the RANSAC [FB81] algorithm to compute the line-based representation of an edge map. The outline of the algorithm is as follows. The algorithm initially hypothesizes a variety of line segments by selecting a small subset of edge points and their directions. The support of each line segment is given by the set of points that satisfy the line's equation up to a small residual, $\nu \geq 0$, and form a continuous structure. The line segment with the largest support is retained, and its supporting points are removed from the set of edge points. The

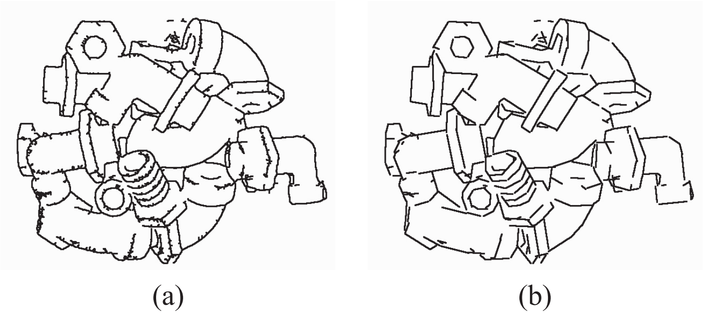

Figure 2: Line-based representation. (a) Edge image. The image contains 11,542 edge points. (b) Line-based representation of the edge image. The image contains 300 line segments.

procedure is repeated with the reduced set of edge points, until the support of the longest line candidate becomes smaller than a few points.

The algorithm only retains edge points with continuity and sufficient support; therefore, the noise and isolated edges are filtered out. In addition, the directions recovered through the line fitting procedure are more precise than would be obtained using local operators such as image gradients. An example of the line-based representation is given in Figure 2, where a set of 11,542 points is modeled with 300 line segments.

3.4 Three-Dimensional Distance Transform

The matching score given in (6) requires finding the minimum matching cost over location and orientation terms for each template edge point. Therefore, the computational complexity of the brute-force algorithm is quadratic in the number of template and query image edge points. Here we present a three-dimensional distance transform representation ($\mathrm{DT3}_V$) for computing the matching cost in linear time. A similar structure was also used in [OH97] for fast evaluation of Hausdorff distances.

This representation is a three dimensional image tensor in which the first two dimensions are the locations in the image plane and the third dimension belongs to a discrete set of edge orientations. We evenly quantize the edge orientation into q discrete channels, $\hat{\Phi} = \{\hat{\phi}_i\}, i = 1, 2, ..., q$, which evenly divide the range $[0 \quad \pi)$. Each element of the tensor encodes the minimum distance to an edge point in the joint location and orientation space:

$$\mathrm{DT3}_V(\mathbf{x}, \phi(\mathbf{x})) = \min_{\mathbf{v}_j \in V} \left(\|\mathbf{x} - \mathbf{v}_j\| + \lambda \|\hat{\phi}(\mathbf{x}) - \hat{\phi}(\mathbf{v}_j)\|_\pi \right), \tag{8}$$

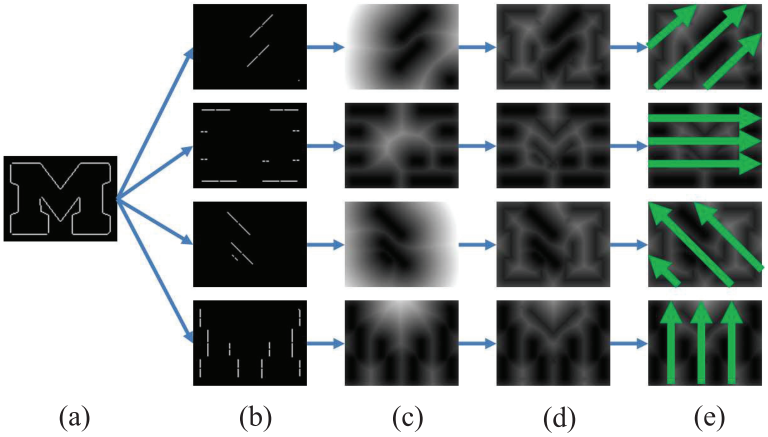

Figure 3: Computation of the integral distance transform tensor. (a) The set V of points in the query edge map is mapped into a set of line segments through a line-fitting procedure. (b) Edges are quantized into discrete orientation channels. (c) Two dimensional distance transform of each orientation channel. (d) The three-dimensional distance transform, $\mathrm{DT3}_V$, is updated based on the orientation cost. (e) The 3D distance transform is integrated along the discrete edge orientations, and the integral distance transform tensor, $\mathrm{IDT3}_V$, is computed.

where $\hat{\phi}(\mathbf{x})$ is the nearest quantization level in the orientation space $\hat{\Phi}$ to the edge orientation $\phi(\mathbf{x})$.

We present an algorithm to compute the $\mathrm{DT3}_V$ tensor in $O(q)$ passes over the image by solving two dynamic programs consecutively. Equation (8) can be rewritten as

$$\mathrm{DT3}_V(\mathbf{x}, \phi(\mathbf{x})) = \min_{\hat{\phi}_i \in \hat{\Phi}} \left(\mathrm{DT}_{V\{\hat{\phi}_i\}} + \lambda \|\hat{\phi}(\mathbf{x}) - \hat{\phi}_i\|_\pi \right) \tag{9}$$

where $\mathrm{DT}_{V\{\hat{\phi}_i\}}$ is the two dimensional distance transform of the edge points in V that have edge orientation $\hat{\phi}_i$.

Initially, we compute q two-dimensional distance transforms $\mathrm{DT}_{V\{\hat{\phi}_i\}}$, which requires $O(q)$ passes over the image using the standard distance transform algorithm [FH04]. Subsequently, the $\mathrm{DT3}_V$ tensor (9) is computed by using a second dynamic program for each image pixel separately. The tensor is initialized with the two dimensional distance transforms, $\mathrm{DT3}_V(\mathbf{x}, \hat{\phi}_i) = \mathrm{DT}_{V\{\hat{\phi}_i\}}(\mathbf{x})$, and is updated with a forward recursion

$$\mathrm{DT3}_V(\mathbf{x}, \hat{\phi}_i) = \min\{\mathrm{DT3}_V(\mathbf{x}, \hat{\phi}_i), \mathrm{DT3}_V(\mathbf{x}, \hat{\phi}_{i-1}) + \lambda \|\hat{\phi}_{i-1} - \hat{\phi}_i\|_\pi\} \tag{10}$$

and a backward recursion

$$\mathrm{DT3}_V(\mathbf{x}, \hat{\phi}_i) = \min\{\mathrm{DT3}_V(\mathbf{x}, \hat{\phi}_i), \mathrm{DT3}_V(\mathbf{x}, \hat{\phi}_{i+1}) + \lambda \|\hat{\phi}_{i+1} - \hat{\phi}_i\|_\pi\} \quad (11)$$

for each pixel $\mathbf{x}$. Unlike in the standard distance transform algorithm, special handling is required for the circular orientation. The forward and backward recursions do not terminate after a full cycle, $i = 1, \ldots, q$ or $i = q, \ldots, 1$ respectively, but the values of the tensor entries continue to be updated in a circular form until the value for a tensor entry is not changed. Note that at most 1.5 cycles are needed for each of the forward and backward recursions, therefore the worst time computational cost is $O(q)$ passes over the image. We illustrate the computation of the three dimensional distance transform in Figure 3(a)–(d). Using $\mathrm{DT3}_V$, the directional chamfer matching score of any template U can be computed as

$$d_{\mathrm{DCM}}(U, V) = \frac{1}{n} \sum_{\mathbf{u}_i \in U} \mathrm{DT3}_V(\mathbf{u}_i, \hat{\phi}(\mathbf{u}_i)), \quad (12)$$

where the complexity is linear in n, the number of edge points in U.

3.5 Integral Distance Transform Tensor

Let $l_{[\mathbf{x}_1, \mathbf{x}_2]}$ represent the line segment in the image plane connecting pixels $\mathbf{x}_1$ and $\mathbf{x}_2$. Let $L_U = \{l_{[\mathbf{s}_j, \mathbf{e}_j]}\}$, $j = 1, \ldots, m$, be the line-based representation of template edge points U, where $\mathbf{s}_j$ and $\mathbf{e}_j$ are the start and end locations of the jth line segment respectively. For ease of notation, we sometimes refer to a line segment with only its index, $l_j = l_{[\mathbf{s}_j, \mathbf{e}_j]}$. We assume that the line segment directions are restricted to q discrete channels $\hat{\Phi}$, which is enforced in the line-based representation. We choose the number of directions q large enough (in our experiments, $q = 60$) to avoid quantization artifacts. The line-based representation of Figure 2(b) is generated from the edge image in Figure 2(a) using $q = 60$ directions.

Since the edge points in a line segment all have the same orientation, which is the direction of the line segment $\hat{\phi}(l_j)$, the directional chamfer matching score (12) can be rearranged as

$$d_{\mathrm{DCM}}(U, V) = \frac{1}{n} \sum_{l_j \in L_U} \sum_{\mathbf{u}_i \in l_j} \mathrm{DT3}_V(\mathbf{u}_i, \hat{\phi}(l_j)). \quad (13)$$

In this formulation, the kth orientation channel of the $\mathrm{DT3}_V$ tensor, $\mathrm{DT3}_V(\mathbf{x}, \hat{\phi}_k)$, is only used for evaluating the matching scores of the line segments having the direction $\hat{\phi}_k$, which is achieved by summing over the points in the line segments.

Integral images are intermediate image representations used for fast calculation of region sums [VJ01] and linear sums [BB09]. Here, we present an integral distance transform representation ($\mathrm{IDT3}_V$) to evaluate the summation of costs over any line segment in $O(1)$ operations, as shown in Figure 3(e).

Let $\mathbf{x}_0$ be the intersection of an image boundary with the line that passes through $\mathbf{x}$ and has direction $\hat{\phi}_i$. Each entry of the $\mathrm{IDT3}_V$ tensor is given by

$$\mathrm{IDT3}_V(\mathbf{x}, \hat{\phi}_i) = \sum_{\mathbf{x}_j \in l_{[\mathbf{x}_0, \mathbf{x}]}} \mathrm{DT3}_V(\mathbf{x}_j, \hat{\phi}_i). \tag{14}$$

The $\mathrm{IDT3}_V$ tensor can be computed in one pass over the $\mathrm{DT3}_V$ tensor. Using this representation, the directional chamfer matching score of any template U can be computed in $O(m)$ operations via

$$d_{\mathrm{DCM}}(U, V) = \frac{1}{n} \sum_{l_{[\mathbf{s}_j, \mathbf{e}_j]} \in L_U} \mathrm{IDT3}_V(\mathbf{e}_j, \hat{\phi}(l_{[\mathbf{s}_j, \mathbf{e}_j]})) - \mathrm{IDT3}_V(\mathbf{s}_j, \hat{\phi}(l_{[\mathbf{s}_j, \mathbf{e}_j]})). \tag{15}$$

3.6 Search Optimization

In this section we present two search optimization techniques based on the bounds on the matching cost and empirically show that the number of evaluated line segments is *sublinear* in the number of template points n.

3.6.1 Bound in the Hypotheses Domain

The $O(m)$ complexity is only an upper bound on the number of computations. FDCM can be used for object detection and for localization. For object detection, we only need to retain the hypotheses for which the template matching cost is less than a detection threshold. For localization, we only need to retrieve the hypothesis with the lowest matching cost.

We order the template line segments with respect to their lengths and start the summation (15) from the longest line segment. A hypothesis is eliminated during the summation if the cost is larger than the detection threshold or the current best hypothesis. Since the lengths of the line segments roughly decay exponentially, for most of the hypotheses only a few arithmetic operations are performed.

3.6.2 Bound in the Spatial Domain

The DCM cost function (6) is smooth and bounded in the spatial domain. We utilize this fact to significantly reduce the number of hypotheses evaluated. Let $\boldsymbol{\delta} \in \mathbb{R}^2$

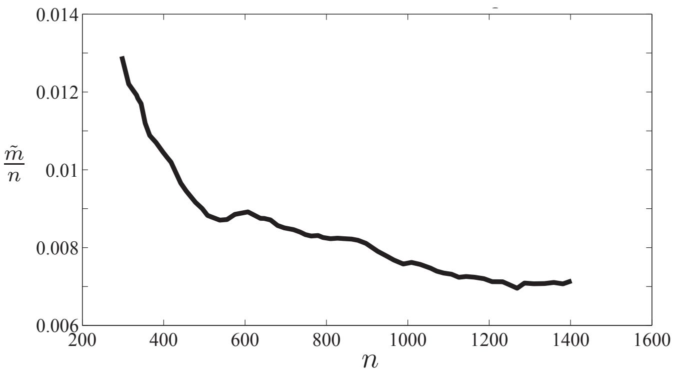

Figure 4: Empirical evidence of sublinear time complexity in the number of template points. The graph plots the ratio of the number of evaluated lines $\tilde{m}$ to the number of template points n vs. the number of template points. (If the complexity were linear in n, the graph would be a horizontal line.)

be a translation of the model U in the image plane. The DCM cost variation due to translation is given by

$$\begin{aligned} d_{\mathrm{DCM}}(U+\boldsymbol{\delta},V) &= \frac{1}{n}\sum_{\mathbf{u}_i\in U}\min_{\mathbf{v}_j\in V}\|\mathbf{u}_i+\boldsymbol{\delta}-\mathbf{v}_j\|+\lambda\|\phi(\mathbf{u}_i)-\phi(\mathbf{v}_j)\|_\pi \\ &\leq \frac{1}{n}\sum_{\mathbf{u}_i\in U}\min_{\mathbf{v}_j\in V}\|\mathbf{u}_i-\mathbf{v}_j\|+\|\boldsymbol{\delta}\|+\lambda\|\phi(\mathbf{u}_i)-\phi(\mathbf{v}_j)\|_\pi = \|\boldsymbol{\delta}\|+d_{\mathrm{DCM}}(U,V). \end{aligned} \tag{16}$$

From Equation (16), the variation of the cost is bounded by the spatial translation $|d_{\mathrm{DCM}}(U+\boldsymbol{\delta},V)-d_{\mathrm{DCM}}(U,V)| \leq \|\boldsymbol{\delta}\|$. If the detection threshold is τ and the cost of the current hypothesis is $\psi > \tau$, then there can not be a target within the $\|\boldsymbol{\delta}\| = |\psi-\tau|$ pixel range that has a matching cost lower than the detection threshold. Therefore, we can skip the evaluation of the hypotheses within this region.

3.6.3 Empirical Evidence of Sublinear Complexity

It is easy to see that the sublinear complexity holds in the case of scaling the template shape. As the number of edge points, n, increases with the template size, the cardinality m of the line-based representation of the template remains the same. Hence, the same number of arithmetic operations is required to compute the matching cost, which means the matching complexity is constant irrespective of the number of edge points.

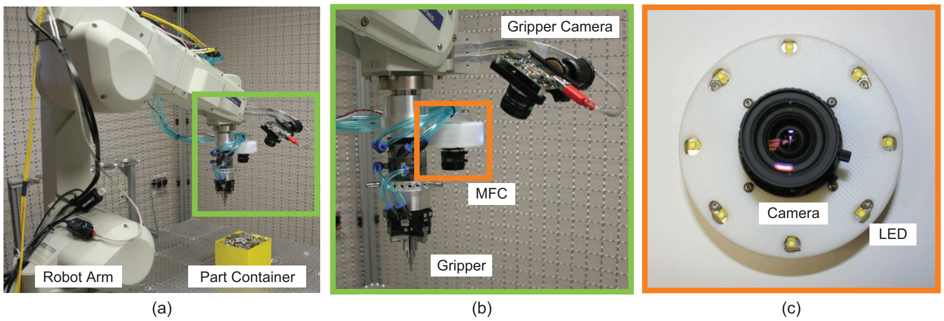

Figure 5: Our robotic grasping system. A multi-flash camera (MFC), shown in detail in (c), is mounted on the robot arm and used to perform detection and pose estimation of objects (parts) placed in a container. The gripper camera is a standard camera that is mounted above the robot's wrist joint and pointed at the tip of the gripper. The gripper camera is used to perform error detection after an object is picked up from the container.

We also provide empirical evidence that in a more general setup, as well, the matching complexity is a sublinear function of the number of template points. As explained above, the $O(m)$ complexity is only an upper bound on the number of evaluations, and on average we need to evaluate only a fraction of the m lines. Empirically, we evaluate the $\tilde{m}$ longest lines, where $\tilde{m}$ is chosen to fit $20\% - 30\%$ of the template points. Most of the energy is concentrated in only a few lines, and we find that $\tilde{m}$ grows sublinearly with n. In Figure 4, we plot the number of template points, n, on the x-axis and the ratio of the number of evaluated lines to the number of template points, $\frac{\tilde{m}}{n}$, on the y-axis. For this graph, $\tilde{m}$ is selected as the number of lines that fit 30% of the template points. The curve is generated using 1,000 shape images from the MPEG-7 dataset. We observe that as the number of template points increases, the fraction of evaluated lines decreases, which provides empirical evidence that the algorithm is sublinear in the number of template points $\big(< O(n) \big)$.

4 Pose Estimation for Robotic Bin Picking

In this section, we present our robotic bin-picking system that uses the shape matching algorithm described in Section 3.

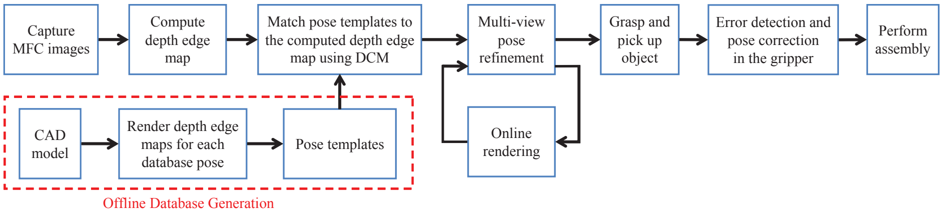
Capture MFC images
Compute depth edge map
Match pose templates to the computed depth edge map using DCM
Multi-view pose refinement
Grasp and pick up object
Error detection and pose correction in the gripper
Perform assembly
Online rendering
CAD model
Render depth edge maps for each database pose
Pose templates
Offline Database Generation

Figure 6: Flowchart of our system.

4.1 System Overview

Figure 5 shows our system setup. We mount an MFC and a standard camera on an industrial robot arm. The MFC is used to perform object detection and pose estimation of objects that are randomly arranged in a part container. The robot arm uses the estimated pose of the object to grasp the object and lift it out of the container. The standard camera, which we call the gripper camera, is focused on the tip of the gripper and is used to perform error detection after the object is picked up. Both cameras are calibrated offline using a checkerboard. The calibration determines internal parameters of the cameras as well as the poses of the cameras with respect to the robot coordinate system (hand-eye calibration).

The flowchart in Figure 6 provides a summary of our system. We give an overview of our algorithm below and explain the details of each process in the following subsections.

1. **Offline database generation** (Section 4.3): For each object, we render the 3D CAD model according to a set of hypothesized poses, extract depth edges, and compute the line-based representation (which was presented in Section 3.3) of the depth edges.

2. **MFC imaging and depth edge extraction** (Section 4.2): We capture 9 images, using the 8 different flashes of the MFC and one image without any flash. The depth edges in the scene are computed using these images.

3. **Object detection and pose estimation**: Using the FDCM algorithm (which was presented in Section 3), we retrieve the database pose and its in-plane transformation parameters that have the minimum matching cost and use these as a coarse pose estimate. The matching algorithm is accelerated using a heuristic that we call *one-dimensional search* (Section 4.4). Further improvement of the coarse estimate is achieved via a multi-view pose refinement algorithm (Section 4.5).

4. **Grasping and picking up the object**: We use the estimated 3D pose to grasp the object with the gripper and lift it out of the part container.

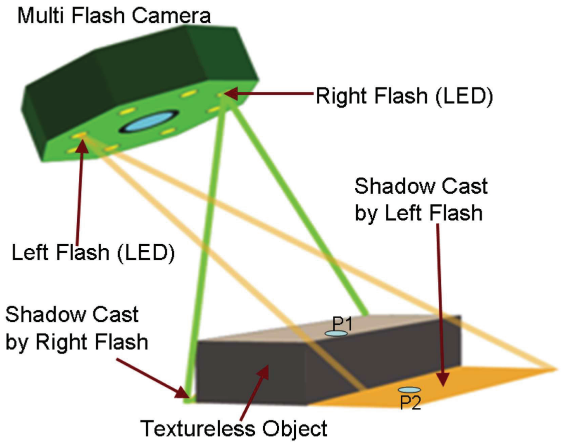

Figure 7: Illustration of the principle of multi-flash camera (MFC) imaging. The scene is illuminated by flashing one LED at a time. Due to the different positions of the LEDs, the shadows cast by the object change. While intensity values of points such as $P1$ (on the top surface of the object) remain nearly constant when illuminated by different LEDs, intensity values of points such as $P2$ (which is in shadow for some of the LEDs) change. This property is exploited to detect depth edges.

5. **Error detection and pose correction** (Section 4.6): We use the gripper camera to detect grasping errors. We evaluate/re-estimate the pose of the object in the gripper. The pose of the object is corrected if necessary.

6. **Assembly**: The pose-corrected object is ready for the next step of the assembly task.

4.2 MFC Imaging and Depth Edge Extraction

We use an MFC [RTF$^+$04] to detect depth edges (depth discontinuities) in the scene. A depth edge is a robust geometric feature. It is invariant to the surface properties of objects (textured, textureless, shiny, etc.) and is unaffected by oil, grime, or dirt on the object surface, which are common in industrial environments. The MFC is equipped with 8 point light sources made of light-emitting diodes (LEDs). They are evenly distributed around the camera in a circular fashion, as shown in Figure 7. During the MFC imaging, these LEDs are sequentially switched on to illuminate the scene. Only one LED is switched on at a time, and an image is taken. This is repeated for each of the 8 LEDs. We also take an image with all the LEDs turned off to record the ambient illumination. The different LED positions

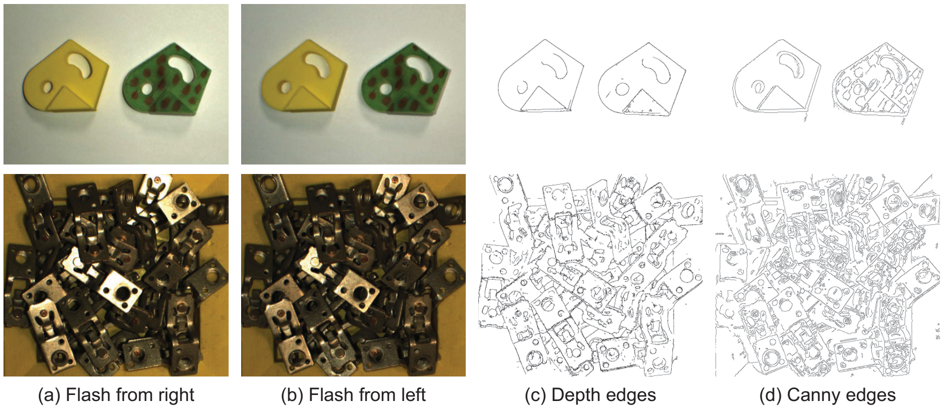

Figure 8: Comparison between depth edges extracted using an MFC and standard Canny edges for a simple scene (top) and a highly cluttered scene (bottom). (a, b) Two out of eight flash images captured with an MFC. Note the different shadow locations. (c) Extracted depth edges. (d) Standard intensity edges computed by using a Canny edge detector on an image captured without flash. Note that the Canny edge results include both texture and depth edges as well as are affected by shadows due to ambient lights.

result in different illumination directions, so the positions of shadows vary across the 8 images. This property can be exploited to detect the depth edges in the scene, as discussed below.

Let I_i denote the image illuminated by the ith LED, after subtracting the ambient image. First, we construct the maximum image, $I_{\max}$, where the cast shadows due to the flashes are removed. We consider each pixel location and find the maximum intensity value at that location across the 8 images:

$$I_{\max}(x, y) = \max_i I_i(x, y). \tag{17}$$

Next, we compute the ratio images

$$RI_i = \frac{I_i}{I_{max}}. \tag{18}$$

Ideally, if a pixel is in a shadow region of image I_i (e.g., point $P2$ in Figure 7), this ratio should be 0 since the contribution of the illumination from the ambient source has been removed. In contrast, the ratio in other non-shadow regions (e.g., point $P1$ in Figure 7) should be close to 1 since these regions are illuminated by all the flashes. Notice that a depth edge corresponds to a point of transition from a non-shadow region to a shadow region along the illumination direction defined

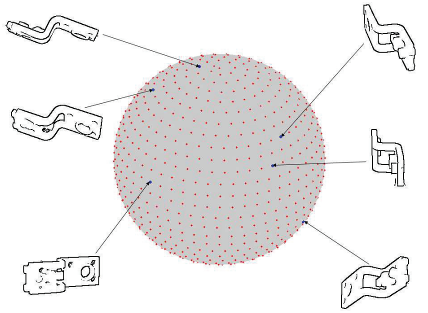

Figure 9: Database generation. We uniformly sample the rotation angles (θ_x and θ_y) on the 2-sphere. The template database is generated by rendering the CAD model of the object at each of the sampled rotations.

by the LED position in each image. Therefore, for each ratio image, we detect this transition by using a Sobel filter whose direction is aligned with the illumination direction, followed by non-maximum suppression. We then add the filter responses across different flash images and use hysteresis thresholding similar to the Canny edge detector [Can86] to obtain a depth edge image.

Comparison with Intensity Edges: Figure 8 compares depth edges extracted using an MFC to standard intensity edges, which were computed by using a Canny edge detector on an image captured without flash. Note that the Canny edge results include texture edges (e.g., the artificially painted object surface in the top row, and small scratches on the surface of the shiny objects in the bottom row). They are also affected by shadows due to ambient light (note the difference of detected edge locations between the MFC depth edge results and the Canny edge results). In contrast, our approach using MFC imaging provides depth edges only, which can be used as robust geometric features for object detection and pose estimation.

4.3 Database Generation

An object exhibits different silhouettes in different poses. Although the matching algorithm in Section 3 models in-plane rotation and translation, it does not model rotations in depth, which can change an object's depth-edge silhouette. To accommodate these variations, we generate a set of templates across the range of possible rotations in depth, denoting this set of templates by $\{U_k\}$. The search problem in (3) is generalized to find the best-matching template in this set, as follows:

$$\arg \min_{k, \mathbf{s} \in \mathrm{SE}(2)} d_{DCM}(W(U_k; \mathbf{s}), V). \tag{19}$$

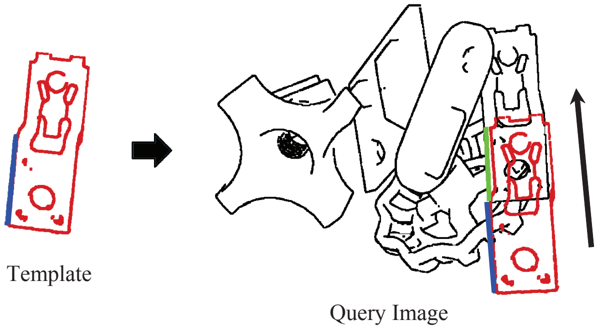

Figure 10: One-dimensional search. A template is rotated and translated such that one template line segment (the blue line segment) is aligned with one line segment in the query image (the green line segment). The template is translated along the query line segment, and the directional chamfer matching cost is evaluated for each translation.

Given a CAD model of the object, we generate a database of depth-edge templates by detecting the depth discontinuities in the model. In this simulation, a virtual camera with the same internal parameters as the real camera is placed at the origin, and its optical axis is aligned with the z-axis of the world coordinate system. The CAD model of the object is then placed on the z-axis at a distance t_z from the virtual camera, which is equal to the actual distance of the part container from the real MFC in our setup. The virtual flashes are switched on one at a time, and eight renderings of the object (including cast shadows) are acquired. The depth edges are detected using the procedure described in Section 4.2.

An arbitrary 3D rotation can be decomposed into a sequence of three elemental rotations about three orthogonal axes. We align the first of these axes to the camera optical axis and refer to the rotation about this axis as *in-plane rotation* (θ_z). The other two axes are on a plane perpendicular to the camera optical axis, and we call the rotation about these two axes *out-of-plane rotation* (θ_x and θ_y). Note that an in-plane rotation simply results in an in-plane rotation of the observed images, whereas the effect of an out-of-plane rotation depends on the 3D structure of the object. Due to this distinction, we only include out-of-plane rotations of the object in the database. We sample K out-of-plane rotations (θ_x and θ_y) uniformly on the 2-sphere, S^2, as shown in Figure 9, and generate the depth-edge template U_k for each rotation $k \in \{1, \ldots, K\}$.

4.4 One-dimensional Search

In order to retrieve the coarse pose of the object in the scene using (19), we sequentially search over all the database templates, U_k, where $k = 1, \ldots, K$. For each template U_k, searching for the best alignment $\hat{\mathbf{s}} = (\theta_z, \bar{t}_x, \bar{t}_y)$ is computationally intensive (three degrees of freedom). Here we present a heuristic method to greatly reduce the search space (from three degrees of freedom to one degree of freedom), which exploits the fact that under the best alignment, the template and query image line segments are well aligned. Additionally, the major lines of the template and the query images are reliably detected during the line-fitting process, since the algorithm favors line segments with larger support.

We order the sets of template and query line segments from longest to shortest and use a few major lines from the ordered sets of template and query line segments to guide the hypothesis search. The template is initially rotated and translated such that a template line segment is aligned with the direction of a query image line segment and the end point of the template line is translated to match the start point of the query line segment, as illustrated in Figure 10. The template is then translated along the query line segment direction, and the cost function is evaluated only at locations where there is an overlap between the two segments. This procedure reduces the three-dimensional search (in-plane rotation and translation) to one-dimensional search along only a few directions. The search time is invariant to the size of the image and is only a function of the number of templates and query image lines and their lengths. With this heuristic, we can efficiently find the minimum-cost template and its alignment parameters.

4.5 Multi-View Pose Refinement

The minimum-cost template, together with its in-plane transformation parameters $(\theta_z,\ \bar{t}_x,\ \bar{t}_y)$, provide a coarse estimate of the 3D object pose. Let θ_x, θ_y be the out-of-plane rotation angles, and let t_z be the distance from the camera, which are used to render the template. We back-project the in-plane translation parameters to 3D using the camera calibration matrix $\mathbf{K}$, and obtain the initial 3D pose of the object, $\mathbf{p}^0$, as the three Euler angles $(\theta_x, \theta_y, \theta_z)$ and a 3D translation vector $(t_x, t_y, t_z)^T$. The 3D pose $\mathbf{p}$ can also be written in matrix form

$$\mathbf{M_p} = \begin{pmatrix} \mathbf{R_p} & \mathbf{t_p} \\ \mathbf{0} & 1 \end{pmatrix} \in \mathrm{SE}(3), \tag{20}$$

where $\mathbf{R_p}$ is the 3×3 rotation matrix computed by a sequence of three rotations around the x–y–z axes, $\mathbf{R}_{\theta_z}\mathbf{R}_{\theta_y}\mathbf{R}_{\theta_x}$, and $\mathbf{t_p}$ is the 3D translation vector.

The precision of the initial pose estimation is limited by the discrete set of out-of-plane rotations included in the database. Below, we present a continuous optimization method to refine the pose estimate. The proposed method is a combination of the iterative closest point (ICP) [Zha94] and Gauss-Newton [BV04] optimization algorithms. It can work with any number (one or more) of views with known camera poses.

Refinement Algorithm: Let $\mathbf{M}^{(j)} \in \mathrm{SE}(3)$ be the 3D rigid transformation matrix representing the pose of the camera corresponding to the jth view in the world coordinate system, and let $\mathbf{P} = (\mathbf{K} \;\; \mathbf{0})$ be the 3×4 projection matrix. As explained in Section 4.1, the projection matrix is known through camera calibration, and the camera poses are known through hand-eye calibration and the motion of the robot. The edge points detected in the jth view are given by the set $V^{(j)} = \{\mathbf{v}_i^{(j)}\}$.

First, we establish a set of correspondences between the 3D CAD model points $\tilde{\mathbf{u}}_i^{(j)}$ and the 2D detected edge points $\mathbf{v}_i^{(j)}$. We find these 3D-to-2D point correspondences via closest-point assignment on the image plane. To do so, we simulate the multi-camera setup and render the 3D CAD model with respect to the current pose estimate $\mathbf{p}$. Let $U^{(j)} = \{\mathbf{u}_i^{(j)}\}$ be the sets of detected edge points in the jth synthetic view and $\tilde{U}^{(j)} = \{\tilde{\mathbf{u}}_i^{(j)}\}$ be the corresponding 3D CAD model points in the jth camera coordinate system. For each point $\mathbf{u}_i^{(j)} \in U^{(j)}$, we search for the nearest point in $V^{(j)}$ with respect to the directional chamfer matching cost as

$$\arg \min_{\mathbf{v}_k^{(j)} \in V^{(j)}} \left\| \mathbf{u}_i^{(j)} - \mathbf{v}_k^{(j)} \right\| + \lambda \left\| \phi\big(\mathbf{u}_i^{(j)}\big) - \phi\big(\mathbf{v}_k^{(j)}\big) \right\|_{\pi} \qquad (21)$$

and establish 3D-to-2D point correspondences $\big(\tilde{\mathbf{u}}_i^{(j)}, \mathbf{v}_i^{(j)}\big)$.

Using the found correspondences, our optimization algorithm minimizes the sum of squared projection errors simultaneously in all the views:

$$\varepsilon(\mathbf{p}) = \sum_{j} \sum_{\tilde{\mathbf{u}}_i^{(j)}} \left\| \mathbf{P}\mathbf{M}^{(j)}\mathbf{M}_{\mathbf{p}}{\mathbf{M}^{(j)}}^{-1}\tilde{\mathbf{u}}_i^{(j)} - \mathbf{v}_i^{(j)} \right\|^2. \qquad (22)$$

Note that both the 3D points $\tilde{\mathbf{u}}_i^{(j)}$ and their projections are expressed in homogeneous coordinates, while the corresponding edge points are expressed in Cartesian image coordinates. With a slight abuse of notation, in this formulation, we assume that the projections of the 3D points have been converted to 2D image coordinates before measuring the distances.

The nonlinear least squares error function given in (22) is minimized using the Gauss-Newton algorithm. Starting with the initial pose estimate $\mathbf{p}^0$, we improve the estimation via the iterations

$$\mathbf{p}^{t+1} = \mathbf{p}^t + \Delta\mathbf{p}. \qquad (23)$$

The update vector $\Delta\mathbf{p}$ is given by the solution of the normal equations

$$(\mathbf{J}_{\boldsymbol{\varepsilon}}^T \mathbf{J}_{\boldsymbol{\varepsilon}})\Delta\mathbf{p} = -\mathbf{J}_{\boldsymbol{\varepsilon}}^T \boldsymbol{\varepsilon}, \tag{24}$$

where $\boldsymbol{\varepsilon} \in \mathbb{R}^N$ is the residual vector comprising the N summed error terms in (22), and $\mathbf{J}_{\boldsymbol{\varepsilon}}$ is the $N \times 6$ Jacobian matrix of $\boldsymbol{\varepsilon}$ with respect to $\mathbf{p}$, evaluated at $\mathbf{p}^t$. Similar to the ICP algorithm, the correspondence and minimization problems are solved iteratively until convergence.

Implementation for Bin Picking: The pose refinement method could be used with a single view to refine the coarse pose estimate. However, we found that the estimation accuracy obtained using a single view is not enough for accurate grasping. To increase the accuracy, we use a two-view approach. We move the robot arm to a second location and capture the scene again using the MFC. The second location is determined depending on the coarse pose estimate of a detected object, such that in the second view the object is captured at the center of the image and from a different out-of-plane rotation angle.

Since we perform the coarse pose estimation on the first view, typically the projection errors in the second view are larger than those in the first view. This is particularly the case when the distance between the camera and the object is very different from the hypothesized distance t_z that was used to generate the database (Section 4.3). To improve convergence, we first perform the refinement using the first view only, for several iterations, and then jointly using both views. In general, we found that 20 iterations suffice for convergence.

4.6 Error Detection and Pose Correction in the Gripper

The estimated 3D pose of the object is used to grasp the object using the gripper and lift it out of the part container. The grasping will fail if the estimated pose is inaccurate. Moreover, even if the estimated pose is correct, the grasping process may introduce errors because of slippage and interference from the other objects. These can result in a grasping failure (object is not picked up) or in the object having the wrong pose in the gripper, which would make it impossible to perform subsequent assembly tasks. Therefore, after grasping, we use the gripper camera to detect these errors and correct the object pose before the next stage in the assembly.

The goal of this error detection and pose correction process is to determine whether the object is grasped with the correct pose. We use a standard camera as the gripper camera and mount it above the wrist joint of the robot arm, as shown in Figure 5. The gripper camera is focused on the tip of the gripper and captures an image of the object after it is lifted out the part container. We use the Canny edge detector to extract edges from the image acquired by the gripper camera. The extracted edges include both texture and depth edges, which are not as robust as the depth edges extracted using the MFC. However, since the object is already isolated in the gripper, we find that these edges work well for error detection.

Since we know the ideal pose of the object in the gripper (the pose that would occur if there were no error in the initial pose estimation and gripping process), we use this ideal pose as the initial guess and apply the pose refinement algorithm described in Section 4.5. If the refined pose is very different from the ideal pose or the matching cost becomes larger than a threshold, then we detect it as an error and drop the object back into the part container. Otherwise, we use the refined pose for the subsequent assembly task. Since the gripper camera is located above the robot's wrist joint, we can obtain a second view of the object in the gripper with a different pose by rotating the wrist and capturing another image using the gripper camera. In the experiments (see Section 5.1.4), we show that single-view pose estimation is sufficient for detecting errors, but for pose correction, two-view pose estimation is preferable due to the higher accuracy required.

Foreground Extraction: We exploit robot motion to make the pose estimation in the gripper more accurate and efficient. The idea is to move the robot arm during the exposure time of the camera, while keeping the relative pose between the camera and the gripper fixed. This can be achieved by fixing the joints of the robot that are between the gripper and the arm segment to which the camera is attached (not moving the wrist joint) and moving the other joints. This robot motion introduces blur only in the background while keeping the foreground object sharp. As shown in Figure 11, images captured during such a robot motion produce sharp edges only on the foreground object (which is stationary relative to the camera), leading to accurate and efficient pose estimation. We call this *foreground extraction*, because it is essentially the reverse of standard background subtraction.

5 Experiments

We conducted extensive evaluations of the proposed algorithm for several applications using challenging real and synthetic datasets. In this section, we first demonstrate results for the robotic bin-picking system described in Section 4 and then present results of the proposed shape matching algorithm (described in Section 3) for other applications: deformable object detection using a hand-drawn shape, and human pose estimation.

Note that in all of our experiments, we emphasize our FDCM algorithm's improvement in accuracy and speed compared to CM and OCM. For deformable object detection and human pose estimation, the performance of FDCM is roughly comparable to state-of-the-art methods, and if desired the FDCM estimates could be further refined by using them as initial hypotheses for more computationally expensive point registration algorithms.

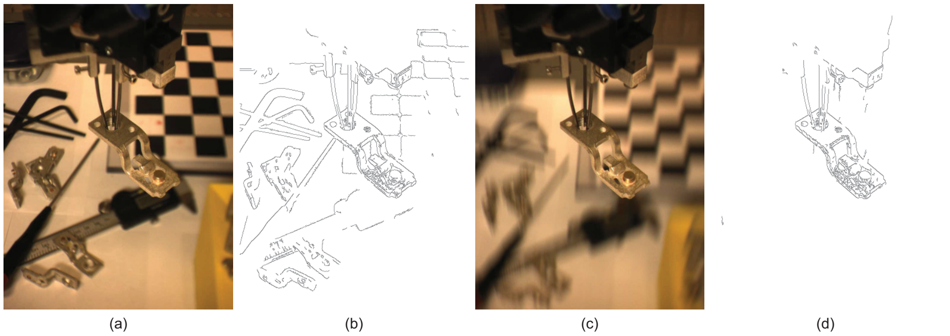

Figure 11: Foreground extraction. Images captured by the gripper camera (a) while the robot arm is fixed and (c) during a robot arm motion in which the relative pose between the object and the gripper camera is fixed. Note that in (c), the background is blurred due to the motion, while the foreground object remains sharp. Corresponding Canny edge detection results using the same threshold are shown in (b) and (d).

In all our experiments, we used $q = 60$ orientation channels. We set the weighting factor $\lambda = \frac{180}{6 \cdot \pi}$, which means that a 6° error in line orientation carries the same penalty as a 1-pixel distance in image plane.

5.1 Pose Estimation for Robotic Bin Picking

5.1.1 Synthetic Examples

We quantitatively evaluated the accuracy of the proposed matching algorithm to detect and localize objects in highly cluttered scenes on an extensive synthetic dataset. The synthetic dataset was generated using 3D models of 6 objects, with 3D shapes of varying complexity, which were placed randomly one over the other to generate several cluttered scenes. We computed depth-edge images by simulating the MFC and its cast shadows in software using OpenGL. The average occlusion of each part in the dataset was 15%, while the maximum occlusion was 25%. Moreover, in order to simulate missing depth edges and other imperfections in MFC imaging, a small fraction (about 10–15%) of the depth edges were removed. Furthermore, the depth-edge images were corrupted with significant noise by adding uniformly sampled line segments. There were a total of 606 such synthetic images rendered under this setup, six of which are shown in Figure 12.

For each object in this experiment, we generated a database containing $K = 300$ shape templates, one for each of the uniformly sampled out-of-plane rotations (see Section 4.3). For each query image, we retrieved the best template pose using

Table 1: Detection failure rates and processing time in highly cluttered scene with multiple objects.

Algorithm	Circuit Breaker	Diamond Part	Ellipse Part	T-Nut	Knob	Wheel	Avg.	Time (sec)
FDCM (ours)	**0.03**	**0.01**	**0.05**	**0.11**	**0.04**	**0.08**	**0.05**	**0.71**
OCM [SBC08]	0.05	0.05	0.14	0.17	**0.04**	0.17	0.10	65.3
CM	0.11	0.22	0.26	0.34	0.26	0.22	0.24	29.1

a brute force search scheme (over all in-plane rotation and translation parameters). Full 3D pose of the objects were then recovered for a known depth using the estimated in-plane transformation parameters together with the out-of-plane rotation parameters that generated the template poses. An estimation was labeled as correct if the position was within 5 mm, and the three estimated angles were each within 10°, of the ground truth pose.

Detection and Localization: We compared the performance of our proposed FDCM to CM (described in Section 3.1) and OCM [SBC08]. The detection failure rates and processing times are shown in Table 1. Whereas CM had an average detection failure rate of 0.24, the proposed FDCM algorithm had a failure rate of only 0.05. It also improved upon the error rate of the competing state-of-the-art matching formulation (OCM) by a factor of 2. Notice that objects with discriminative shapes, such as the diamond part and the circuit breaker part, are easier to detect and localize. In contrast, the T-nut object, which has a simple shape, is relatively difficult to detect, since false edges from clutter and other objects frequently mislead the optimization algorithm. Our FDCM algorithm is $40\times$ faster than CM, and $90\times$ faster than OCM: The average detection time of FDCM was 0.71 seconds, compared to 29.1 seconds for CM and 65.3 seconds for OCM. Several examples of successful detections for various objects in challenging scenarios are shown in Figure 12.

Robustness to Occlusion: We further quantitatively evaluated the robustness of the FDCM algorithm to varying degrees of occlusion, from no occlusion to an average occlusion of 30%. The results are presented in Figure 13. We achieved greater than 99% detection rate up to 5% occlusion, and about 85% detection rate when one-fourth (25%) of the object is occluded.

Two-View Pose Refinement: In this experiment, we evaluate the accuracy of the pose refinement algorithm described in Section 4.5. Using the same set of 6 objects, we render one object at a time in random poses. After a coarse pose estimate was computed, both the refinement schemes using one view and that using two views were applied independently to further refine the estimates. The final pose estimates were compared to the ground truth pose. The results in Table 2, averaged over 6 objects and 100 trials each, demonstrate that the two-view ap-

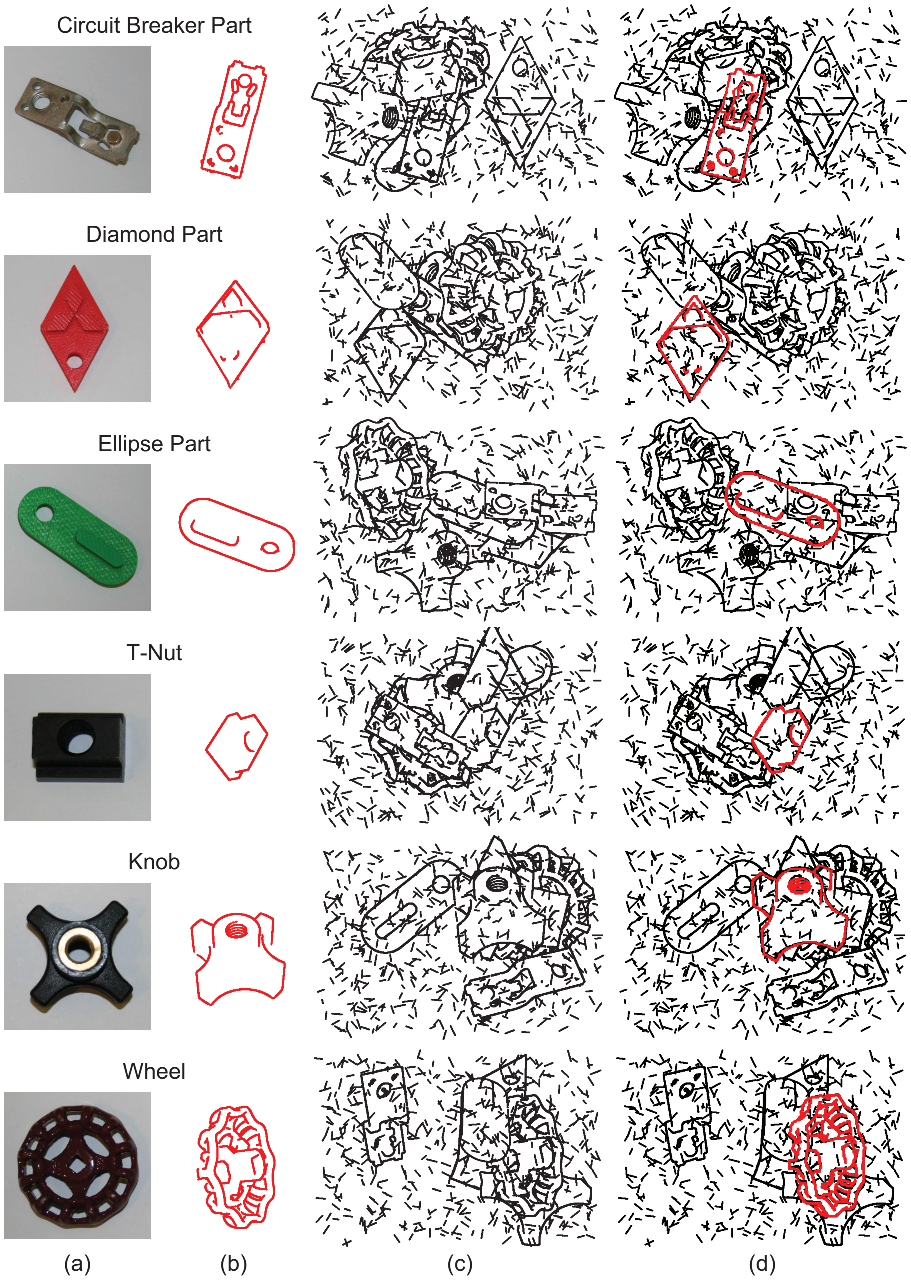

Figure 12: Examples of successful pose estimation on the synthetic dataset. (a) Photo of each part. (b) Sample depth-edge template. (c) Rendered query image. (d) Pose estimation result.

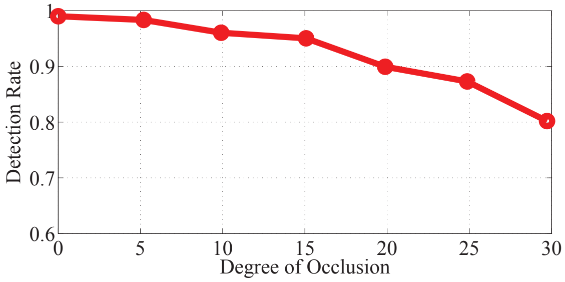

Figure 13: Detection rate versus percentage of occlusion.

Table 2: Comparison of the average absolute pose estimation error between the one-view and two-view approaches.

Average absolute error	t_X mm	t_Y mm	t_Z mm	θ_X degree	θ_Y degree	θ_Z degree
1 View	0.127	0.165	1.156	0.674	0.999	0.349
2 View	0.094	0.096	0.400	0.601	0.529	0.238

proach outperformed the one-view approach. In the rendered images, 1 mm corresponded to about 6.56 pixels on the image plane, indicating that the two-view estimate achieves sub-pixel accuracy.

5.1.2 Real Examples

Object Detection and Pose Estimation in Cluttered Scenes: To quantitatively evaluate performance, we performed several real experiments. Six different types of objects were laid one on top of another in a cluttered manner as shown in Figure 14. We then extracted depth edges using the MFC and performed object detection and pose estimation on the resulting depth-edge images. In each trial of this experiment, we used the system to detect a single instance of an object type. Over several hundred trials, the average detection rate was 95%. Shown in Figure 14 are some typical example trials of this experiment. On each image, we overlay the silhouettes of the detector outputs for three different object types. Notice that some of the parts have no texture, while others are quite specular. In such challenging scenarios, methods based on traditional image edges (e.g., Canny edges) usually fail, but the MFC enables us to robustly extract depth edges. Also notice that since the depth edge features are not affected by texture, our method works robustly even for parts that have artificial texture painted on them. This indicates that the method can work in the presence of oil, grime, or dirt (which are all common in industrial

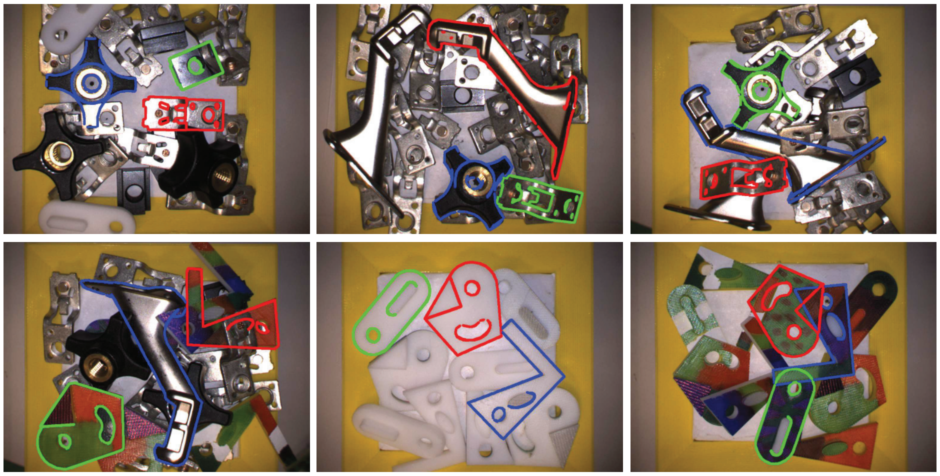

Figure 14: Results using real examples. The system detected and accurately estimated the pose for specular (shiny metal) objects, textureless objects (such as the ones in the bottom center image), and objects that have potentially misleading texture painted on them (such as the ones in the bottom right image). Overlaid on each image is the top detector output for each of three different object types.

environments), all of which add artificial texture to the surface of objects.

Statistical Evaluation: In order to statistically evaluate the accuracy of the proposed system, we need a method of independently obtaining the 3D ground truth pose of the object. Since there was no simple way of obtaining this (especially when objects were stacked on top of each other or piled in a bin), we instead devised a method to evaluate the consistency of pose estimate across multiple viewpoints of the camera. We placed an object in the scene and commanded the robot arm to move to several rotations and translations, so that data are collected when the camera is pointing at the object using many different camera poses. The camera poses were maintained such that the distance along the z-axis between the camera and the object is ± 10 mm from the hypothesized distance t_z that was used to generate the database (Section 4.3). From each camera pose, MFC images were captured, and our algorithm was used to estimate the pose of the object in the camera coordinate system. Since the object is static, the estimated pose of the object in the world coordinate system should be identical irrespective of the viewpoint of the MFC. For each view, the estimated pose of the object was transformed to the world coordinate system using the known position and orientation of the robot arm. We repeated this experiment for 7 different objects, with 25 trials for each object (the object was placed in a different pose for each trial). During each of these independent trials, the robot arm was moved to 40 different viewpoints in order

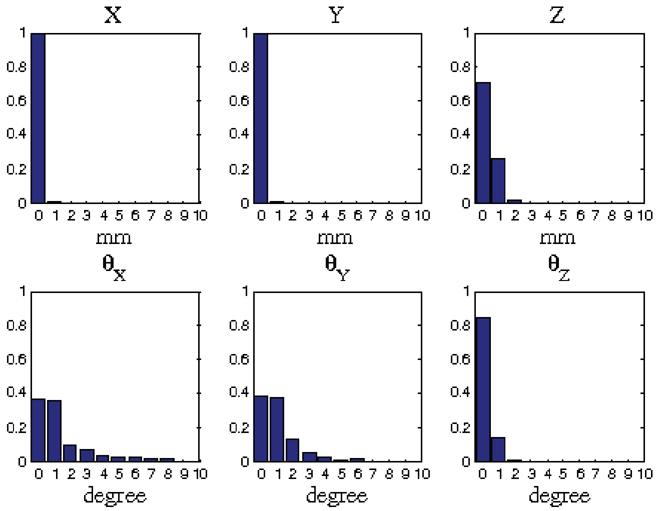

Figure 15: Results from real examples. Histograms of deviations from the median pose estimate, in mm (top) and degrees (bottom), across multiple trials of pose estimation.

to evaluate the consistency of the pose estimates. The histogram of the deviations from the median pose estimate is shown in Figure 15. The results demonstrate that the algorithm computes consistent estimates, with a standard deviation of less than 0.5 mm in the in-plane directions (x, y) and about 2 degrees in each of the three orientation angles. The standard deviation of the estimate in the z direction (along the optical axis of camera) is slightly larger (approximately 1.2 mm).

Effect of Depth Variation: In our experiments, the system was optimized for a part container with a depth variation of 40 mm and a distance along the z-axis of 275 mm from the camera to the top of the part container. As explained in Section 4.3, the pose estimation algorithm requires a rough value of the distance, t_z, from the camera to the objects along the z-axis. In this experiment, we analyze how deviations of the true object distance from the hypothesized distance t_z affect pose estimation accuracy.

We placed a single object in the scene and performed pose estimation at several different camera poses with offsets along the z-axis from the hypothesized distance of 275 mm. At each z offset (height), we repeated the pose estimation for 100 trials by randomly changing the camera pose in the (x, y) directions. As in the previous experiment, we used the median pose estimate as the ground truth pose. An estimate is labeled as correct if the translation error, computed as the Euclidean distance between the (x, y, z) translation vectors, is less than 3 mm and the rotation error, computed as the geodesic distance between two 3D rotations, is less than $8°$.

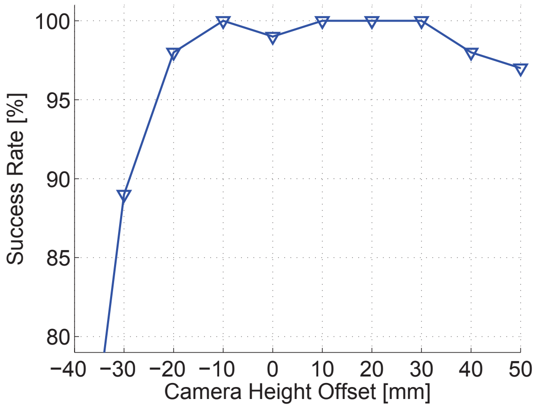

Figure 16: Effect of depth variation on pose estimation.

The accuracy of the system is shown in Figure 16. The pose estimation algorithm is quite robust to depth variations between $[-20, +50]$ mm, which is significantly larger than our target capture range. Outside of this range, our two-view pose refinement algorithm failed to converge to the true solution for several trials, due to the incorrect distance assumption causing large projection errors. This experiment suggests that for part containers with larger depth variations, coarse pose estimation should be performed at multiple scales, targeting different depths, to get a better initial depth estimation. Alternatively, we could move the robot arm and change the height of the capture position based on previous object pose estimates in order to maintain a roughly constant distance between the camera and the objects.

5.1.3 Bin-Picking System Performance

We evaluated the performance of bin picking using the robotic system shown in Figure 5. Extension 1 demonstrates our system accomplishing this task in real time. Figure 5 shows a part container (bin) containing a large number of circuit breaker parts. The gripper (end effector) of the robot arm is designed to grasp each of the objects by first inserting its three metal pins in the closed state through a hole in the object. The gripper then opens by moving the three pins radially outward, thereby exerting outward horizontal forces on the inside edges of the hole. The gripper has a diameter of 3 mm in its closed state, while the hole in the object has a diameter of about 6 mm. Therefore, in order to successfully insert the gripper inside the hole (before lifting the object), the pose estimate error in the (x, y) directions must be less than 1.5 mm. If the pose estimate error is greater, the pins will not be inserted into the hole, resulting in a failure to grasp the object.

Our system was able to successfully guide the robot arm in the grasping task, achieving a grasping success rate of 94% over several hundred trials. There were two main causes for the grasping failures in 6% of the trials: (1) This particular target object has very similar depth edges when it is flipped upside-down, which occasionally led to inaccurate pose estimates; (2) Even when the pose estimation was correct, the hole of the object was occasionally occluded by other objects, resulting in a grasping failure. It is important to note that all of these grasping failures were detected by the error-detection process using the gripper camera, so they did not affect the subsequent assembly task. Among the instances of successful grasping, a few trials resulted in the object being picked up by the gripper in an incorrect pose, due to interference from neighboring objects during the pickup process. These cases were also detected and were corrected automatically by our system using the gripper camera and our process for pose estimation and correction in the gripper (described in Section 4.6).

Processing Time: The entire pose estimation process requires less than 1 second for an object in an extremely cluttered environment (on an Intel quad-core 3.4 Ghz CPU with 3 GB memory). The decomposition of processing time is 0.6 seconds for FDCM and 0.3 seconds for the multi-view pose refinement algorithm. As shown in Extension 1, almost all of the computation occurs during robot motion, so the computation time has almost no effect on the system operation speed. In environments with minimal clutter, the algorithm runs about twice as fast, since there are significantly fewer edges in the captured images.

5.1.4 Pose Estimation in the Gripper

To evaluate the system's potential for error correction in the gripper, we measure the accuracy of pose estimation in the gripper using different numbers of views. In this experiment, we picked up circuit breaker parts from the part container as described in Section 5.1.3. After each pickup, we captured 8 images at different wrist rotation angles, as shown in Figure 17, using the gripper camera and foreground extraction during robot motion (described in Section 4.6). We performed the pose refinement algorithm (Section 4.5) using from 1–8 views (for these experiments, we used the ideal pose of the object as the initial guess).

Figure 18 shows pose estimation errors using different numbers of views. Since the ground truth of the object pose in the gripper is not available, we used the pose that was estimated using all 8 views as the ground truth, and compared it with the poses estimated using different numbers (1–7) of views. The plots show the average estimation errors and standard deviations of these estimation errors (error bars) over 100 trials. Similar to our observations from synthetic data (shown in Table 2), the translation errors on real data are smaller for two-view estimation than for one-view estimation (see Figure 18, left). Two-view pose estimation is

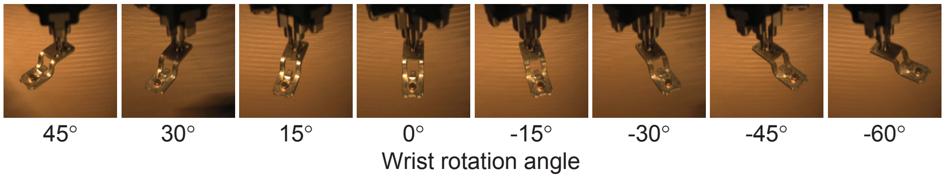

Figure 17: Example eight views captured with different wrist rotation angles.

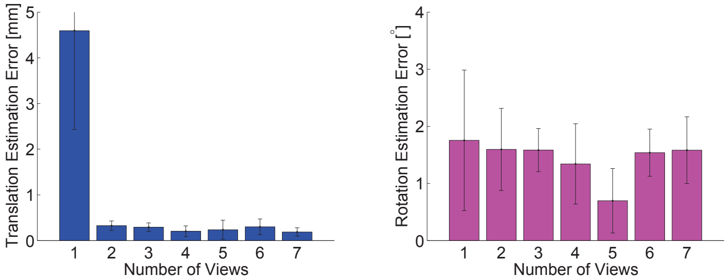

Figure 18: Pose estimation errors in the gripper using different numbers of views.

sufficient for accurate pose correction (error less than 0.5 mm), and using more than two views does not further improve the estimation. The rotation estimation errors are roughly the same (< 2 degrees) for each number of views. Figure 19 shows a typical example of two-view pose estimation. The estimated pose of the object matches the object's outline in both views quite closely, illustrating the high accuracy of the estimated pose. The difference between this estimated pose and the ideal pose provides an idea of the typical size of the initial pose error in the gripper. Our system automatically estimates and corrects this error in the gripper.

5.2 Deformable Object Detection

We applied the FDCM algorithm to object detection and localization on the ETHZ shape class dataset [FTG06]. The dataset consists of 255 images, each of which contains one or more objects from five different object classes: apple logos, bottles, giraffes, mugs, and swans. The objects have large variations in appearance, viewpoint, size, and non-rigid deformation. We followed the experimental setup proposed in [FTG06, FJS10], in which a single hand-drawn shape for each class is used to detect and localize its instances in the dataset.

Our detection system is based on scanning using a sliding window. We retain all the hypotheses whose matching costs are less than the detection threshold. We densely sampled the hypothesis space and searched the images at 8 different scales

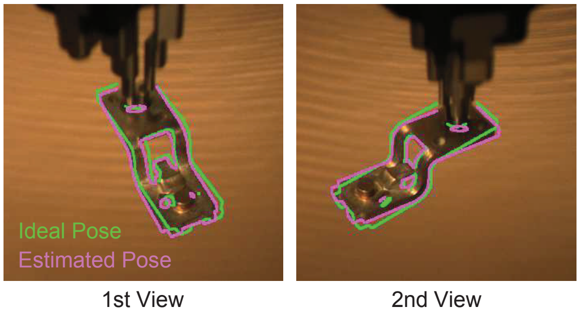

Figure 19: Two-view pose estimation in the gripper. The ideal pose is used as the initial guess and refined to give the estimated pose. (Both poses are superimposed on the input images of the two views).

and 3 different aspect ratios. The ratio between two consecutive scales is 1.2 and between consecutive aspect ratios is 1.1. We performed non-maximal suppression by retaining only the lowest-cost hypothesis among any group of detections that have significant spatial overlap.

In Figure 20, we plot detection rate vs. false positives per image. The curve is generated via altering the detection threshold for the matching cost. We compared our approach with OCM [SBC08] and two recent studies by Ferrari et. al. [FTG06, FJS10]. Our approach outperforms OCM at all the false positive rates and is comparable to [FJS10]. Compared to [FJS10], our results are better for two classes (giraffes and bottles) and slightly worse for the swans class, while for two other classes (apple logos and mugs), the numbers are almost identical. As shown in the detection examples (Figure 21), object localization is highly accurate. Note that [ZWWS08] and [RJM08] report slightly better performance on this dataset, but we could not include their results in our graphs because their results were only reported in graphical format (as precision-recall curves). Also note that these methods are orders of magnitude slower than FDCM.

Complexity Comparison: The average number of points in the shape templates were $1,610$, computed over five classes. Our line-based representation used an average of 39 line segments per class. Note that the number of lines per class provides an upper bound on the number of computations required. Since the algorithm retrieves only the hypotheses having a smaller cost than the detection threshold, the summation was terminated for a hypothesis if the cost exceeded this value. By using this bound in the hypothesis domain (see Section 3.6.1 for more details), on average only 14 line segments were evaluated per hypothesis.

The average evaluation time for a single hypothesis was $0.40\ \mu s$ using FDCM, whereas this process took $51.50\ \mu s$ for OCM and $17.59\ \mu s$ for CM. The proposed

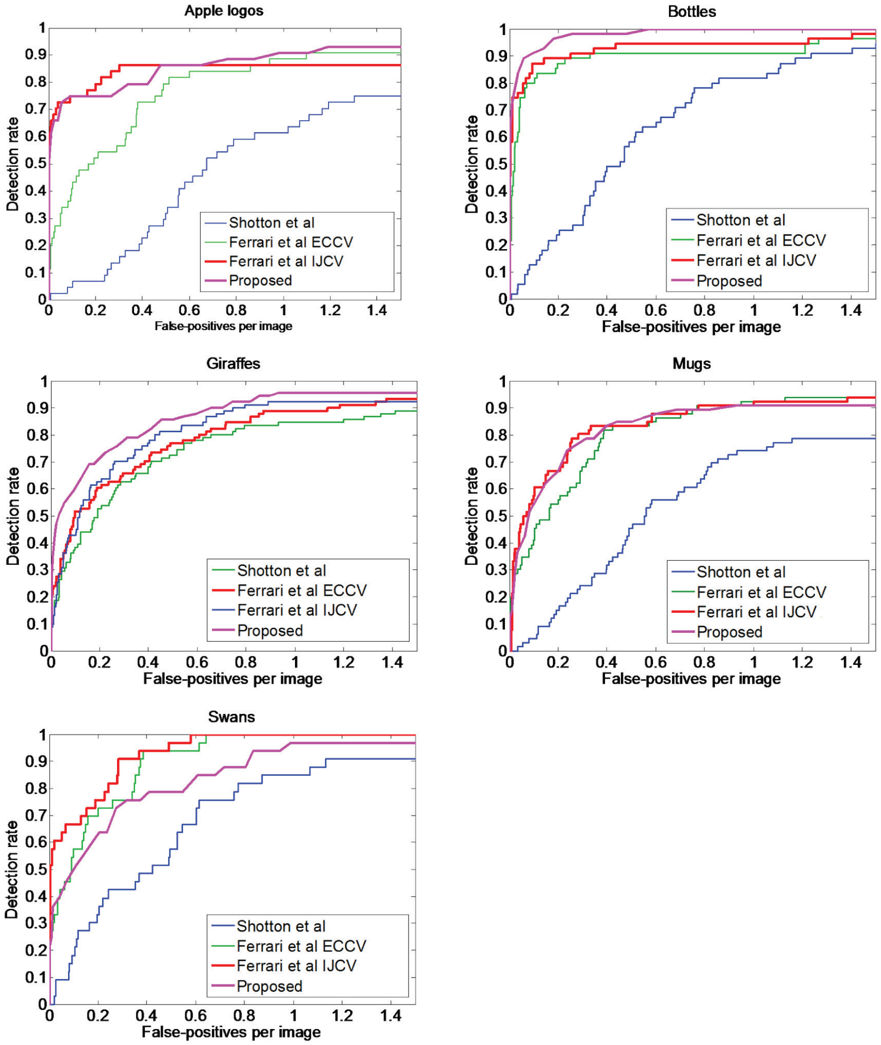

Figure 20: Receiver operating characteristic (ROC) curves on the ETHZ shape dataset comparing our proposed approach to OCM [SBC08] and two recent studies by Ferrari et. al. [FTG06, FJS10].

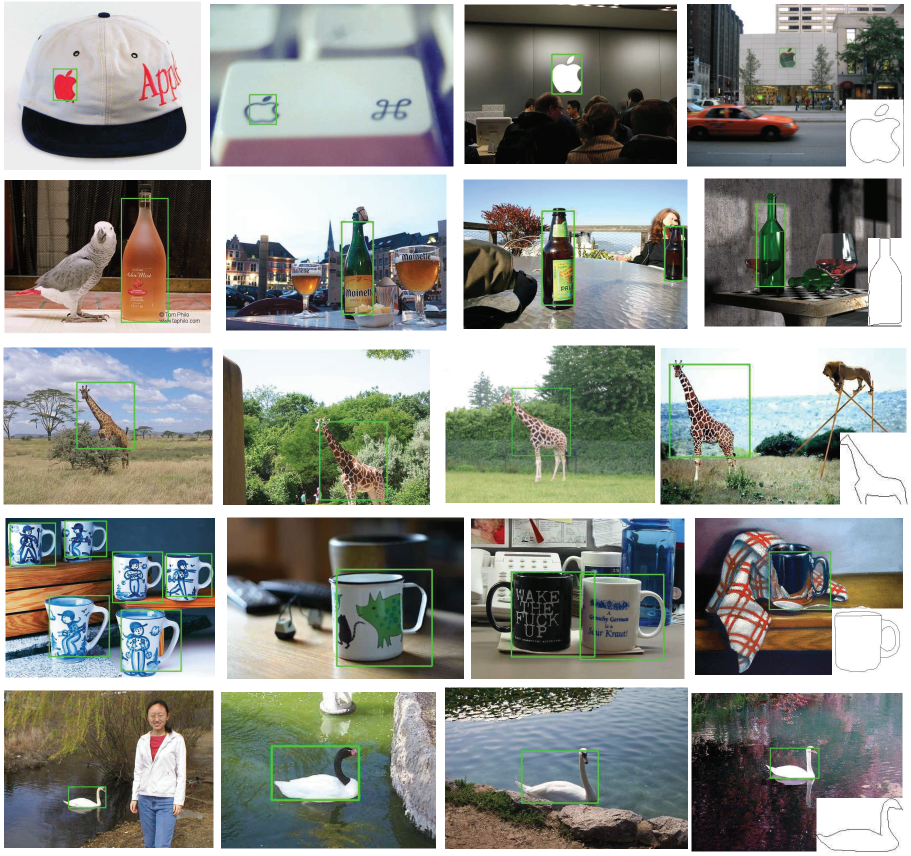

Figure 21: Several localization results on the ETHZ shape dataset. The images are searched using a single hand-drawn shape shown in the lower right of the images in the rightmost column.

method is $43\times$ faster than chamfer matching and $127\times$ faster than oriented chamfer matching. Note that the speed up is more significant for larger-sized templates, since our cost computation is insensitive to the template size, whereas the cost of standard chamfer matching increases linearly.

On average, we evaluated 1.05 million hypotheses per image, which took 0.42 seconds. Using the bound in the spatial domain presented in Section 3.6.2 enabled 91% of the hypotheses to be skipped, reducing the average evaluation time per image to 0.39 seconds. Note that the speedup is not proportional to the fraction of hypotheses skipped because in order to use the bound in the spatial domain, we could no longer use the bound in the hypothesis domain (Section 3.6.1).

Table 3: Pose estimation errors on three action sequences. Errors are measured as the mean absolute pixel distance from the ground truth marker locations.

Algorithm	Walking	Jogging	Boxing	Average
FDCM (ours)	**7.3**	**12.5**	**9.7**	**9.8**
OCM [SBC08]	15.0	15.3	13.6	14.6
CM	9.3	13.6	10.6	11.2

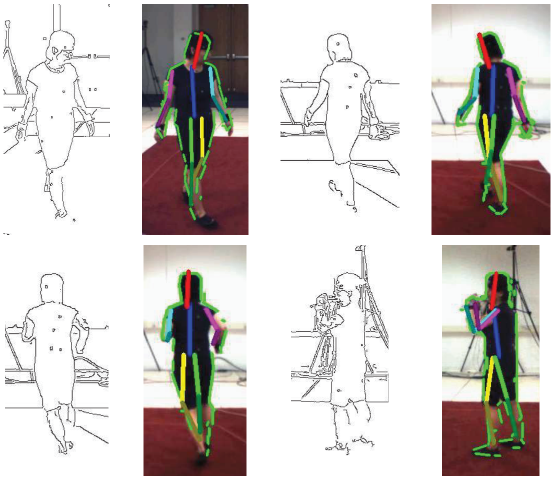

Figure 22: Human pose estimation results. *First row:* Walking sequence. *Second row:* Jogging and boxing sequences. Estimated poses and contours are overlayed on the images.

5.3 Human Pose Estimation

We utilized our shape-matching framework for human pose estimation, which is a highly challenging task due to the large set of possible articulations of the human body. As proposed in [MM02], we matched a gallery of human shapes that have known poses to each test image. Due to articulation, the size of the pose gallery needed for accurate pose estimation is large. Hence, it becomes increasingly important to have an efficient matching algorithm that can cope with background clutter.

The experiments were performed on the HumanEva dataset [SB06a], which contains video sequences of multiple human subjects performing various activities captured from different viewing directions. The ground truth locations of human joints at each image were extracted using attached markers. Shape gallery templates were acquired in two steps. First, we computed the human silhouettes via

HumanEva background subtraction code. Then, using the Canny edges around the extracted silhouette outlines, we obtained the shape templates. We performed the experiment on video sequences from one subject of three actions: walking, jogging, and boxing. For each action, we included all of the images from the subject's training sequence (about $1,000$–$2,000$ images) in the shape gallery. We used this to estimate the subject's pose in the corresponding validation sequence. As we extracted Canny edges directly from the validation images, they included us significant amount of background clutter. The best shape template, together with its scale and location, were then retrieved via the matching framework. We quantitatively evaluated the mean absolute error between the ground truth marker locations and the estimated pose on the image plane. The results, presented in Table 3, demonstrate significant improvements in accuracy compared to OCM and CM. Our proposed approach can evaluate more than 1.1 million hypotheses per second, whereas CM and OCM can evaluate only $31,000$ and $14,000$ hypotheses per second, respectively. Examples of pose estimation are shown in Figure 22.

The code for FDCM can be downloaded from the first author's website.

6 Conclusion

We presented a practical robotic system that uses novel computer vision hardware and algorithms for detection and 3D pose estimation of industrial parts in a cluttered bin. Our implementation of the system on a robotic arm achieves accuracies on the order of 1 mm (reduced to less than 0.5 mm with automatic error correction) and $2°$, with a total processing time of less than 1 second. Given a CAD model, a new object can be integrated into our system in less than 10 minutes. Our goal with this line of research is to make substantial progress towards more versatile and easy-to-customize robotic bin-picking systems.

Acknowledgements

We thank Amit Agrawal, Srikumar Ramalingam, Ramesh Raskar, Keisuke Kojima, and Jay Thornton for their helpful comments and suggestions. Special thanks to John Barnwell for his invaluable help with the hardware. This research was mostly conducted at MERL with support from MERL. Ming-Yu Liu and Ashok Veeraraghavan contributed to the work while they were at MERL. At Rice University, Ashok Veeraraghavan was partially supported by NSF grant NSF 1117939. Rama Chellappa was supported by MURI from the Office of Naval Research under the Grant N00014-08-I-0638.

A Index to Multimedia Extensions

The multimedia extensions to this article are at: http://www.ijrr.org.

Table 4: Multimedia Extensions

Extension	Type	Description
1	Video	System demo video. In the video, we demonstrate the robustness and real-time performance of the proposed system for picking up parts in a challenging setting. Through showing 10 consecutive pickups of an industrial part from a highly cluttered bin, we illustrate the applicability of the proposed system to industrial assembly. Each of the major components of the algorithm is highlighted. The video also contains several close-up views of the pickups.

References

[ASBR10] Amit Agrawal, Yu Sun, John Barnwell, and Ramesh Raskar. Vision-guided robot system for picking objects by casting shadows. *International Journal of Robotics Research*, 29:155–173, February 2010.

[AT06] Ankur Agarwal and Bill Triggs. Recovering 3d human pose from monocular images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(1):44–58, 2006.

[BB09] Csaba Beleznai and Horst Bischof. Fast human detection in crowded scenes by contour integration and local shape estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2246–2253, 2009.

[BBM05] Alexander C. Berg, Tamara L. Berg, and Jitendra Malik. Shape matching and object recognition using low distortion correspondences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 26–33, 2005.

[BETG08] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc J. Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[BJ03] Ronen Basri and David W. Jacobs. Lambertian reflectance and linear subspaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(2):218–233, 2003.

[BMP02] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, 2002.

[BN10] Prabin Bariya and Ko Nishino. Scale-hierarchical 3d object recognition in cluttered scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[BTBW77] Harry G Barrow, Jay Martin Tenenbaum, Robert Coy Bolles, and Helen C Wolf. Parametric correspondence and chamfer matching: Two new techniques for image matching. In *Proceedings of the 5th international joint conference on Artificial intelligence*, pages 659–663, 1977.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, March 2004.

[Can86] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679 –698, 1986.

[CSD$^+$10] Yan Cui, Sebastian Schuon, Chan Derek, Sebastian Thrun, and Christian Theobalt. 3d shape scanning with a time-of-flight camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[DBdFF02] Pinar Duygulu, Kobus Barnard, João F. G. de Freitas, and David A. Forsyth. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In *Proceedings of the European Conference on Computer Vision*, pages 97–112, 2002.

[DCS09] Oscar Danielsson, Stefan Carlsson, and Josephine Sullivan. Automatic learning and extraction of multi-local features. In *Proceedings of the IEEE International Conference on Computer Vision*, 2009.

[DD92] Daniel DeMenthon and Larry Davis. Model-based object pose in 25 lines of code. In *Proceedings of the European Conference on Computer Vision*, pages 335–343, 1992.

[DT05] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 886–893, 2005.

[DUNI10] Bertram Drost, Markus Ulrich, Nassir Navab, and Slobodan Ilic. Model globally, match locally: Efficient and robust 3d object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 998–1005, 2010.

[FB81] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Graphics and Image Processing*, 24(6):381–395, 1981.

[FFFP06] Li Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611, 2006.

[FFJS08] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid. Groups of adjacent contour segments for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(1):36–51, 2008.

[FH04] Pedro Felzenszwalb and Daniel Huttenlocher. Distance transforms of sampled functions. Technical Report TR2004-1963, Cornell Computing and Information Science, 2004.

[FJS10] Vittorio Ferrari, Frederic Jurie, and Cordelia Schmid. From images to shape models for object detection. *International Journal of Computer Vision*, 87:284–303, 2010.

[FS07] Pedro Felzenszwalb and Joshua Schwartz. Hierarchical matching of deformable shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[FTG06] Vittorio Ferrari, Tinne Tuytelaars, and Luc Van Gool. Object detection by contour segment networks. In *Proceedings of the European Conference on Computer Vision*, volume 3953, pages 14–28, 2006.

[Gav98] Dariu M. Gavrila. Multi-feature hierarchical template matching using distance transforms. In *Proceedings of the IEEE International Conference on Pattern Recognition*, pages 439–444, 1998.

[GLP84] Eric Grimson and Tomás Lozano-Pérez. Model-based recognition and localization from sparse range or tactile data. *International Journal of Robotics Research*, 3:3–35, 1984.

[Hor87] Berthold K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society A*, 4(4):629–642, 1987.

[JH99] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):433–449, 1999.

[LJ07] Haibin Ling and David W. Jacobs. Shape classification using the inner-distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(2):286–299, 2007.

[Low87] David Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, 31(3):355–395, 1987.

[Low91] David Lowe. Fitting parameterized three-dimensional models to images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(5):441–450, 1991.

[Low04] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[LSP06] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2169–2178, 2006.

[LTV$^+$10] Ming-Yu Liu, Oncel Tuzel, Ashok Veeraraghavan, Rama Chellappa, Amit Agrawal, and Haruhisa Okuda. Pose estimation in heavy clutter using a multi-flash camera. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2010.

[LTVC10] Ming-Yu Liu, Oncel Tuzel, Ashok Veeraraghavan, and Rama Chellappa. Fast directional chamfer matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 2010.

[MM02] Greg Mori and Jitendra Malik. Estimating human body configurations using shape context matching. In *Proceedings of the European Conference on Computer Vision*, volume 3, 2002.

[MREM04] Greg Mori, Xiaofeng Ren, Alexei A. Efros, and Jitendra Malik. Recovering human body configurations: Combining segmentation and recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 326–333, 2004.

[NFF07] Juan Carlos Niebles and Li Fei-Fei. A hierarchical model of shape and appearance for human action classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[NPK96] Bradley Nelson, N.P. Papanikolopoulos, and Pradeep Khosla. Robotic visual servoing and robotic assembly tasks. *IEEE Robotics and Automation Magazine*, 3(1):23–31, June 1996.

[OH97] Clark F. Olson and Daniel P. Huttenlocher. Automatic target recognition by matching oriented edge pixels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(1):103–113, 1997.

[OT01] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175, 2001.

[RJM08] Saiprasad Ravishankar, Arpit Jain, and Anurag Mittal. Multi-stage contour based detection of deformable objects. In *Proceedings of the European Conference on Computer Vision*, pages 483–496, 2008.

[RTF$^+$04] Ramesh Raskar, Kar-Han Tan, Rogerio Feris, Jingyi Yu, and Matthew Turk. Non-photorealistic camera: depth edge detection and stylized rendering using multi-flash imaging. *ACM Transactions on Graphics*, 23(3):679–688, 2004.

[SB06a] Leonid Sigal and Michael J. Black. Humaneva: Synchronized video and motion capture dataset for evaluation of articulated human motion. Technical report, Brown University, 2006.

[SB06b] Leonid Sigal and Michael J. Black. Measure locally, reason globally: Occlusion-sensitive articulated pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2041–2048, 2006.

[SBC08] Jamie Shotton, Andrew Blake, and Roberto Cipolla. Multi-scale categorical object recognition using contour fragments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(7):1270–1281, 2008.

[SI73] Y. Shirai and H. Inoue. Guiding a robot by visual feedback in assembling tasks. *Pattern Recognition*, 5(2):99 –108, 1973.

[SM92] Fridtjof Stein and Gérard Medioni. Structural indexing: Efficient 3-d object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):125 –145, 1992.

[SS03] Daniel Scharstein and Richard Szeliski. High-accuracy stereo depth maps using structured light. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[STFW05] Erik B. Sudderth, Antonio Torralba, William T. Freeman, and Alan S. Willsky. Learning hierarchical models of scenes, objects, and parts. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1331–1338, 2005.

[TPM08] Oncel Tuzel, Fatih Porikli, and Peter Meer. Pedestrian detection via classification on riemannian manifolds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(10):1713–1727, 2008.

[TSTC03] Arasanathan Thayananthan, Bjorn Stenger, Philip H. S. Torr, and Roberto Cipolla. Shape context and chamfer matching in cluttered scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 127–133, 2003.

[Vis] VisionPro. http://www.cognex.com/visionpro/. *Cognex*.

[VJ01] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 511–518, 2001.

[Zha94] Zhengyou Zhang. Iterative point matching for registration of free-form curves and surfaces. *International Journal of Computer Vision*, 13(2):119–152, 1994.

[ZWWS08] Qihui Zhu, Liming Wang, Yang Wu, and Jianbo Shi. Contour context selection for object detection: A set-to-set contour matching approach. In *Proceedings of the European Conference on Computer Vision*, pages 774–787, 2008.