The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

Data-Driven Grasping with Partial Sensor Data

Corey Goldfeder, Matei Ciocarlie, Jaime Peretzman, Hao Dang and Peter K. Allen

Abstract— **To grasp a novel object, we can index it into a database of known 3D models and use precomputed grasp data for those models to suggest a new grasp. We refer to this idea as data-driven grasping, and we have previously introduced the Columbia Grasp Database for this purpose. In this paper we demonstrate a data-driven grasp planner that requires only partial 3D data of an object in order to grasp it. To achieve this, we introduce a new shape descriptor for partial 3D range data, along with an alignment method that can rigidly register partial 3D models to models that are globally similar but not identical. Our method uses SIFT features of depth images, and encapsulates "nearby" views of an object in a compact shape descriptor.**

I. Introduction

An important body of recent work in dexterous grasping has focused on empirical approaches, such as simulation [23], [24], [12], demonstration [1] and stochastic optimization [5], rather than anayltical solutions. A number of researchers have investigated data-driven grasp strategies. Bowers and Lumia [3] collected grasps for a small number of planar objects and used fuzzy logic to extrapolate grasping rules. Morales *et al.* [24] used our *GraspIt!* tool [22] to compute offline grasps for a small database of CAD objects, and successfully executed those grasps on the real versions of those objects. Saxena *et al.* [31] generated 2D renderings of a large set of example objects, and learned a model-free mapping from images to graspable features. They also considered the problem of finding grasp points from partial shape information [32]. Glover *et al.* learned robust probabilistic models to recognize several classes of graspable objects from example images [11]. Noel and Jing [6] created a small database of example objects with grasps planned in *GraspIt!* and used simple object features to select pre-grasps. Li and Pollard [19] collected a database of 17 hand poses, and used shape matching to match poses to objects, rather than matching new objects to known objects. An advantage of our approach is that we can plan grasps on occluded parts of a model, if similar models are present in the database, while an advantage of their approach is that it always plans grasps on known geometry. In addition, building their pose database required significant manual annotation, which may be difficult to scale to larger datasets. Finally, several data-driven grasping algorithms have been targeted at grasp animation rather than grasp synthesis [7], [2], [34].

We have previously introduced the Columbia Grasp Database (CGDB) [13], a freely available collection of 238,737 form closure grasps on the 1,814 models of the Princeton Shape Benchmark (PSB) [33], at 4 discrete scales using several robotic hands. We have also shown how this database can be used to synthesize grasps for novel objects. Our starting assumption is that geometrically similar models can be grasped in similar ways. Based on this, our algorithm for data-driven grasping is to match an object against the Columbia Grasp Database, using some measure of shape similarity, and to reuse known good grasps from similar models. Our previous work required a full 3D model as input, which limited its applicability in unstructured environments. In this paper we extend our data-driven grasp planning algorithm to function on partial shape information that can be realistically collected by a robot's sensors.

Switching to partial sensor data poses several new challenges. In [13] we used 3D Zernike descriptors [25] to match an object against the models in the database, and the Principal Axes Transform to globally align all models to a canonical coordinate system. However, matching and aligning *partial* 3D models is a much more difficult problem.

After choosing similar models from the CGDB and aligning their coordinate systems with the object to be grasped, we can transfer the known good grasps from the neighbors to the object being grasped. A single model in the CGDB may have as many as 15 known good grasps, and so we need a ranking function to determine which grasp to actually output. When a full 3D model was available, we were able to simulate all of the candidate grasps on the object's own geometry, allowing us to unambiguously select the best candidate. However, without access to the full object geometry we must look for an alternative ranking algorithm.

This paper makes the following contributions:

- a shape similarity measure for matching partial sensor scans to 3D models,
- a new alignment method to transform partial sensor scans into the coordinate systems of 3D models, and
- a new method of ranking grasps by how well they are expected to generalize to other models

We note that the first two contributions are not specific to grasping, and may find applications in other domains.

II. Matching with Partial Sensor Data

Our goal in this work is to demonstrate data-driven grasp planning using only *partial* sensor data. Specifically, we are interested in range sensor data collected from a small number of viewpoints that don't cover the entire surface of the object. We do not assume any particular range sensor, and in principle our method can be applied to scan data from a variety of sources, such as lidar, structured light, or stereo

All authors are with the Dept. of Computer Science, Columbia University, NY, USA. Email: {coreyg, cmatei, allen}@cs.columbia.edu, {jp2642, hd2181}@columbia.edu. This work was funded in part by NIH BRP grant 1RO1 NS 050256-01A2, and by a Google research grant.

reconstruction. We assume that the robot has some degree of mobility, and can move its sensor to a few nearby viewpoints. We formalize what we mean by "nearby" below. Figure 1 shows an example of our full pipeline, from matching through alignment and grasping, for an object acquired with a NextEngine laser scanner. We will refer to this figure as each step is explained.

Finding 3D models in range data is a well-studied problem [16], [9]. However, it is usually formulated as a *recognition* task rather than as a *matching* task. In a recognition task we are looking to recognize an instance of a known object, and perhaps recover pose and scale. In a matching task we assume that the object in the scan is not previously known, and we are interested in detecting similarities to known objects. These two tasks necessitate somewhat different techniques. In particular, in the shape matching case, meaningful local correspondences between features in the query model and the nearest match in the database may not exist.

A. Shape Matching with Depth Images

Our approach builds on work on 3D shape matching for triangle soups. Although a large number of shape matching algorithms have been presented based on 3D features [25], [17], [28], in recent years the most successful attempts have been based on 2D silhouette images rendered from viewpoints sampled on a sphere and looking in the direction of the model's center of mass [4]. Ohbuchi *et al.* [27] rendered synthetic *depth images* of a 3D model from multiple viewpoints. They computed SIFT features [20] and clustered them into a representative codebook. Each model was described by a bag-of-feature that drew features from all viewpoints. The dissimilarity between models was taken as the Kullback-Leibler divergence [18] of their histograms.

When working with range sensor data there are significant advantages to operating in the depth image space rather than using reconstructed point clouds. Although different range sensors have different noise profiles, we have observed across a variety of sensors that depth gradients are more consistent than the actual depth values. The SIFT descriptor encodes gradients rather than depths, potentially allowing it to adapt to sensor noise. Additionally, depth images contain more scene information than point clouds, since they distinguish between known empty space and unseen space.

B. Extension to Partial Sensor Data

The key innovation of [27] was to describe a 3D model using a single bag-of-features, encoding features drawn from many views of the model in the same histogram. Although they used views from all directions in their descriptor, in principle, one could construct a bag-of-features using any subset of views, with the caveat that the bag would describe only those parts of the model that could be seen from those views. Suppose that instead of collecting images from all around an object's enclosing sphere, we collected images only from a circular portion of the sphere; that is, from a spherical cap. (The "image" from a given viewpoint may be a simulated depth image rendered on a computer, or may be a sensed image of a real object.) We can identify a spherical cap by its center point v on the sphere and by the solid angle Ω that it subtends. If we collect features from viewpoints chosen on the cap, we can combine them into a single bag-of-features histogram. We use the notation $Cap_\Omega(v)$ to refer to the histogram of codebook features drawn from viewpoints on the spherical cap centered on v and subtending the solid angle Ω. We define the "$CapSet_\Omega$ descriptor" of a model as the set $\{Cap_\Omega(v) \; \forall \; v\}$.A complete sphere subtends 4π steradians, and so the $CapSet_{4\pi}$ descriptor contains only a single histogram that incorporates features from all viewpoints. For notational convenience, we refer to the histogram of a single depth image - that is, the features collected from a single view v, as $Cap_0(v)$. The first line of Figure 1 shows how a set of depth images from viewpoints within a spherical cap can be used to construct a *Cap* descriptor. If we take v to be the initial location of a robot's movable sensor, and the spherical cap to represent nearby positions that the sensor can be moved to, then we can think of $CapSet_\Omega(v)$ as encoding the portion of the object that the robot is able to see. If $\Omega = 4\pi$ the sensor can be moved anywhere around the object, whereas if $\Omega = 0$ the sensor is fixed and cannot be moved at all. As expected, what a robot can see is a function of the initial position of its sensor and the distance the sensor can be moved.

There are an infinite number of points on a sphere, and so $CapSet_\Omega \; , \Omega < 4\pi$ is an infinite set as well. Suppose that we have computed feature histograms for a set of viewpoints V which are distributed (approximately) uniformly on the sphere. Given a viewpoint $v \in V$ and a solid angle Ω, we can approximate $Cap_\Omega(v)$ by simply adding the histograms of all sampled views that are within the cap of size Ω centered on v. The number of possible *Cap* descriptors - that is, an approximation of the *CapSet* - is just $|V|$. The number of useful values of Ω is also a function of sampling density $|V|$, since the difference between the *Cap* descriptors for two different values of Ω is only distinguishable if the difference in cap size is large enough to include at least one additional sampled viewpoint. For the remainder of this paper we will use the $Cap_\Omega(v)$ and $CapSet_\Omega$ notations to refer to these sampled approximations.

Given a distance $D(a,b)$ between *Cap* descriptors, we can define the distance between a *CapSet* A and a *Cap* b as the minimum distance between b and any $a \in A$

$$D(A,b) = min \; D(a,b) \; a \in A \tag{1}$$

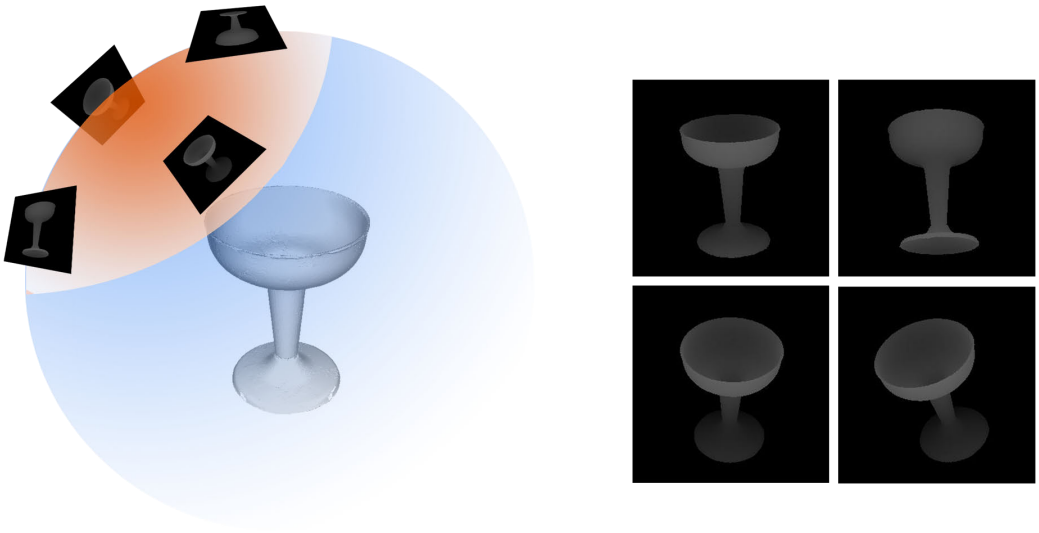
and a distance between *CapSet* descriptors A and B as the minimum distance between any pair of their *Cap* descriptors

$$D(A,B) = min \; D(a,b) \; a \in A, b \in B. \tag{2}$$

C. Implementation and Matching Results

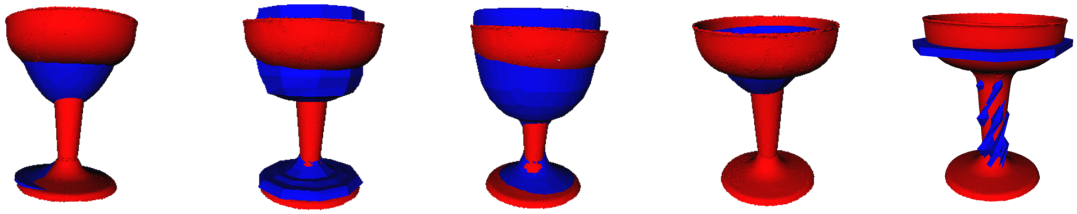
For each model of the CGDB we rendered 60 grayscale depth images, using linearly spaced w-buffer values rather than exponentially spaced z-buffer values, from viewpoints placed at the vertices of an enclosing truncated icosahedron. We computed SIFT descriptors on each image, finding an

A *Cap* descriptor of a 3D object captures how the object appears to the depth sensor from a particular viewpoint. The *Cap* is a function of the object, the initial viewpoint, and the range of motion available to the depth sensor. SIFT features from nearby views are aggregated into a single binary vector that partially describes the object.



These CGDB models had Cap_{π} descriptors similar to that of the sensed model.

Using a combination of *Cap* matching and ICP, we can closely align the CGDB models with the sensed model. Although we show the alignments using the full 3D geometry of the sensed model, the transformations were actually computed using only the partial geometry seen by the sensor from its limited viewpoints.



We can transfer pre-grasps from the neighbor objects to the sensed object. Here we show the candidate from each neighbor with highest grasp quality transferred to the sensed object. Four of the transferred grasps result in form closure and one does not.

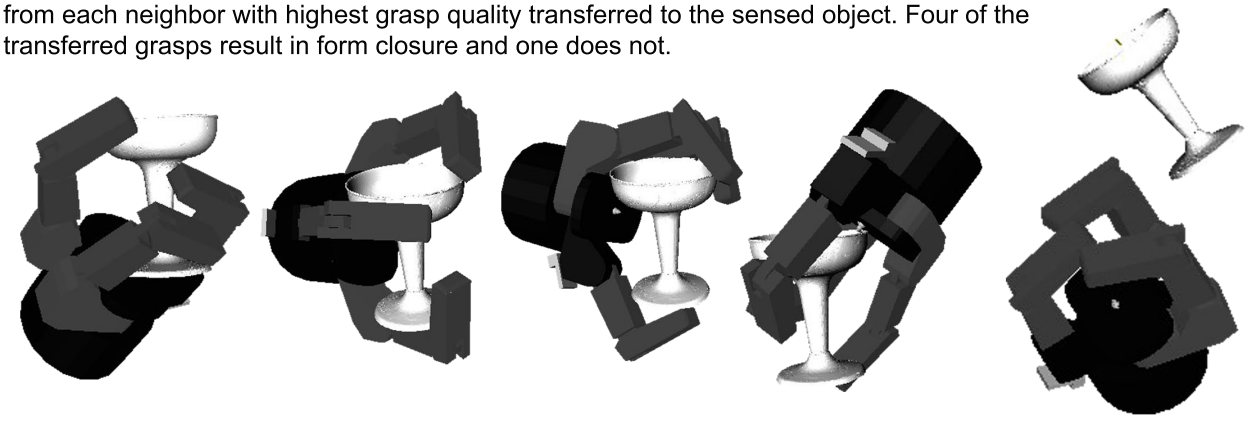


Fig. 1. Our full pipeline, for a real example using depth data acquired with a NextEngine scanner.

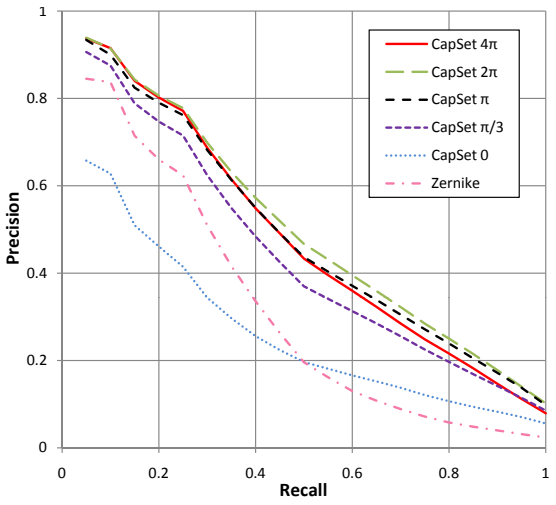


Fig. 2. Precision/Recall plots for *CapSet* descriptors with 5 values of Ω, as compared to Zernike descriptors.

average of 24 features per image. For 1,814 models with 60 images apiece this gave us 2.6 million individual SIFT features. To create the bag-of-features codebook, we down-sampled this set to 234,081 features by using only features rendered from the 6 sides of the bounding box. For each feature we found the distance to its 500 nearest neighbors, creating a sparse distance matrix which we fed into affinity propagation [8] to find 1496 clusters. We chose the mean of the features in each cluster as a codebook entry[1].

For each depth image we find the codebook entries which best approximate its SIFT features. Unlike [27] we use a binary bag of features rather than a histogram, using the simple absence/presence rule that was shown to perform well in [26]. This is an intentional choice, as we argue that the repetition of features between closely space viewpoints is to be expected, and may be considered a sampling artifact rather than a feature of the model. With binary bags we can also use the Jaccard distance [15]

$$J_\delta(A,B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|} \tag{3}$$

as a distance between *Cap* descriptors, avoiding the need for normalizing empty bins associated with Kullback-Leibler divergence.

Figure 2 shows the precision/recall values for *CapSet* descriptors with 5 distinguishable Ω values, ranging from 0 (features drawn from a single image) to 4π (features drawn from the full sphere). The values are averaged over the "test" classes of the Princeton Shape Benchmark. For comparison, we provide the plot for the Zernike descriptors [25] we have previously used [13]. What is most interesting is that the plots for $\Omega \geq \pi$ are nearly indistinguishable, with 94% precision for the first recalled match. This means that the *CapSet* descriptors work essentially as well using only the minimums between π steradians as they do using the entire sphere. Even dropping to $CapSet_{\pi/3}$ - using *Cap* descriptors representing approximately a single steradian - results in a descriptor that substantially outperforms the Zernike descriptors and only slightly lags the descriptor for the full sphere. At $\Omega = 0$, using descriptors based on only single depth views, the initial precision is still a respectable 66%. The second line of Figure 1 shows matching results from the PSB, using a Cap_π descriptor representing the portion of the object seen from the spherical cap shown on the first line of the figure.

Since we take the minimum distance between *Cap* descriptors in a *CapSet* as the overall distance, these plots can be thought of as partial matching results using the "most descriptive" viewpoint for each model. One could, in theory, construct a model that is featureless on one hemisphere, which would make matching based on views from that hemisphere difficult. Still, while the choice of viewpoint is indeed important for small values of Ω, as Ω increases it becomes less so, since more of the model is seen from any starting viewpoint. In practice we have found that for $\Omega \geq \pi$ nearly all viewpoints are "good enough" and the choice of v does not significantly impact the resulting matches.

III. Grasping with Partial Sensor Data

To grasp an object, we collect range scans of it from a number of viewpoints. These scans are processed for SIFT features, and are used to form a single $Cap_\Omega(v)$, where v is the most central viewpoint for which we have data, and Ω is an estimate of the solid angle that we have collected data for. (Collecting more images will improve the matching results, but as we showed in Section II-C the method will gracefully degrade with less data, even down to a single depth image.) We can compare this sensed *Cap* to the $CapSet_\Omega$ descriptors of the database models, using Equations (3) and (1), and choose k neighboring models.

A. Aligning Partial Data to Neighbor Models

Once we have selected neighbors we must align their coordinate systems with the sensed partial 3D model. There is a good deal of literature on range scan registration [10], [30], but our problem is significantly more difficult, as we are attempting to align a partial scan of one model with the geometry of a *different*, neighboring model, rather than with an overlapping scan of the same model from a different viewpoint. Many researchers have proposed alignment methods that find correspondences between local features [14], [16]. We prefer to avoid methods that build upon local feature correspondences since globally similar models may have few local correspondences suitable for alignment. Makadia *et al.* [21] correlated Extended Gaussian Images of two point clouds using spherical harmonics, and achieved excellent results but recommended a minimum of 45% overlap between the models, which we cannot reasonably assume.

As is customary, we break alignment into a rough stage and a fine stage, with the fine stage consisting of the Iterative Closest Point (ICP) [29] algorithm. The challenge

[1]As per [26] we were striving for 1500 clusters, but affinity propagation makes it difficult to control the final number of clusters exactly. We built the codebook using features from both the "test" and "train" collections of the PSB, as our goal was to build a good codebook for matching unknown objects against the entire CGDB.

is choosing a rough alignment as input for the ICP stage. We can use the *Cap* descriptors themselves to produce a good initial transformation. We first align the center of mass of the sensed point cloud with the center of mass of the 3D model. Given a *Cap c* representing the partial data and a *CapSet M* representing the database model, we can find the viewpoint v that minimizes their distance as

$$v = \arg\max |c \cap m|, \; m \in M \tag{4}$$

where c is held fixed. We assume that c and v represent the same part of their respective objects, and so we rotate the models so that the center viewpoints of both the c and v lie along the same ray from the origin. The ambiguous roll about the view vector is resolved by taking silhouettes of both models from the shared view direction and template-matching the rotations to find the best overlap. The output of this stage is an initial alignment which can be refined by ICP. The third line of Figure 1 shows how our method aligns the sensed object with its neighbors, using only the partial 3D data seen from the limited set of viewpoints. Our work on partial model alignment represents ongoing research. Our alignment results appear qualitatively good to human observers, but we have not yet benchmarked our method against other alignment methods for partial data.

B. Candidate Selection

Each model in the CGDB has previously been annotated with between 5 and 15 form closure grasps derived from stochastic search [13]. For each grasp, we also know the pre-grasp position, which can be thought of as "pure" grasp information untainted by contact with an actual object. Once we have chosen k neighbors for our object, and aligned the coordinate system of the object with those of the neighbors, we can borrow any of those pre-grasps as a candidate for grasping the new object. However, not all of the grasps in the CGDB generalize well to neighboring models. Some grasps rely on very specific features of a model and would not work on even a very close neighbor if those features were removed or changed. Ideally we would like some way to detect this situation, so that we can output only "generalizable" grasps that are likely to work on the new object.

Given a set of candidate pre-grasps drawn from neighbors, how can we know how generalizable they are? If a full 3D model were available we could simulate all of the candidates on the model and measure this directly, but absent such a model we need an indirect way to predict which grasps to output. Our solution is to "cross-test" grasps between neighbors. Although we don't have a full 3D model of the sensed object, we do have full models for its neighbors from the CGDB. Given the set of neighbors, $N = \{n_1 \ldots n_k\}$, we simulate the candidate pre-grasps from neighbor n_i on the other $k-1$ neighbors. We can then rank the candidate grasps by the number of neighbors for which the pre-grasp resulted in form closure. Within each rank, we can further order the grasps by the average Ferrari-Canny ε-quality of the form closure grasps (including the quality on the model itself). A grasp that is highly generalizable can be expected to work well on the other neighbors of the sensed object, and so by ranking grasps in this fashion we increase the likelihood of choosing a good candidate grasp.

IV. Experimental Results

We tested our pipeline using a simulated Barrett hand in *GraspIt!* [22]. As in [13] we report aggregate results over the 1,814 scale 1.0 models of the CGDB. For each model, we constructed 6 Cap_π descriptors, each centered on a viewpoint perpendicular to a face of the oriented bounding box. This was done so as to demonstrate the robustness of the method to the choice of views. While we do have full 3D models of each CGDB object, our matching and alignment methods were only given the partial geometry visible from the viewpoints of the input *Cap*.

For each of the 6 *Cap* descriptors we found 3 nearest neighbors[2] from among the other models of the CGDB. We cross-tested the grasps from each neighbor model on the other neighbor models, and ranked the grasps as described in the preceding section. We then simulated the highest ranked grasps on the full 3D geometry of the object.

We evaluated the candidate grasps in two ways. We first performed "static" analysis of the grasp by placing the hand in the pre-grasp position and closing the fingers until they contacted the object. If this resulted in a form closed grasp we reported success; if not, we reported failure. Static analysis is overly conservative, often failing on grasps that appear visually correct, as it does not allow the object to move within the hand as grasp forces are applied. For this reason, we also employed "dynamic" analysis. We used a time-stepping numerical integration to compute the movements of all bodies in the system, including robot links and the object being grasped, based on the actuator forces applied at the joints of the hand. For details on our simulation methodology we refer the reader to [22]. This allowed us to evaluate the grasp taking into account friction and joint constraints, inertial effects, and movement due to grasp forces. We are interested only in the intrinsic quality of the grasp, and so we do not simulate the surface that the object lies on or other environment-specific properties of the system. In a real robotic application we expect the path planning module to decide if a grasp is unreachable due to the table or other obstacles and if so to choose the next best grasp; as such the grasp planner can always assume that the grasp it proposes is obstacle-free. Nevertheless, we emphasize that this is a dynamic grasp *analysis* rather than a physically realistic simulation.

The models in the CGDB come from the Princeton Shape Benchmark, which is one of the only freely usable 3d model collections of nontrivial size. However, the PSB models were not chosen with an eye towards graspability, and many of the models, such as insects and plants, have sharp features that are hard to grasp even when rescaled to "toy"

[2]As described in [13], each neighbor could actually represent 2 models, since we have 4 scaled versions of each CGDB model and we use both the smallest version larger than the test object and the largest version smaller than the test object, if both are available.

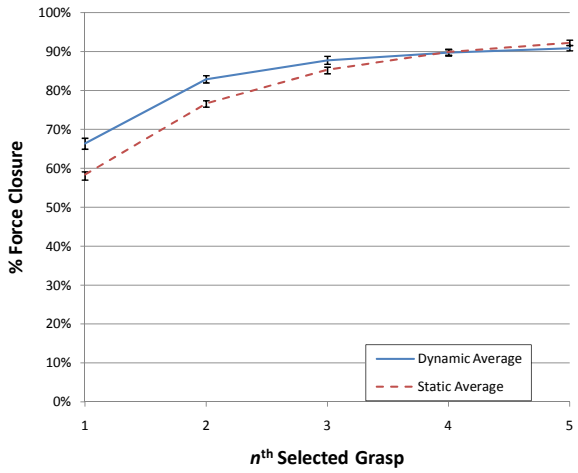


Fig. 3. Static and dynamic grasp analyses, averaged over grasps planned on the 1,814 models in the CGDB. We found candidate pre-grasps using partial data from 6 sets of nearby views, and ordered them by their "cross testing" score. We report the percentage of form closure grasps within the first 5 ordered candidates, averaged over the 6 sets of view. The error bars show the standard deviation between the views, which we note is very small.

size. Additionally, approximately 15% of the models exhibit problematic topology, such as inconsistent normals or jagged unclosed edges. These models could not be analyzed by the dynamic method, and so while static results were averaged over the entire database, dynamic results were only averaged over the 85% we could process. The broken models were distributed fairly evenly throughout the database.

Figure 3 shows the results for static and dynamic analysis of the grasps chosen using each of the 6 sets of views. We report the percentage of models that were successfully grasped within the first n selected grasps as ranked by our cross-testing approach, averaged over the 6 sets of views we tested. It is apparent from the very small standard deviations that the choice of view does not materially affect the likelihood of finding a form closure grasp. The static analysis suggests that 77% of the models in the CGDB were grasped successfully by either the highest or second-highest ranked pre-grasp, while the dynamic analysis raises this number to 83%. Both analyses report that 90% of the models were successfully grasped by one of the top 4 candidate grasps. Given that 3 neighbors from the CGDB can mean as many as 90 candidate grasps, these numbers imply that our cross-testing algorithm is intelligently ranking the candidates.

V. Conclusions and Future Work

We have presented a grasp planner that works on partial 3D scans of an object, along with tools for matching and aligning such data. We showed results for the models in the CGDB at scale 1.0, using both static and dynamic analyses. We also perfomed initial experiments with real data from a NextEngine laser scanner, as shown in Figure 1. In future work we will experiment further with real robotic hands.

References

[1] J. Aleotti and S. Caselli. Robot grasp synthesis from virtual demonstration and topology-preserving environment reconstruction. In *IROS*, 2007.

[2] Y. Aydin and M. Nakajima. Database guided computer animation of human grasping using forward and inverse kinematics. *Computers and Graphics*, 23, 1999.

[3] D. L. Bowers and R. Lumia. Manipulation of unmodeled objects using intelligent grasping schemes. *Transactions on Fuzzy Systems*, 11(3), 2003.

[4] D.Y. Chen, X.P. Tian, Y.T. Shen, and M. Ouhyoung. On visual similarity based 3d model retrieval. *Computer Graphics Forum*, 22, 2003.

[5] M. Ciocarlie, C. Goldfeder, and P. K. Allen. Dimensionality reduction for hand-independent dexterous robotic grasping. In *IROS*, 2007.

[6] N. Curtis and J. Xiao. Efficient and effective grasping of novel objects through learning and adapting with a knowledge base. In *IROS*, 2008.

[7] G. ElKoura and K. Singh. Handrix: Animating the human hand. In *Symposium on Computer Animation*, 2003.

[8] B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315, 2007.

[9] A. Frome, D. Huber, R. Kolluri, T. Bulow, and J. Malik. Recognizing objects in range data using regional point descriptors. In *ECCV*, 2004.

[10] N. Gelfand, N. J. Mitra, L. J. Guibas, and H. Pottmann. Robust global registration. In *Symposium on Geometry Processing*, 2005.

[11] J. Glover, D. Rus, and N. Roy. Probabilistic models of object geometry for grasp planning. In *RSS*, 2008.

[12] C. Goldfeder, P. K. Allen, C. Lackner, and R. Pelossof. Grasp planning via decomposition trees. In *ICRA*, 2007.

[13] C. Goldfeder, M. Ciocarlie, H. Dang, and P. K. Allen. The Columbia Grasp Database. In *ICRA*, 2009.

[14] D. F. Huber and M. Hebert. Fully automatic registration of multiple 3D data sets. *Image and Vision Computing*, 21(7), 2003.

[15] P. Jaccard. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*, 37, 1901.

[16] A. E. Johnson and M. Hebert. Surface registration by matching oriented points. In *3DIM*, 1997.

[17] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz. Rotation invariant spherical harmonic representation of 3D shape descriptors. In *Symposium on Geometry Processing*, 2003.

[18] S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22, 1951.

[19] Y. Li, J. L. Fu, and N. S. Pollard. Data-driven grasp synthesis using shape matching and task-based pruning. *Transactions on Visualization and Computer Graphics*, 13(4), 2007.

[20] D. G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, 1999.

[21] A. Makadia, A. Patterson, and K. Daniilidis. Fully automatic registration of 3D point clouds. In *CVPR*, 2006.

[22] A. Miller, P. K. Allen, V. Santos, and F. Valero-Cuevas. From robot hands to human hands: A visualization and simulation engine for grasping research. *Industrial Robot*, 32(1), 2005.

[23] A. Miller, S. Knoop, H. I. Christensen, and P. K. Allen. Automatic grasp planning using shape primitives. In *ICRA*, 2003.

[24] A. Morales, T. Asfour, P. Azad, S. Knoop, and R. Dillmann. Integrated grasp planning and visual object localization for a humanoid robot with five-fingered hands. In *IROS*, 2006.

[25] M. Novotni and R. Klein. 3D Zernike descriptors for content based shape retrieval. In *Solid Modeling and Applications*, 2003.

[26] E. Nowak, F. Jurie, and B. Triggs. Sampling strategies for bag-of-features image classification. In *ECCV*, 2006.

[27] R. Ohbuchi, K. Osada, T. Furuya, and T. Banno. Salient local visual featuers for shape-based 3D model retrieval. In *SMI*, 2008.

[28] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin. Shape distributions. *Transactions on Graphics*, 21(4), 2002.

[29] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *3DIM*, 2001.

[30] J. Salvi, C. Matabosch, D. Fofi, and J. Forest. A review of recent range image registration methods with accuracy evaluation. *Image and Vision Computing Archive*, 25(5), 2007.

[31] A. Saxena, J. Driemeyer, and A. Ng. Robotic grasping of novel objects using vision. *International Journal of Robotics Research*, 27(2), 2008.

[32] A. Saxena, L. L. S. Wong, and A. Y. Ng. Learning grasp strategies with partial shape information. In *AAAI*, 2008.

[33] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser. The Princeton Shape Benchmark. In *Shape Modeling and Applications*, 2004.

[34] K. Yamane, J. Kuffner, and J.K. Hodgins. Synthesizing animations of human manipulation tasks. *Transactions on Graphics*, 23(3), 2004.