2017 IEEE International Conference on Robotics and Automation (ICRA)
Singapore, May 29 - June 3, 2017

Fast Task-Specific Target Detection via Graph Based Constraints Representation and Checking

Wentao Luan[1], Yezhou Yang[3], Cornelia Fermüller[2], John S. Baras[1]

Abstract— **We present a framework for fast target detection in real-world robotics applications. Considering that an intelligent agent attends to a task-specific object target during execution, our goal is to detect the object efficiently. We propose the concept of early recognition, which influences the candidate proposal process to achieve fast and reliable detection performance. To check the target constraints efficiently, we put forward a novel policy which generates a sub-optimal checking order, and we prove that it has bounded time cost compared to the optimal checking sequence, which is not achievable in polynomial time. Experiments on two different scenarios: 1) rigid object and 2) non-rigid body part detection validate our pipeline. To show that our method is widely applicable, we further present a human-robot interaction system based on our non-rigid body part detection.**

I. Introduction

When robotics researcher address applications requiring visual perception to allow for interaction with the environment, they usually adopt Computer Vision techniques. However, the state-of-the-art Computer Vision pipelines are not well suited for autonomous robotics. Take as example the object recognition pipeline. Most recent approaches rely on a general object candidate proposal procedure to generate regions (both RGB or RGB-D), which likely contain objects. After this object proposal stage, pre-trained classifiers, such as pre-trained Convolutional Neural Nets (CNN), evaluate each candidate's region and determine whether the region contains one of the target objects [4], [9].

The above pipelines are considered effective and efficient for Multimedia applications, such as image tagging and retrieval. However, they are not directly applicable for Robotics applications. The reason is that during the execution of a task or a particular phase of the task, the robot needs to localize only the task-specific object in a fast and reliable fashion. For example, for a humanoid robot to open a microwave, only the microwave's exact pose and handle location are critical for successful execution, while other objects that happen to be in the scene can either be ignored or simply represented as generic geometric objects, such as boxes or cylinders, for collision check.

Thus, the general object recognition pipeline based on object candidate proposals becomes redundant for two reasons: 1) before executing a task, the robot is aware of what object to focus on from the task description; 2) considering the general situation without specific task, will hurt the system's overall detection performance.

The authors are with Institute for Systems Research, University of Maryland, College Park, USA[1], Computer Vision Lab, University of Maryland, College Park, USA[2], School of Computing, Informatics, and Decision Systems Engineering, Arizona State University[3] Email: wluan@umd.edu, yz.yang@asu.edu, fer@umiacs.umd.edu, baras@umd.edu

Here, we present a novel strategy to tackle the object recognition problem in a robotic manipulation setting. We propose to consider the constraints from the target object early on during the candidate proposal process in order to speed up the task-specific object detection during robotic execution. However, the main technical difficulty of the new pipeline is due to the large number of constraints for real world objects. Let's consider the underlying distribution of the total real world target objects, each detection constraint shall contribute differently to the target localization. In this work, we formulate the problem as a filtering problem and by achieving a sub-optimal order of constraints to check, our system is able to reject the negative instances early and thus significantly reduce the amount of time for target detection.

We summarize our contribution as follows:

1) We demonstrate the feasibility and benefits of introducing target descriptions early into the segmentation and object candidate proposal procedure for robotic applications.
2) The process of checking a target's constraints is formulated as a shared filter problem, and we prove that a greedy strategy of organizing constraint filters has a bounded performance with regard to the optimal checking sequence. The optimized order can be interpreted intuitively as a task-specific attention mechanism under the current working conditions.
3) We implement and apply the presented framework to two different real world scenarios: the detection of drawers with handles and the detection of a user's hands and arms. The experimental results show that: 1) the optimized constraints checking order is time-efficient; 2) our detection framework is general enough to deal with both rigid objects and deformable objects.

II. Related Work

Object detection and recognition is a problem widely studied within the Computer Vision and Robotics communities. Various object detection pipelines have been proposed for different contexts and different applications.

Object candidate proposal followed by classification has become a dominant procedure for object detection. First, proto-objects or possible object areas are generated either by segmentation [3], [19] or searching [18], [23] using low-level visual cues. High-level knowledge such as context [6], [14], [25] and bio-inspired attention [8] can be added to help reduce the number of candidates and make the search more efficient. After pruning the search space, features [2] and attributes [15] can be extracted and classified by one or multiple statistical models [13], [24]. Recently, deep neural network based approaches [4], [12] became popular due to their performance and their way of handling features and classification simultaneously. However, a general candidate proposal approach is not suited well to deliver a target-specific task for a robot. The traditional detection pipeline would be computationally redundant, given the potentially large number of object candidates.

Another class of methods widely adopted in robotics applications employs keypoints [1] and model matching [7]. Especially when depth is available, 3D descriptors [20], [21] can encode the shape, and they perform well when considering them in conjunction with color [10]. However, though dealing with specific object instances, these methods spend a significant amount of resources on finding the key points. Also, by storing the complete 3D model and comprehensive views, the detection process is redundant and difficult to generalize.

Here, we propose the concept of early recognition, which influences the candidate proposal process to achieve a fast and reliable target detection. In our framework, the target object is described as a graph, and visual cues like attributes are treated as the constraints to be followed by the graph elements. In image processing, graph related models like Markov Random Fields have been used to recognize or segment the target [26]. Previous approaches, however, focus on the recognition accuracy and thereby require a full list of attributes. In our framework, we present a novel way to speed up the detection process by optimizing the order of the visual constraints to check. Similar to algorithms in data mining [11], [17], we adopt a greedy algorithm in ordering the visual constraints. Moreover, we provide a theoretical foundation for our approach by proving its submodular property [5].

III. Our Approach

We illustrate the system's workflow in Figure 1. Given an input RGB-D image (Fig. 1(a)), the system first generates a scene graph $G = \{V, E\}$ by segmenting the image into surfaces V (Fig. 1(b)). At first, E contains all the possible connections, and G is a fully-connected graph. We then represent the knowledge about the target object as a set of template graphs $\mathbb{GT}$, with constraint functions associated with each vertex and edge. During the main detection procedure, our system checks sequentially the constraints provided by the description of the target object $\mathbb{GT}$ to remove negative matches between the scene and the template graph (Fig. 1(c)). In the end, the system returns the subgraphs satisfying all the target constraints, which provides the target object candidates (Fig. 1(d)).

Since our system considers the constraints from the target object early in the process, the procedure of finding candidate proposals becomes target-specific, and the recognition phase becomes a part of the constraints checking. Here, the task of efficient target detection can be formulated as "how to find the target (a subgraph that matches the template graph) from the segmented input image (scene graph) efficiently."

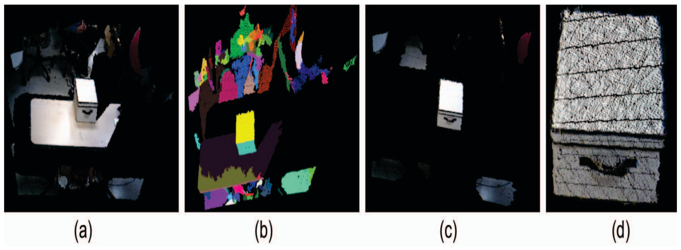


Fig. 1: Detection pipeline. (a) Input image; (b) Scene graph after segmentation; (c) Intermediate result: a visualization of scene graph nodes with more than 1 active match with the template graph after constraints checking; (d) Detection output after matching template graphs.

IV. Problem Formulation

The first stage of our pipeline segments the input image and generates a scene graph $G = \{V, E\}$. Here, V is the set of segmented surfaces, and E represents the relationship between surfaces.

As mentioned before, we describe the target object as a set of template graphs $\mathbb{GT}$ along with a set of constraint functions. $\mathbb{GT} = \{G_i | i = 1, 2, ..., N\}$ and $G_i = \{V_i, F^{V_i}, E_i, F^{E_i}\}$, where V_i denotes the nodes (surfaces) in i-th template graph, $F^{V_i} = \cup_{v \in V_i}\{F^v\}$ represents the set of vertex constraints F^v for each node $v \in V_i$. The result of matching each constraint F with vertex v is a random variable F_v with values of $\{false, true\}$. Here, value $true$ means a constraint is satisfied. If F is evaluated to be $false$, the matching to the template vertex v will be rejected. In our case, one constraint can be used to match with different vertices from the template graphs, i.e. it is possible to have $F \in F^{v1}$ and the same $F \in F^{v2}$, where $v1 \neq v2$.

E_i is the set of edges in i-th template graph. $F^{E_i} = \cup_{e \in E_i}\{F^e\}$ are the edge constraints for all of them. It is worth noting that the nodes in different template graphs have the same index if both their nodes and edges constraints are identical $V_T = \cup_i V_i$ and $E_T = \cup_i E_i$ denote all the nodes and edge labels in the template graph. $\mathbb{F} = \{\cup_{v \in V_T} F^v)\} \cup \{\cup_{e \in E_T} F^e\}$ represent all the constraints associated with nodes and edges.

Given a scene graph of an input image, $G = \{V, E\}$, we want to find all the subgraphs of G which match one of the templates in $\mathbb{GT}$ efficiently. That is,

$$\begin{aligned} \text{minimize} \quad & c_{avg}(I) \\ \text{subject to} \quad & I \in \text{Permutation}(1, 2, ..., |\mathbb{F}|), \end{aligned} \tag{1}$$

where I denotes one of the constraint checking sequence. $c_{avg}(I)$ denotes the expected cost of checking the constraints following the order of I. Here the cost originates from the temporal ordering, because the goal of our system is to detect the target object as fast as possible.

A naive approach of searching template graphs in G is to check all the constraints $\mathbb{F}$ in random order. The downside of such an approach is obvious. It neglects the cost for checking the constraints.

On the other hand, searching for the optimal order is computationally expensive because of the potentially exponential number of possible graph matches. The computational complexity to exhaustively test each of the constraint checking orders is non-polynomial. Thus, a computationally affordable strategy for determining the constraints checking sequence is desirable for efficient target search. In the following sections we will introduce our take. Our system's output is a sub-optimal constraint checking order. The experimental results show that our approach is able to significantly reduce the time for target detection.

V. Constraints Order Searching Policy

In this section, we first clarify the constraints' checking procedure, then put forward our constraints order searching algorithm. Finally, we prove that the checking order determined by our policy holds theoretical performance guarantee under a set of assumptions.

For the sake of clarity, we introduce $U(s|\mathbb{F}')$ to be the set of possible template graph node labels that a scene node s matches, after checking the set of constraints in $\mathbb{F}' \subseteq \mathbb{F}$. Intuitively, s could match with template graph node v if all the node and edge constraints in $\mathbb{F}'$ associated with node v are satisfied.

We present in Algorithm 1 the procedure of matching a scene graph with template graphs. In a nutshell, the algorithm checks all the template graph constraints in $\mathbb{F}$ for each vertex or edge in the scene graph G. After filtering out the negative matches between the scene and template graph, our system returns all the subgraphs from the scene graph whose corresponding matched nodes and edges form one of the template graphs.

In our use case application, one legitimate assumption is that the number of possible matches for each scene vertex, after checking all the constraints in $\mathbb{F}$, is limited. Thus, returning the remaining subgraphs of scene graph that matches one of the template graphs (Algorithm 1 line 16) is expected to take a reasonable amount of time. Traditional graph searching algorithms such as depth-first search and breath-first search are also expected to deliver decent performance. Here, we treat the constraints associated with the graph edges like the node constraints, Thus a matching is determined by whether the scene vertices pass through the filter (or satisfy the constraints) of the template node.

It is not hard to notice that in Algorithm 1, the checking order of constraints (line 2 - 15 in Algorithm 1) influences the overall processing time, though it does not alter each node's final matching output $U(s|\mathbb{F})$. Intuitively, if a constraint can exclude a large portion of scene vertices from matching template graph vertices using low temporal cost, then the computational cost of the following constraints checking (with higher computational cost) is expected to be reduced significantly. In other words, the order of the constraints to check matters.

Algorithm 1: The Procedure of Matching the Scene's Subgraphs to Template Graphs

```
Parameters:
    A set of template graphs $\mathbb{GT}$;
    An ordered list of constraints to check: $I$
Input:
    Scene graph $G$.
Output:
    A set of subgraphs of $G$ matching to one of the
    template graphs.
1:  Initialize the observation set $\mathbb{F}_{ob} = \emptyset$,
    Every node can match to any template graph node at
    the beginning: $U(s|\mathbb{F}_{ob}) = V_T \ \forall s \in V$,
2:  for $i = 1, 2, ..., |I|$ do
3:      Denote $R(F_{I_i}) = \{v \in V_T | F_{I_i} \in F^v\}$
4:      for all $v_r \in R(F_{I_i})$ do
5:          for all $s \in G$ do
6:              if $v_r \in U(s|\mathbb{F}_{ob})$ then
7:                  Check constraint $F_{I_i}$ on vertex $s$ if $F_{I_i}$ is a
                    node constraints. Or check on $s$'s edges if it
                    is an edge constraint.
8:                  if false then
9:                      Update $U(s|\mathbb{F}_{ob}) = U(s|\mathbb{F}_{ob}) / \ \{v_r\}$
10:                 end if
11:             end if
12:         end for
13:     end for
14:     $\mathbb{F}_{ob} = \mathbb{F}_{ob} \cup \{F_{I_i}\}$
15: end for
16: Return all the subgraphs matching one of the template
    graphs.
```

To formulate the ordering problem, let us denote c_j as the cost of checking constraint F_j on a scene vertex, and $P(F_{I_t}|I)$ as the probability that constraint F_{I_t} needs to be checked following the checking order of I. Intuitively, we aim to minimize the expected cost of checking a scene node s following constraints checking order I:

$$\begin{aligned} \text{minimize} \quad & \sum_{t=1}^{|\mathbb{F}|} c_{I_t} P(F_{I_t}|I) \\ \text{subject to} \quad & I \in \text{Permutation}(1, 2, ..., |\mathbb{F}|) \end{aligned} \tag{2}$$

The optimization formulation can also be interpreted as a dual problem: given a cost budget, minimize the possible matches between the vertices of the scene and the template graph. Here we assume that the cost of checking each constraint is static and independent from other constraints.

First consider a special case, where there is only one vertex in the template graph. Without loss of generality, let us denote $V_T = \{v\}$, and $\mathbb{F} = F^v$.

Lemma 5.1: Define a greedy constraint checking order

IG:

$$IG(k) \in \underset{i \in \{1,...,|\mathbb{F}|\}/\{IG\}}{\text{argmax}} \begin{cases} \frac{P(F_i = false)}{c_i} & \text{if } k = 1, \\ \frac{P(F_i = false | F_{IG_1} = true, ..., F_{IG_{k-1}} = true)}{c_i} & \text{o.w.} \end{cases} \quad (3)$$

If the conditional filtering effect of each constraint in F^v is non-increasing, i.e. if index set $A_1 \subseteq A_2$ and $\forall j \notin A_2, P(F_j = false | F_i = true, i \in A_2) \leq P(F_j = false | F_i = true, i \in A_1)$, then the expected cost of checking if a scene graph vertex matches with v following the checking order of IG is the minimum among all the static sequences.

Proof: If a scene node can match a template node v, then all the constraints in $\mathbb{F}$ are satisfied. Thereby all possible permutations have the same checking cost: $\sum_{i \in \mathbb{F}} c_i$, because the algorithm checks each scene vertex with all the constraints. So we want to find a sequence to minimize the expected cost of rejecting matching a scene node to v.

Define an objective function $g : 2^{|\mathbb{F}|} \times 2^{|\mathbb{F}|} \to R^+$ as

$$g(A, O(A)) = \begin{cases} 1 & \text{if } \exists i \in A, F_i = false, \\ 0 & \text{o.w,} \end{cases} \quad (4)$$

where $O(A)$ is the constraints checking result of constraints indexed by A. We assume that at least one of the constraints in $\mathbb{F}$ returns false because we are dealing with a non-matching case. Here, since we are considering a special case of a single node in the template graph, our goal is to minimize the expected cost of constraints checking when g reaches value 1. We have:

(1) g is strong adaptive non-decreasing: $\forall A \subseteq \{1, 2, ...|\mathbb{F}|\}$ and for all possible corresponding observations $O(A)$, $g(A, O(A)) \leq g(A \cup \{j\}, O(A) \cup \{F_j = o\})$ $\forall o \in \{true, false\}, \forall j \notin A$. It means that the objective function value does not decrease with more observations coming in.

(2) g is adaptive submodular: $\forall A_1, A_2$, s.t. $A_1 \subseteq A_2$, $O(A_1) \subseteq O(A_2)$, $\forall j \notin A_2$,

$$\begin{aligned} \mathbb{E}[g(A_2 \cup \{j\}) | O(A_2)] - g(A_2 | O(A_2)) \leq \\ \mathbb{E}[g(A_1 \cup \{j\}) | O(A_1)] - g(A_1 | O(A_1)) \end{aligned} \quad (5)$$

Intuitively, this means that the marginal gain of the objective function g is non-increasing.

Here is our proof. If one of the constraints in A_2 returns false, the left-hand side of Eq. (5) is 0 since the matching has been rejected, while the right side of the inequality can be 0 or 1. When none of the elements in A_2 returns false, based on the assumption in the lemma, that the conditional filtering effect is non-increasing, (5) still holds.

(3) g is self-certifying: we know immediately once g reaches value 1 based on the current observations. Because we are dealing with the case that a scene node will be rejected, our observation space does not contain non-zero possibility events of passing all the constraints checking. So, based on Proposition 9 in [5], function g is a self-certifying instance.

Based on Theorem 11 in [5], when g reaches 1, the average cost of greedy sequence IG is smaller than $(1 + ln(\frac{Q}{\eta}))$ times the optimal time cost. However, in our case, we have value $Q = 1$, and $\eta = 1$, so the cost of IG is equal to the optimal.

A theorem of adaptive strategy is adopted to prove Lemma 1 for our static sequence, because we are dealing with a special case of single label matching ($Q = 1$). If the adaptive strategy continues, it implies that all the observed constraints return true. Under such a scenario, both static and adaptive sequences are the same.

■

Theorem 5.2: Assume the constraint's conditional filtering effect for the same template vertex is non-increasing (as defined in Lemma 5.1) and constraints belonging to different template vertices are independent. i.e. for $i \neq j$, if $\nexists v \in V_T$, s.t. $F_i \in F^v and F_j \in F^v$, then $F_i \perp\!\!\!\perp F_j$. Then the cost of a greedy constraint checking order will be upper bounded by μ times optimal cost, where μ is the maximum number of template vertices that have the same constraint.

The proof is similar to the proof of Theorem 3.4 in [17]. The idea is that for any single template vertex, the expected cost of the optimal sequence should be at least as large as the one returned by a greedy policy, as proved in Lemma 5.1. Since one constraint can appear at most μ times for different template vertices, the cost of the greedy strategy can be at most μ times that of the optimal strategy.

Furthermore, under an arbitrary distribution of constraint responses, the cost of greedy sequence checking is still bounded.

Theorem 5.3: For any distributions of constraints in $\mathbb{F}$, the average time cost of checking constraints with the greedy sequence method is bounded by 4μ times optimal average cost. μ is the maximum number of template vertices sharing the same constraint.

Proof: Similar to the proof of Theorem 5.2, we start with the expected cost of a single template vertex case, then extend it to the general case.

Based on Theorem 2.3 in [11], the average cost of checking constraints following the greedy policy is at most 4 times the optimal cost.

Thereby, when multiple template graph vertices exist and at most μ nodes share the same constraint in $\mathbb{F}$, the greedy sequence checking order is at most 4μ times the optimal expected cost.

■

Our constraints' order determination policy is listed in Algorithm 2, which has a time complexity of $O(n^2)$ where n is the cardinality of $\mathbb{F}$.

VI. Experiments

We apply our target detection framework to two different real-world robotic tasks to validate its generality and effectiveness. The first scenario is to detect a drawer with a handle as shown in Fig 1. Its shape is a cuboid with a handle on the front surface. The second case is to detect a human hand in a pointing gesture. Hand localization is of great interest in the field of Human-Robot Interaction (HRI). In our scenario,

Algorithm 2: Determine the Constraints Checking Order

```
Input:
    The set of constraints: $\mathbb{F}$.
    The cost of checking constraint $F \in \mathbb{F}$ for one node
    $c(F)$
    The distribution of constraints checking results
Output:
    The sequence of the constraints to check:
    $I \in Permutation(1, ..., |\mathbb{F}|)$.
1: Initialize observed set of constraints: $\mathbb{F}_{ob} = \emptyset$,
   ordered list $I$ = empty queue.
2: while $\mathbb{F}$ is not empty do
3:     for all $F \in \mathbb{F}$ do
4:         Compute
           $h(F) \triangleq P(F = false|\forall F' \in \mathbb{F}, F' = true)/c(F)$
5:     end for
6:     Select $F^* \in argmax(h(F))$.
7:     $\mathbb{F}_{ob} = \mathbb{F}_{ob} \cup \{F^*\}$.
8:     $I$ enqueue $F^*$.
9:     Remove $F^*$ from set $\mathbb{F}$.
10: end while
```

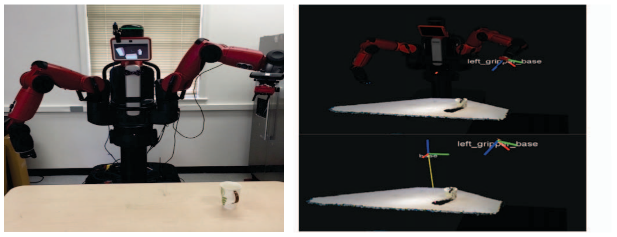

(a) System outlook (b) Illustration of coordinate system

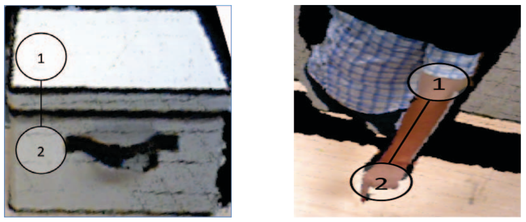
Fig. 2: Robot and camera setup

we consider the hand together with the arm a single target object. This makes the task difficult, because the hand and the arm together no longer form a rigid object. Experimental results show that our framework can still work as long as the target object can be represented as a template graph.

In the phase of determining the constraint checking order, we use $\frac{n_F}{N}$ to approximate $h(F)$ in Algorithm 2, where N denotes the times of checking constraint F, and n_F are the rejected matches between scene and template nodes.

A. Experimental Setup

As shown in Figure 2 (a), we attach an ASUS Xtion PRO camera to the left wrist of a Baxter humanoid robot. Our system maintains and provides transforms between the Baxter base frame and other joints. Since the camera's pose is fixed to the wrist, we calibrate the camera's coordinate to the "left_gripper_base" frame. By propagating the tf (transform) tree, the system projects the point cloud data from the ASUS camera into the robot base frame, which enforces the z-axis to point upwards and x-axis to face forwards (Figure 2 (b)).



(a) Handle box (b) Hand with arm

Fig. 3: Illustration of template graphs

B. Segmentation

An input RGB-D image from the ASUS camera is over-segmented into surfaces to generate a scene graph. In our implementation, we apply the plane-fitting algorithm from [19], which uses depth-adaptive normal calculations and takes into account the noise from the depth measurements. Since the adaptive operations are based on the assumption that the z-axis value is the depth value, we calculate the surface normal and fit a plane in the camera's original frame ("camera_link") before transforming the measurements to Baxter's base frame.

C. Handle Drawer Detection

As shown in Figure 1, a handle drawer has the shape of a box with a horizontal handle on one of its surfaces. Depending on the viewpoint, two (top, front) or three (top, front, and side) surfaces of the box are visible. In our implementation, while maintaining a high success rate, we model the template graph of the drawer as two nodes. Figure 3 (a) shows the template graph. The constraints of the handle drawer template graph are listed in Table I.

Template component	Constraints
Node 1	Size($node_1$), Orientation($node_1$)
Node 2	Size($node_2$), Orientation($node_2$), Has_handle
Edge $(1, 2)$	Pairwise_vertical, Convex_box

TABLE I: Constraints for handle box detection

We check the size constraint by comparing the first two principal components of the surfaces with target-specific thresholds. For example, because node 1 of the handle box is a rectangular surface, we restrict the first principal component (length) to be within $(0.3, 0.7)$ meters and the second dimension (width) to be in the range $(0.25, 0.6)$. For the orientation constraint we check whether a surface's normal (third principal component) is along a particular direction. Here, node 1 in the handle drawer graph is upward while node 2's direction aligns with the horizontal plane. The checking of the surface's handle constraint is done in two steps. First, we extract points within a 3D bounding box in front of the surface patch. Then we validate if there is a connected component on that surface that has the shape of a handle, i.e. satisfying the size constraint of a handle.

For the two edge constraints, the pairwise vertical constraint returns true if the two surfaces are adjacent to each other, and their orientations are perpendicular to each other.

The convex box constraint checking is done by checking if two surfaces not spanning the same plane form a convex shape, which means the two surfaces shall not segment each other, and the shape they form must form a convex box instead of a concave corner, as illustrated in Figure 4.

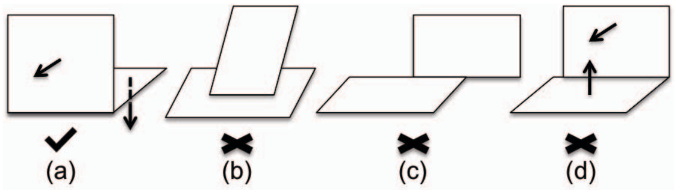


Fig. 4: (a) Illustration of a convex box. Three failure cases: (b) top surface segments the bottom one; (c) not enough shared boundary; (d) two surfaces form a corner (faces towards)

D. Hand Pointing with Arm Detection

As second scenario we consider hand detection. Here we show an application of detecting the hand together with part of the arm visible in the scene. Figures 3 (b) and 7 show the experimental setup. Note that the target object is no longer rigid in this situation because the angle between the finger and the wrist is flexible during pointing.

As shown in Figure 3 (b), we use one node to denote the arm in the template graph and the other for the hand. The constraints for checking template graphs are listed in Table II.

Template component	Constraints
Node 1	Size($node_1$), Location($node_1$)
Node 2	Location($node_2$), size($node_2$), Hand_shape
Edge $(1, 2)$	Hand_arm_relationship

TABLE II: Constraints for pointing hand detection.

The size constraint is handled in the same way as the handle drawer detection except that the size thresholds need to be set for surfaces that belong to the hand and the arm. The location constraint returns true if the centroid of a scene surface is in a given cuboid area. Since we have already transformed the point cloud to the Baxter's base frame, which aligns well with human perception, it is not hard for people to manually annotate a 3d range of possible locations of the target. For example, because we do not expect to see the arm or the hand on the ground or flying high around the ceiling in this scenario, we can set the location threshold on the z-axis to reject surfaces heights that are too large or too small. The hand shape constraint checks if the contour of a surface patch has the shape of a pointing hand.

The hand-arm relationship is encoded as edge relationship between the hand and the arm. We check if there is a hand node close to the arm node and enforce the constraint that the hand is along the direction of the arm.

E. Optimization of the constraints checking order

As discussed in Section V, the order of checking the constraints influences the time for target detection, and we proved that the greedy constraint ordering algorithm (algorithm 2) has a bounded computational cost w.r.t the optimal computational cost.

To show the efficiency of the presented algorithm, we compare the average computational time for constraint checking (line 2-15 in algorithm 1) of four different checking order policies: random checking order, best order among sampling 300 sequences, the greedy order proposed, and the optimal order.

Random ordering does not need any training data and it represents the most naive policy. The time cost of random order is the mean of the running time of 300 random checking sequences. The order determined by sampling has minimum running time among 300 random checking sequences on the training data. The optimal order is obtained by exhaustively trying all possible sequences and then selecting the sequence with minimal cost as the optimal one. The handle drawer has $7! = 5040$ possible sequences and the hand detection has $6! = 720$. Note that an exhaustive search for the optimal sequences is not feasible because the number of permutations grows exponentially when the number of constraints increases.

For both, handle box and hand detection, we collected 50 point clouds each for training and testing. Also, another 100 background point clouds were collected to serve as negative samples with one half used in training and the other half in testing. The result is shown in Table III.

time (ms)	Random	Sampling	Algorithm 2	Optimal
Handle box	382.6	28.0	24.6	19.3
Pointing hand	43.2	8.1	8.8	8.1

TABLE III: Running time of four constraint checking order on testing data.

The machine used in the experiments has an Intel i7-6700 CPU of 3.4GHZ and the memory is 16 GB. We did not use GPU or parallel computing.

From the result, we can see that a random ordering without any optimization would take the longest time to execute. Our proposed greedy algorithm is close to the optimal order. When the total number of possible sequences is not large, a sampling method would perform better than our method as is shown by the case of the pointing hand. But when the possible number of sequences is large so that sampling cannot cover a reasonable portion, our algorithm works better as is shown for the case of the handle box. Also note that to determine a checking order, algorithm 2 is much faster than the sampling method because algorithm 2 only needs 28 and 21 sequence checking runs while sampling needs 300 in our experiment.

VII. A LIVE HRI APPLICATION

In this section we describe an application of human-robot interaction that is built on the pointing hand detection discussed in section VI.

Our system allows humans to interact with real world objects through pointing gestures, and it then generates a command for the robot accordingly. For instance, let us suppose that a user intends to heat an object using the microwave or put an object into the refrigerator. As shown in Figure 7, using our system, a user can select the target object using a pointing gesture. After the selection is confirmed, the target object can be virtually dragged to its target location through human guidance.

After applying the constraint checking order optimization, our hand detection works faster (than a naive approach), thence our system provides a smoother interaction user experience. To complete the whole scenario, we introduce other components of the interaction system aside from hand detection in the following sections. We provide a video showing the interaction process in the supplementary material.

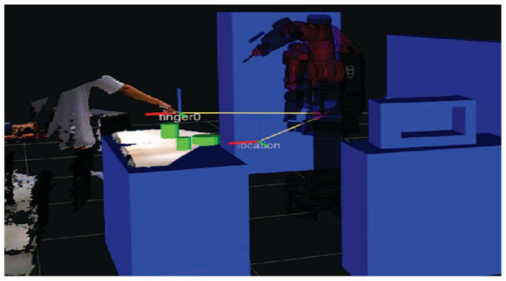


Fig. 5: Visualization of the fingertip and its pointing location.

A. *Surrounding objects recognition*

Large surrounding objects, such as a fridge, a table and the shelf holding the microwave can be detected at the beginning of the process and are kept in a stored world model since they will be static for a long time. The table top objects can be detected and recognized online by a tabletop point cloud reconstruction, segmentation [3] and recognition. The recognition outputs are stored in the same world model describing the real world configurations. It is worth mentioning that our pipeline is applicable for both the tabletop objects and detection of other objects. For example, the microwave in Figure 7 is detected using the same pipeline.

B. *Obtaining the object that the hand is pointing at and its location*

We use a straight line to represent the pointing direction after detecting the pointing hand and arm. The direction of the line is the detected arm's direction, which is calculated by taking the eigenvector corresponding to the first principal component of the point cloud belonging to the arm. The fingertips are at the starting point of the straight line. We detect the fingertips by searching the 3D points on the hand that are furthest along the pointing direction. Figure 5 shows a visualization of a fingertip point.

During the object selection phase, we simply treat the object closest to the pointing line as the target object of the user. Also a threshold is used to limit the distance between the object and the pointing ray.

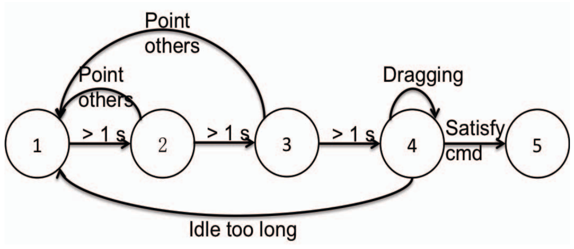


Fig. 6: An illustration of the state machine for human-robot interaction. State 1: initial state; state 2,3: state for object pointing confirmation; state 4: object virtual moving state; state 5:ending state after parsing the command successfully.

After an object is confirmed, the user can start moving it around by pointing to other locations in our virtual environment. At this time, we treat the intersection point between the pointing ray and a plane spanning table top as the target location (Figure 5). Thence, the selected object's position can be updated as the new location.

C. *Robot feedback interface*

A proper way to display the current configuration of the world model and the status of pointing is necessary for a smooth and accurate interaction between human and robot.

In our system, we directly show the current status of the system through the Baxter robot's screen. Specifically, we visualize the virtual world using the PCL visualizer [22].

D. *Interaction flow finite state machine*

The underlying logic flow is implemented as a finite state machine as shown in Figure 6.

At the beginning, the system is in state 1 waiting for the human's command (Figure 7 (a)). The system will advance to the object selection state 2 once a pointing to the object happens for more than 1 second and the selected object will turn red (Figure 7 (b)). After two more seconds of consistent pointing, the object is confirmed to be chosen and the system reaches state 4 (7 (d)). Then the object can be virtually moved around. Finally, when the object reaches a certain area of the target location for more than 1 second, the system gets a command that it was successful and enters the final state 5 (Figure 7(e)).

VIII. Future Work

As shown by experiments and an HRI application, our framework is able to detect target objects in robotic applications in a reliable and effective way. In future work, we plan to improve the learning pipeline by generating template graphs automatically. Additionally, with the number of types of constraints increasing, how to select a reasonable subset is another problem that deserves further investigation.

In the application scenario outlined we make a first step to teaching the robot via pointing to the target area. Using this approach of teaching the robot with bare hands, trajectory learning and adaptation [16] could become more friendly.

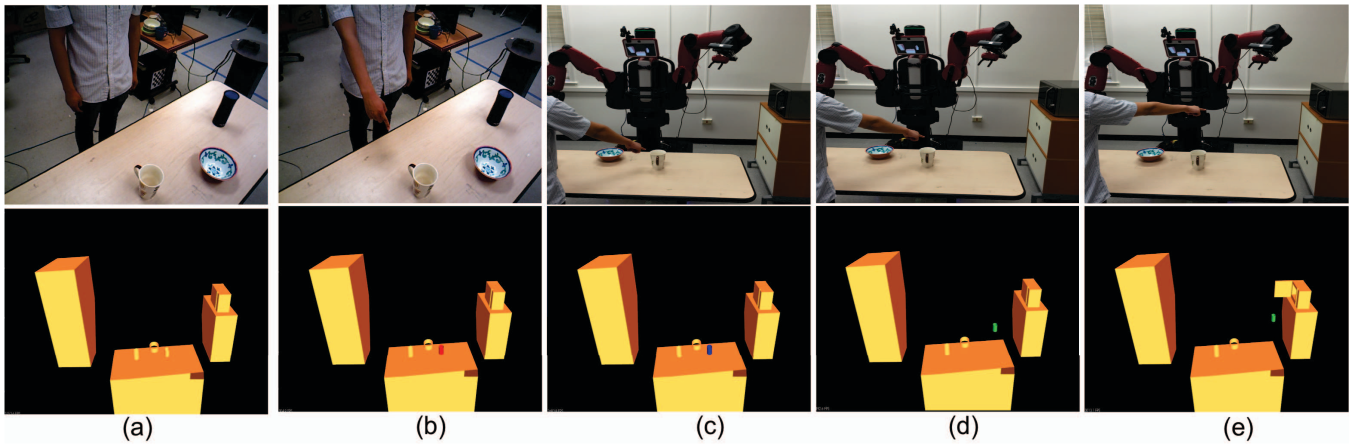


Fig. 7: Sending a command: <heat the mug> via HRI system. The top row shows the human operation; the bottom row shows the status of the interaction process. (a) Initial stage with objects detected in the scene; (b)(c) target object selection by pointing and object confirmation; (d) the selected object is virtually dragged to another functional place, i.e. microwave; (e) the system receives the command.

IX. Acknowledgement

This work was funded by the support of DARPA (through ARO) grant W911NF1410384, by NSF through grants CNS-1544787 and SMA-1540917, and the University research program of Northrop Grumman.

References

[1] Aitor Aldoma, Federico Tombari, Luigi Di Stefano, and Markus Vincze. A global hypotheses verification method for 3d object recognition. In *Proceedings of the 12th European Conference on Computer Vision*, ECCV'12, Berlin, Heidelberg, 2012.

[2] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *International Conference on Computer Vision & Pattern Recognition*, June 2005.

[3] Aleksandrs Ecins, Cornelia Fermüller, and Yiannis Aloimonos. Cluttered scene segmentation using the symmetry constraint. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[4] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition*, 2014.

[5] Daniel Golovin and Andreas Krause. Adaptive submodularity: A new approach to active learning and stochastic optimization. *CoRR*, 2010.

[6] Abel Gonzalez-Garcia, Alexander Vezhnevets, and Vittorio Ferrari. An active search strategy for efficient object detection. *CoRR*, 2014.

[7] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary R. Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *11th Asian Conference on Computer (ACCV) Vision, Daejeon, Korea, 2012*.

[8] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1998.

[9] Dinesh Jayaraman and Kristen Grauman. Look-ahead before you leap: end-to-end active recognition by forecasting the effect of motion. *CoRR*, 2016.

[10] Asako Kanezaki, Zoltan-Csaba Marton, Dejan Pangercic, Tatsuya Harada, Yasuo Kuniyoshi, and Michael Beetz. Voxelized Shape and Color Histograms for RGB-D. In *International Conference on Intelligent Robots and Systems (IROS)*, San Francisco, USA, 2011.

[11] Haim Kaplan, Eyal Kushilevitz, and Yishay Mansour. Learning with attribute costs. In *Proce.Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC '05, New York, NY, USA, 2005. ACM.

[12] H. Li, Y. Li, and F. Porikli. Deeptrack: Learning discriminative feature representations online for robust visual tracking. *IEEE Transactions on Image Processing*, April 2016.

[13] X. Liu and J. S. Baras. Trust-aware crowdsourcing with domain knowledge. In *2015 54th IEEE Conference on Decision and Control (CDC)*, Dec 2015.

[14] Wentao Luan, Ren Mao, and John S. Baras. Active sampling exploiting detector response pattern for efficient target detection. In *2016 19th International Conference on Information Fusion (FUSION)*, July 2016.

[15] Wentao Luan, Yezhou Yang, Cornelia Fermüller, and John S. Baras. Reliable attribute-based object recognition using high predictive value classifiers. In *European Conference of Computer Vision*, 2016.

[16] R. Mao, Y. Yang, C. Fermüller, Y. Aloimonos, and J. S. Baras. Learning hand movements from markerless demonstrations for humanoid tasks. In *2014 IEEE-RAS International Conference on Humanoid Robots*, 2014.

[17] Kamesh Munagala, Utkarsh Srivastava, and Jennifer Widom. Optimization of continuous queries with shared expensive filters. In *Proceedings of the Twenty-sixth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '07, New York, NY, USA, 2007. ACM.

[18] M. Nishigaki, C. Fermüller, and D. DeMenthon. The image torque operator: A new tool for mid-level vision. In *Computer Vision and Pattern Recognition (CVPR),*, June 2012.

[19] A. Richtsfeld, T. Mrwald, J. Prankl, M. Zillich, and M. Vincze. Segmentation of unknown objects in indoor environments. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4791–4796, Oct 2012.

[20] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. In *IEEE Int. Conf. on Robotics and Automation*, pages 3212–3217, May 2009.

[21] Radu Bogdan Rusu, Gary Bradski, Romain Thibaux, and John Hsu. Fast 3d recognition and pose using the viewpoint feature histogram. In *Proceedings of the 23rd IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, 2010.

[22] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[23] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 2013.

[24] Ziang Xie, Arjun Singh, Justin Uang, Karthik S. Narayan, and Pieter Abbeel. Multimodal blending for high-accuracy instance recognition. In *Proceedings of the 26th IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[25] Xiaodong Yu, C. Fermüller, Ching Lik Teo, Yezhou Yang, and Y. Aloimonos. Active scene recognition with vision and language. In *2011 International Conference on Computer Vision*, Nov 2011.

[26] Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu. Mining and-or graphs for graph matching and object discovery. In *The IEEE International Conference on Computer Vision (ICCV)*, 2015.