# Interactive Segmentation, Tracking, and Kinematic Modeling of Unknown 3D Articulated Objects

Dov Katz, Moslem Kazemi, J. Andrew Bagnell and Anthony Stentz[1]

***Abstract*— We present an interactive perceptual skill for segmenting, tracking, and modeling the kinematic structure of 3D articulated objects. This skill is a prerequisite for general manipulation in unstructured environments. Robot-environment interactions are used to move an unknown object, creating a perceptual signal that reveals the kinematic properties of the object. The resulting perceptual information can then inform and facilitate further manipulation. The algorithm is computationally efficient, handles partial occlusions, and depends on little object motion; it only requires sufficient texture for visual feature tracking. We conducted experiments with everyday objects on a robotic manipulation platform equipped with an RGB-D sensor. The results demonstrate the robustness of the proposed method to lighting conditions, object appearance, size, structure, and configuration.**

## I. Introduction

We propose an interactive perceptual skill for the manipulation of unknown objects that possess inherent degrees of freedom (see Fig. 1). This category of objects—rigid articulated objects—includes many everyday objects ranging from pliers, scissors and staplers, to windows, doors, drawers, books, and toys (see Fig. 2 for some examples). To manipulate an articulated object, the robot must be able to deliberately change its configuration. For example, to manipulate a door, the degree of freedom (hinge) connecting the door to its frame is actuated.

In this paper, we develop the necessary interactive perceptual capabilities to segment an object out of its cluttered background, track the object's position and configuration over time, and model its kinematic structure. These capabilities are fundamental for purposeful manipulation.

We contend that perception and manipulation fundamentally cannot be separated: it is difficult, if at all possible, to visually segment an unknown object and determine its kinematic structure; equally, it is difficult to manipulate a complex object without continuous sensing feedback. We refer to the combined process as "Interactive Perception" (Fig. 1). In interactive perception, the robot manipulates the environment in order to assist the perceptual process. Perception, in turn, provides information necessary for successful manipulation. Interactive perception enables the robot to perceive the outcome of its interactions, and therefore to interpret the sensor stream within the context of its actions.

In the following, we rely on interactive perception to generate relative motion, a strong visual signal. This relative motion reveals properties of the environment that would otherwise remain hidden. In our case, interaction reveals the degrees of freedom of objects.

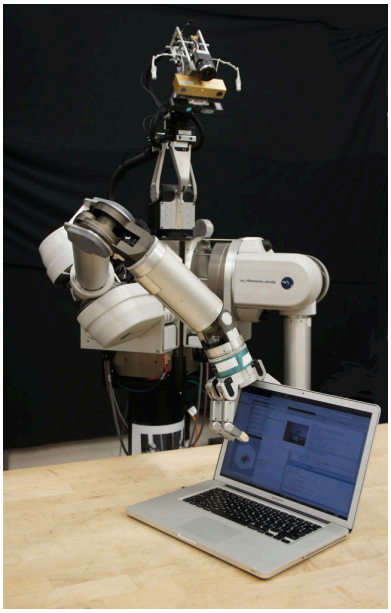


Fig. 1: Our manipulator interacts with an unknown articulated object (laptop). The proposed interactive perceptual skill segments the object, tracks its position, and models its kinematic structure.

Our perceptual skill relies on an RGB-D sensor. This sensor co-registers color and depth images (e.g. structured light sensors or co-registered LADAR and cameras). It provides our robot with the visual information necessary to model new objects. We complement this visual information with interaction. In this paper, the robot's interactions with the environment are scripted. These interactions can also be random (not efficient), or better: learned from experience [7]. Our skill assumes no prior object models. It is insensitive to changes in lighting conditions, object size, configuration and appearance. It can model objects with an arbitrary number of degrees of freedom. It is also computationally efficient. Additionally, it allows for objects to be occluded by the manipulator during the interaction. The algorithm relies on only two assumptions: objects have enough texture to enable feature tracking, and the robot is capable of exciting the object's degrees of freedom—a reasonable assumption given that our goal is to acquire information for manipulating the object.

Our algorithm outperforms the state of the art in several aspects: it is 3 orders of magnitude faster, more accurate, simpler to implement, and does not require any tuning. These improvements are essential if a robot is to use this skill in

[1]The authors are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA

Fig. 2: Objects that possess inherent degrees of freedom: drawer, train toy, cd box, book, checkerboard, papertowel, stroller, folder, and window. These degrees of freedom are closely related to the function of the object. Without prior knowledge, the degrees of freedom of an object cannot be extracted from visual information alone. They have to be discovered through interaction.

real-world manipulation tasks.

In the following, we describe our interactive perceptual skill in detail. In Section II we discuss related work and argue that our approach outperforms prior work (including our own) in speed, robustness, and accuracy. In sections III—VI, we provide the details of our implementation. Then, in section VII, we present experiments that demonstrate our robot's ability to segment, track and model the kinematic structure of everyday objects.

## II. Related Work

Our perceptual skill is composed of two important steps: segmenting the distinct rigid bodies in the scene, and determining the kinematic constraints between these bodies. We now review the methods and techniques that are most relevant to each of the two steps.

### A. Object Segmentation

To manipulate an object, a robot must be able to segment it out of a cluttered scene. Indeed, image segmentation has been an ongoing challenge for the computer vision community for over three decades. In image segmentation, boundaries around image regions with consistent properties are identified [4]. Existing segmentation methods typically analyze a single image, identifying discontinuities in color, brightness, texture, or depth [1], [4]. However, as objects may be multi-colored, multi-textured, and possess internal depth discontinuities, the regions computed by image segmentation methods rarely correspond to object boundaries.

Manipulation provides grounded semantics for segmentation: an image region should correspond to a single rigid body. A rigid body is defined by the fact that the distance between any two points on the body remains constant in time regardless of external forces exerted on it. It follows that in order to segment an unknown object, without assuming prior knowledge, one must observe the object in motion. A new generation of interactive segmentation algorithms leverages this insight that relative motion can greatly simplify image segmentation. These algorithms [3], [8] rely on the robot's body to generate motion, thereby revealing a visual signal which can be analyzed to segment the moving rigid bodies. Our approach too relies on interaction to solve the image segmentation problem.

### B. Modeling Kinematic Constraints

The problem of acquiring kinematic models from sensor data is fundamental for autonomous manipulation, yet only recently researchers have started to address this problem. Most existing approaches assume prior knowledge about the modeled object. Our goal is manipulating unknown objects. Therefore, we focus on those methods that do not require prior object models.

Yan and Pollefeys [14] rely on structure from motion to extract 3D feature trajectories from a 2D camera, and then use spectral clustering to identify rigid bodies and their kinematic relationship. This work assumes affine geometry of the scene and only considers revolute joints. Ross et al. [11] also rely on structure from motion, but use maximum likelihood estimation to cluster features into rigid bodies. The strength of these two algorithms is that they can handle bodies that undergo slight deformation during motion. However, both approaches only handle revolute joints and make strong assumptions to simplify the perception problem. They are computationally very expensive and require large relative motions. Therefore, the above methods are not practical for autonomous manipulation.

Katz et al. [5], [7] extract kinematic models of planar articulated objects. This work relies on the insight discussed in section II-A: deliberate interaction are used to generate relative motion, which in turn enables segmentation and joint detection. This approach is computationally efficient and can handle an arbitrary number of rigid bodies and degrees of freedom, but is limited to planar objects.

Sturm et al. [12], [13] learn models of kinematic joints from three-dimensional trajectories of a moving plane. Motion is generated from deliberate interaction with the environment. The strength of this approach is that it can model arbitrary joint types. However, it requires that only a single rigid body is moving at a time. It is also limited to objects that can be approximated as a moving plane, such as drawers and doors.

In our own prior work [6], we have extended the interactive perception approach initially developed in [5], [7] for planar objects to handle general 3D articulated objects. The algorithm in [6] requires that objects have sufficient texture for feature tracking. It relies on structure from motion to recover depth information, and therefore depends on large relative motions. It is also prohibitively slow for autonomous manipulation, averaging 15 minutes per object, and, similar to all of the above methods, it does not handle occlusion during interaction. Finally, the algorithm in [6] depends on several hand-tuned parameters. Nevertheless, we consider the work in [6] to be the precursor of the method we present in this article.

Our algorithm proposes an efficient and simple solution that is practical for autonomous manipulation. In our experiments, the runtime never exceeded 50 milliseconds (about 3 orders of magnitude faster than the performance of [6]). Furthermore, our method can model objects in the presence of occlusion during interaction—an important property as

interaction for perception is unlikely to be gentle and precise. In addition, our approach acquires 3D measurements from the sensor. Thus, we do not need to reconstruct depth from motion, and therefore require little motion to segment and model unknown objects. Finally, our algorithm depends on a single parameters: the amount of relative motion necessary to separate two points into two different rigid bodies. This parameter is directly related to the physical properties of our RGB-D camera.

## III. Perceiving Unknown Objects

The proposed interactive perceptual skill is composed of three steps. The first step collects perceptual information that provides the input to our algorithm. In this step, we initialize and track visual features throughout the robot's interactions with the environment. We offer two complementary implementations for this step using either the simple KLT feature tracking or SIFT matching. The second step analyzes the trajectories of these features, and computes a clustering of features and their trajectories into rigid bodies. And the third step of our algorithm determines the kinematic constraints between pairs of rigid bodies.

## IV. Collecting Perceptual Information

The first step of our algorithm collects perceptual information. We rely on the robot's manipulation capabilities to create object motion. By physically causing objects to move, the robot generates a strong perceptual signal for object segmentation. Here, the robot's motion is scripted. However, this restriction can be removed by learning interaction strategies, as demonstrated in [7].

Once motion is induced, the robot collects perceptual information by tracking visual features using an RGB-D sensor. Thus, for every tracked feature, we have both color and depth information. Our perceptual skill uses one of two types of visual features and corresponding tracking algorithms. The first is the Lucas-Kanade (LK) feature tracker, using Shi-Tomasi corner features [10]. The second is based on matching scale invariant features (SIFT) between keyframes [9].

The LK tracker assumes that brightness remains consistent between the same pixels from one frame to the next, and that only small movements occur between frames. The tracker estimates the velocity of a moving point feature by the ratio of the derivative of the intensity over time divided by the derivative of the intensity over space, and the resulting algorithm can track in real-time a large number of features. However, the LK feature tracker poorly handles occlusion: if the robot's arm blocks parts of the scene, all features that are temporarily invisible are lost or become unreliable.

SIFT features [9] overcome this limitation of the LK tracker. These features describe interesting points in an image so as to be robust to changes in scale, noise, and illumination enabling features to be matched between images despite significant changes in position. Using SIFT features provides robustness against occlusion: a feature that cannot be found in a certain frame due to occlusion is not declared lost; it may be detected in a later frame. Computing SIFT features descriptors and matching across frames is significantly slower compared to the LK tracker, so our implementation matches SIFT features in a sparse sequence of images containing $10\%$ of the frames used by the LK tracker. The ability to use SIFT features is due to the fact that our algorithm exploits the depth information provided by the RGB-D sensor. Thus, we do not have to recover structure from motion, and do not require continuous feature tracking. This improvement allows our algorithm to be robust against partial occlusions.

Throughout its interaction with the environment, the robot tracks a set of features. The robot records the features' images coordinates $(u, v)$, 3D coordinates $(x, y, z)$, and color intensities $(r, g, b)$ for each time $t$: $f_i(t) = \{u, v, x, y, z, r, g, b\}$. This process of collecting perceptual information is identical, regardless of whether features are generated by the LK tracker or SIFT matching.

Both methods for feature tracking make no assumption about the shape, size, or color of objects, about their motion, or the motion of the camera, and only require that the scene contains sufficient texture. However, both SIFT and LK are highly unreliable in unstructured scenes. LK features can jump between image regions, be lost, swapped, or drift along edges in the image. And SIFT features can be wrongly paired. The next step of the algorithm will automatically eliminate this noisy data.

## V. Segmenting Rigid Bodies

The second step of our algorithm computes rigid-body segmentation. To segment an unknown scene into rigid bodies, we leverage the following definition: "The distance between any two given points of a rigid body remains constant in time regardless of external forces exerted on it." Thus, segmenting a scene into rigid bodies becomes the task of clustering the tracked features from the previous step into subsets of features that maintain constant relative distance throughout the robot's interaction with the environment.

For efficiency, we first extract all features that remain static throughout the interaction. These features belong to a single "background" cluster. Next, we cluster the remaining features using a greedy algorithm. We pick an unclustered feature, and assign to it a new cluster ID. We then compute the relative motion between this feature and each of the remaining unclustered features. The relative motion is defined to be the difference between the maximal and minimal distance between the two features throughout the interaction. If this difference is smaller than some $\epsilon_{motion}$, we assign the same cluster ID to the second feature. Note that if the current cluster already contains more than one feature, we require that the newly added feature maintains a relative motion of less than $\epsilon_{motion}$ with all features in the cluster.

The computational complexity of this greedy algorithm is easy to determine. For $n$ features our algorithm will do at most $O(n^3)$ relative motion computations. Each relative motion computation requires computing the 3D distance between two points in $k$ frames. Together, our algorithm's complexity is $O(kn^3)$. In our experiments, interactions with

the environment are about $k = 150$ frames long (the equivalent of 10 seconds of video) and the robot is typically tracking about $n = 300$ features.

We set $\epsilon_{motion}$ to be $2cm$. This value was chosen based on the physical properties of our RGB-D camera (Microsoft Kinect), and considering the range in which objects must be for the robot to interact with them while observing the outcome. Specifically, for objects at a distance of about $2m$, and given the Kinect's accuracy of up to $\frac{1}{8}$ of a pixel, we can expect a quantization error of about $2cm$: $z = fb/d$ where $z$ is the measured depth value, $f = 575$ pixels is the Kinect's focal length, and $b = 0.075m$ is the Kinect's baseline. For example, a disparity $d$ of 20 pixels corresponds to $z = 2.156m$, and a disparity $d = 20.125$ pixels corresponds to $z = 2.142$, a difference of $0.015m$ due to quantization.

Our rigid body segmentation step has four main advantages: First, our runtime performance is about 3 orders of magnitude faster than that of [6], rendering our approach practical for real-time applications. Second, our algorithm does not make restrictive assumptions about the type or number of joints, does not assume that objects can be approximated as planes, it allows for multiple parallel motions, and requires only very little relative motion (Fig. 3). Third, it requires no tuning, and only relies on a single parameter $\epsilon_{motion}$ which is sensor specific and has a clear physical interpretation. And finally, clustering features into rigid bodies works even in the presence of partial occlusion by computing SIFT matching and only considering relative distance changes for frames in which both features are visible.

## VI. Detecting Kinematic Constraints

The third step of the algorithm determines the kinematic structure of an object. The input to this step is a set of rigid bodies, each represented by a cluster of point features and their trajectories. Our goal is to determine, for every pair of rigid bodies, whether they are connected by a revolute joint, a prismatic joint, or are disconnected.

The kinematic relationship between a pair of rigid bodies is determined by their relative motion. Fig. 4(a) and 4(b) show the trajectories of two sets of point features, each associated with one side of a checkerboard. Because the two bodies move together, while also rotating with respect to each other, it is difficult to determine that they are connected by a revolute joint. Removing the common global motion, however, provides very clear evidence for the revolute joint (Fig. 4(c)).

To extract the relative motion between pairs of rigid bodies, we first compute the relative homogeneous transformation ${}^0H_k$ for one rigid body between its initial pose at time $t_0$ and its pose at every time $t_k$ along its trajectory. To handle measurement noise, we find the optimal transformation by solving a least square minimization problem using singular value decomposition (SVD) [2]. We obtain the relative motion trajectories between the bodies by applying the transformations ${}^0H_k$ computed for the first body to the feature trajectories of the second body (Fig. 4(c)).

The relative motion trajectories between two rigid bodies reveal their kinematic relationship. If the bodies are connected by a prismatic joint, the relative motion will have the form of straight, parallel lines of equal lengths. Our algorithm calculates the lengths of all relative motion trajectories. If the lengths are approximately equal, we use RANSAC line fitting, and calculate a fitness measure for each relative motion trajectory. A good fit (above 80% in our experiments) indicates a *prismatic* joint (Fig. 5).

If the prismatic fitness measure is low, we check whether the relative motion can be explained by a *revolute* joint. Again, we use RANSAC, but this time to fit a circle to each trajectory. We then fit a line through the centroids of these circles. If the line is close to the majority of the centroids, the relative motion is due to a revolute joint, and the line is its axis of rotation (Fig. 4(c)). Otherwise, we declare the two bodies to be *disconnected*. Figure 4(d) shows a complex motion between one side of the checkerboard and the static background. As expected, the trajectories cannot be explained by a single degree-of-freedom joint.

Our kinematic constraints detection step has two advantages: First, it can be applied to any type of articulation model by fitting the appropriate motion model to the relative motion trajectories. We provide implementation for all single degree of freedom joints, but that implementation can easily be extended to other joint models. Second, our algorithm does not require a structure from motion step. Therefore, we do not have to reason about the relative scale in which pairs of rigid bodies are reconstructed (see [6] for detailed explanation). The result is a simpler, faster and more accurate kinematic modeling.

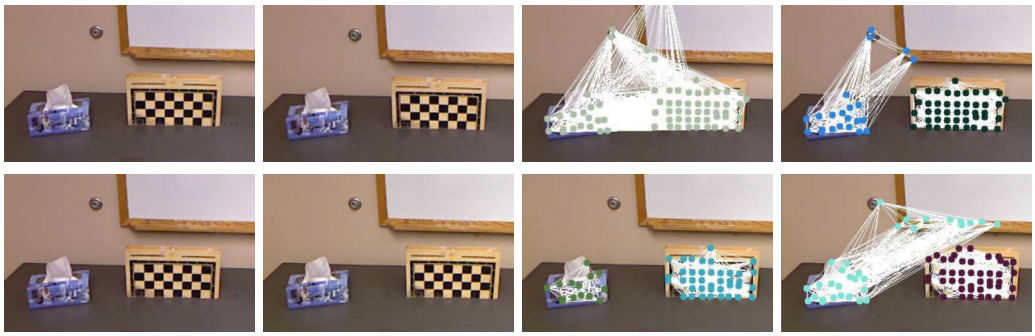
## VII. Experimental Validation

We validate the proposed perceptual skill in real-world experiments. In our experiments, we used a cheap off-the-shelf RGB-D sensor, the Microsoft Kinect. We have conducted dozens of experiments with many different objects (e.g., the objects in Fig. 2, 6, and 7). Our goal is to demonstrate that the robot can segment the target object and its rigid parts, track the position of each rigid body over time, and identify the kinematic relationship between rigid bodies.

Our experimental validation is divided into three parts: First, we analyze the output of the three steps of the algorithm on four representative objects. Second, we present experimental results showing that the algorithm's runtime greatly outperforms the state of the art. And third, we analyze the robustness and accuracy of tracking the configuration of modeled objects (the position, orientation, and configuration of the intrinsic and extrinsic degrees of freedom).

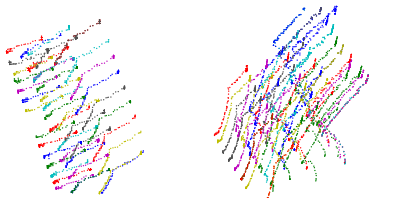
### A. Modeling Articulated Objects

Figure 6 illustrates the output of the three steps of our algorithm for four articulated objects: a laptop, a door, a set of drawers, and a firetruck. Each row corresponds to one of the four objects. Columns one and two show the object before and after the interaction. Column three shows the output of the clustering of tracked features into rigid bodies (clusters

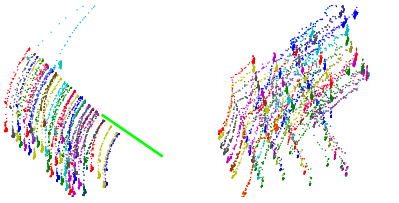


(a) Initial pose (b) Translation of 1cm (top) and 4cm (bottom) (c) [6]'s Segmentation (d) Our Segmentation

Fig. 3: Comparing our segmentation performance and [6]. Our method requires smaller displacements, and therefore separates the board from the background after a translation of only $1cm$, compared to $4cm$ for [6]. Also, our algorithm is over 15000 times faster (28msec. vs. 8 min)



(a) Left board motion (b) Right board motion



(c) Relative motion between the boards: revolute joint detected (d) Relative motion board-background: declared disconnected

Fig. 4: Feature trajectories for a checkerboard. The two sides of the board move with respect to the background, and rotate with respect to each other.

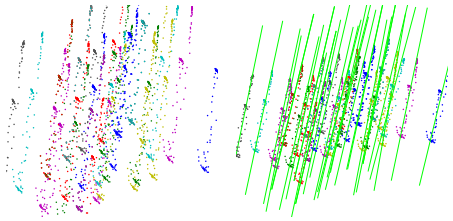


Fig. 5: Left: motion trajectories of a drawer. Right: successful RANSAC line fitting to the relative motion.

are color coded; for clarity static background features are excluded). Columns four and five show the output of the joint detection step. We choose these examples to show that our method can be applied to different joint types (prismatic and revolute), different joint orientations, different number of degrees of freedom, and overall a variety of objects. We will now discuss the output of each of the four experiments in detail.

In the first experiment, the robot interacts with a set of two drawers by closing them. The segmentation into rigid bodies is accurate. The green cluster corresponds to the top drawer and the red cluster to the bottom drawer. Joint detection assigns high confidence values to a prismatic joint between the background and each one of the drawers, and the orientation of the prismatic axes is correct. Our algorithm works in this case because it can detect and analyze any number of rigid bodies, even when moving simultaneously.

In the second experiment, the robot interacts with a toy laptop by pushing it on the table and actuating the intrinsic degree of freedom connecting the keyboard to the monitor. The segmentation into rigid bodies is correct. Joint detection assigns high confidence values to a revolute joint between the parts of the laptop, and the two laptop parts are detected to be disconnected from the static background. The position and orientation of the revolute joint is correct. We note the difference in type of joint, scale, appearance and motion between the drawers and the laptop.

The third experiment shows an interaction with an office door. Here too, a revolute joint is accurately detected between the door and its frame. The door has a revolute joint, just like the laptop. However, the orientation of the joint is different. We also notice the significant differences in appearance and size. For both experiments, only very little motion is required for detecting the revolute joint.

The last experiment we analyze shows an interaction with a firetruck toy. This is an interesting experiment. The truck translates along the table, while the two wheels rotate. We expect to find a revolute joint connecting the wheels to the frame of the truck. This is more challenging than the previous three experiments as the perceived motion of the wheels is composed of both translation and rotation at the same time. The clustering step provides very good results. The three clusters correspond to the back wheel (yellow), front wheel (blue), and truck frame (red). We also note that some red features are present on the wheel, right where we expect the axis to be. The algorithm accurately detects the two revolute joints.

The results presented in Fig. 6 demonstrate the performance of the different steps of our algorithm on four

Fig. 6: Experimental results showing the process of segmenting a scene into its individual objects. Left to right: The first column shows the object before the interaction; the second column shows the object after the interaction; the third column shows the results of clustering the tracked features into rigid bodies; the fourth and fifth columns show the results of joint detection obtained for the first two rows.

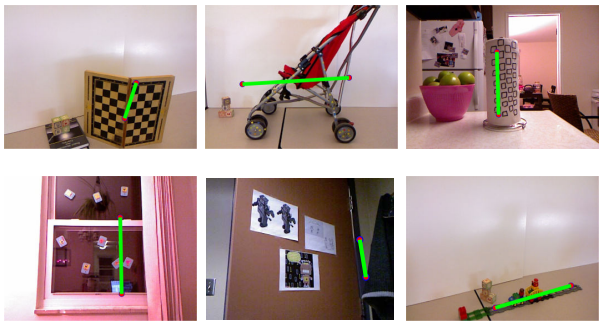


Fig. 7: Additional experimental results showing the detected joint for more everyday objects: a checkerboard (revolute), a door (revolute), a paper-towel (revolute), a stroller (prismatic), a toy train (prismatic), and a window (prismatic). The detected joints are marked in green.

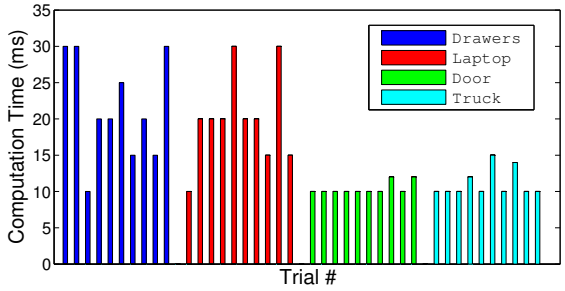
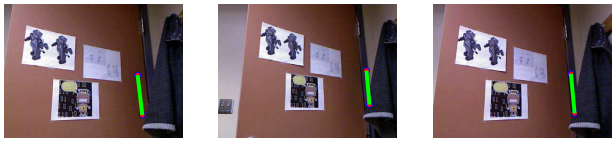



Fig. 8: Runtime of our algorithm in milliseconds with four objects (drawers, laptop, door, and firetruck). Each experiment was repeated 10 times. The configuration of the object changes between trials.

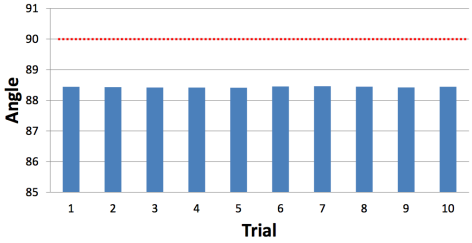
objects. We choose this set of objects because it includes objects with one degree of freedom (door, laptop) and several degrees of freedom (drawers and firetruck), small objects (laptop) and large objects (door), prismatic joints (drawers, firetruck) and revolute joints (laptop, door, firetruck), and interesting variation in colors and shape. This is, however, just a representative set of results. We conducted many more experiments. We include a few more examples, without the detailed analysis, in Fig. 7.

### B. *Run time Analysis*

One of the important contributions of our algorithm is its near real-time performance. We evaluated the runtime of our algorithm on a desktop computer with a 3.3GHz processor and 8GB of RAM. For each one of the ten experiments conducted with the drawers, laptop, door and truck we measured the execution time. The results are shown in Fig. 8. None of our experiments took more than $30ms$, most required less than $20ms$. While this is only an estimate of the actual run time, it shows that our method is very fast and can practically be applied every few frames. This represents a significantly superior performance compared to the method in [6]. A direct comparison with [6] (see Fig. 3)



(a) Detecting a door's hinge (10 trials, conditions vary)




(c) Detecting a drawer's prismatic joint (10 trials, conditions vary)

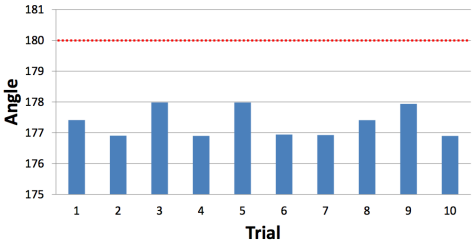



Fig. 9: To demonstrate repeatability we conducted experiments with two objects: door and drawer. Each experiment was repeated 10 times, while changing lighting, sensor position and orientation, and initial and final object configuration. Three examples of the detected joint (marked in green) are shown for each object. The plots show the angle between the detected axis and the floor. Door: mean = $88.43^\circ$, std. dev. = 0.016, Drawer: mean = $177.3^\circ$, std.dev. 0.94. Results are close to expectations ($90^\circ$ and $180^\circ$). The difference ($\leq 3^\circ$) is due to misalignment of the sensor with the floor.

reveals a performance boost of three orders of magnitude. This improvement is due to our greedy clustering algorithm. It is simple and very efficient, while not compromising correctness or accuracy.

### C. Repeatability

To verify the repeatability of our algorithm, we conducted all the experiments in this paper 10 times. Between experiments we varied the lighting conditions, the position and orientation of the sensor, and the configuration of the object. We now analyze two representative examples: a door and a set of drawers.

Figure 9 shows three instances of the two experiments (out of 10). The detected joint is marked in green. For each experiment, we computed the angle between the detected joint axis and the floor. For the door, we expect that angle to be $90^\circ$ (axis perpendicular to the floor) and for the drawers $180^\circ$ (axis parallel to the floor). The two plots show the outcome of each of the ten repetitions. The mean

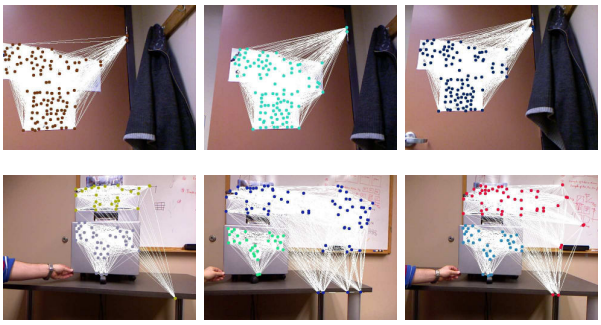


Fig. 10: We performed 10 experiments with a door and 10 with a drawer under varying lighting conditions, sensor position and orientation, and object initial and final configuration. Segmentation results are consistent and repeatable.

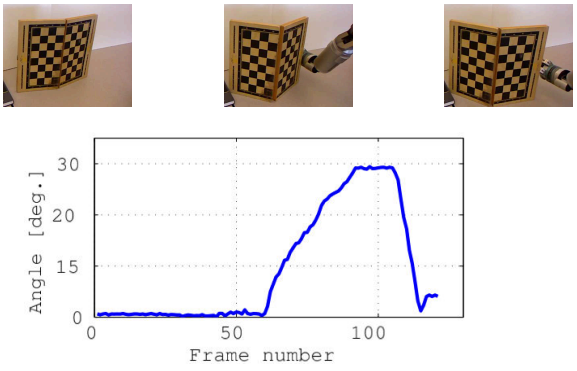



Fig. 11: Tracking the configuration of a checkerboard: the robot translates the object, rotates the right-half, then rotates it back. The plot shows the resulting change in configuration.

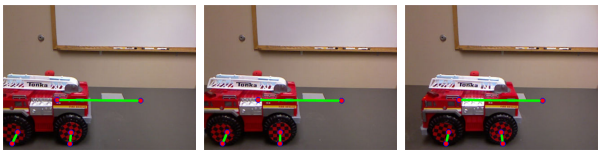


Fig. 12: Experimental results showing a sequence of interactions with a firetruck toy and the corresponding joint detection results. Joint detection is complex here, as the revolute axes translate with the body of the truck. The sequence shows that joint detection remains reliable throughout the sequence.

angles are $88.43^\circ$ and $177.3^\circ$ respectively. Considering the inevitable small misalignment in the orientation of the sensor with respect to the floor, manufacturing and construction inaccuracies and sensor resolution, we believe this level of repeatability and accuracy is sufficient for manipulation.

### D. Configuration Tracking

The clusters of point features are tracked throughout the interaction with the object. This enables the robot to monitor its progress towards achieving its manipulation objective. Figure 11 illustrates this process for a checkerboard. Here, the robot correctly clustered features for each part of the board and detected the revolute joint between them. The figure shows the perceived change in angle throughout the robot's interaction with the object.

To manipulate and monitor the execution of a manipulation plan, the robot must also be able to track the position and orientation of the joints throughout the interaction. Figure 12 shows the performance of our algorithm in tracking the position and orientation of degrees of freedom over time. The sequence shows a firetruck toy in several configurations, with the detected joints. We note that the axes of the revolute joints move as the body of the truck translates. Together with the ability to monitor the configuration of each rigid body, our algorithm enables the robot to monitor the progress of its manipulation task and detect any failures.

## VIII. Discussions

As shown through our experimental results, the proposed algorithm detected, segmented, and tracked the segmentation of all rigid bodies containing a sufficient number of visual features. This robustness is achieved using an off-the-shelf RGB-D sensor (Kinect), and without tuning any parameters. Experiments were performed under uncontrolled lighting conditions, different sensor positions and orientations, and for different initial poses of the objects. The demonstrated robustness and effectiveness provides evidence that the presented perception skill is suitable for manipulation in unstructured environments.

Our algorithm has four important advantages: First, its run time is only a few milliseconds, significantly outperforming other methods. Second, when using SIFT features, it is insensitive to occlusions during the interaction. Third, it requires very small object motions—an important property when interacting with unknown objects. And last but not least, it makes no assumptions about the shape and kinematic structure of objects, and unlike existing methods, requires no tuning.

The most significant limitation of our method is its dependency on feature tracking. In our ongoing research we are developing perceptual features that are suitable for texture-less objects. We believe that with these features, also texture-less objects could be modeled following the ideas we presented. A second limitation is the efficiency of the robot's interactions with the environment. As explained earlier, in this work the interactions were scripted. In [7], a reinforcement learning algorithm is used to learn manipulation strategies. We believe that this can be extended to our 3D problem as well.

## ACKNOWLEDGMENT

This work was conducted (in part) through collaborative participation in the Robotics Consortium sponsored by the U.S Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement W911NF-10-2-0016. The authors also gratefully acknowledge alliance and a grant from the Intel Science and Technology Center for Embedded Computing, and funding under the DARPA Autonomous Robotic Manipulation Software Track (ARM-S) program.

## References

[1] J. Aloimonos, I. Weiss, and A. Bandyopadhyay. Active Vision. *IJCV*, 1(4):333–356, January 1988.

[2] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-Squares Fitting of two 3D Point Sets. *IPAMI*, 9(5):698–700, September 1987.

[3] P. Fitzpatrick. First Contact: An Active Vision Approach to Segmentation. In *IROS*, pages 2161–2166, 2003.

[4] D. A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002.

[5] D. Katz and O. Brock. Manipulating Articulated Objects with Interactive Perception. In *ICRA*, pages 272–277, Pasadena, CA, USA, May 2008. IEEE Press.

[6] D. Katz, A. Orthey, and O. Brock. Interactive perception of articulated objects. In *ISER*, India, 2010.

[7] D. Katz, Y. Pyuro, and O. Brock. Learning to Manipulate Articulated Objects in Unstructured Environments Using a Grounded Relational Representation. In *RSS*, pages 254–261, Zurich, Switzerland, June 2008.

[8] J. Kenney, T. Buckley, and O. Brock. Interactive Segmentation for Manipulation in Unstructured Environments. In *ICRA*, pages 1343–1348, Kobe, Japan, May 2009. IEEE Press.

[9] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*, 60(2):91–110, 2004.

[10] B. D. Lucas and T. Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *IJCAI*, pages 674–679, Canada, August 1981.

[11] D. A. Ross, D. Tarlow, and R. S. Zemel. Unsupervised Learning of Skeletons from Motion. In *ECCV*, pages 560–573, Germany, 2008. Springer-Verlag.

[12] J. Sturm, A. Jain, C. Stachniss, C. Kemp, and W. Burgard. Operating Articulated Objects Based on Experience. In *IROS*, Taiwan, 2010.

[13] J. Sturm, K. Konolige, C. Stachniss, and W. Burgard. Vision-Based Detection for Learning Articulation Models of Cabinet Doors and Drawers in Household Environments. In *ICRA*, Alaska, May 2010. IEEE.

[14] J. Yan and M. Pollefeys. Automatic Kinematic Chain Building from Feature Trajectories of Articulated Objects. In *CVPR*, pages 712–719, USA, 2006.