2019 International Conference on Robotics and Automation (ICRA)
Palais des congres de Montreal, Montreal, Canada, May 20-24, 2019

Using Geometric Features to Represent Near-Contact Behavior in Robotic Grasping

Eadom Dessalene[1], Yi Herng Ong[2], John Morrow[2], Ravi Balasubramanian[2], Cindy Grimm[2]

Abstract— **In this paper we define two feature representations for grasping. These representations capture hand-object geometric relationships at the near-contact stage — before the fingers close around the object. Their benefits are: 1) They are stable under noise in both joint and pose variation. 2) They are largely hand and object agnostic, enabling direct comparison across different hand morphologies. 3) Their format makes them suitable for direct application of machine learning techniques developed for images.**

We validate the representations by: 1) Demonstrating that they can accurately predict the distribution of ϵ-metric values generated by kinematic noise. I.e., they capture much of the information inherent in contact points and force vectors without the corresponding instabilities. 2) Training a binary grasp success classifier on a real-world data set consisting of 588 grasps.

I. INTRODUCTION

Fundamentally, the goal of a grasping metric or representation is to reduce the complex physics of hand-object interaction down to a small set of values that can be reasoned about. Desirable properties include: Stability with respect to noise (joint angles, pose uncertainty, object shape variation), concise, useful for prediction, efficient to calculate, hand morphology agnostic, and suitable for machine learning/control strategies.

In this paper we define two feature representations that are driven by the need to describe near-contact grasping (just *before* the fingers close) in a form suitable for machine learning. Broadly speaking, these representations capture the geometric relationships between the surfaces of the fingers and the object as they come into contact. By capturing data in a grid structure (rectangular set of samples per finger pad/palm) we can directly employ existing image-based machine learning algorithms.

More specifically, for every potential finger contact surface (typically the pad of each finger and the palm, but we can include all of the finger links if desired) we define a grid of points. For each of those points, we record the distance (and optionally, orientation) to the object (see Figure 1). All of these grids can be combined into a single 2D “image” using tiling.

Our two feature representations differ by how they calculate distance. In the first representation, called the *signed distance feature*, we use a Signed Distance Function (SDF) around the object to calculate the distances. The second feature representation, called the *wedge distance feature*, uses the closest distance within a *wedge* projected out from the finger pad. This approach more accurately captures the potential contact surface (assuming the finger pad is moving in the direction it is facing) at increased computational cost.

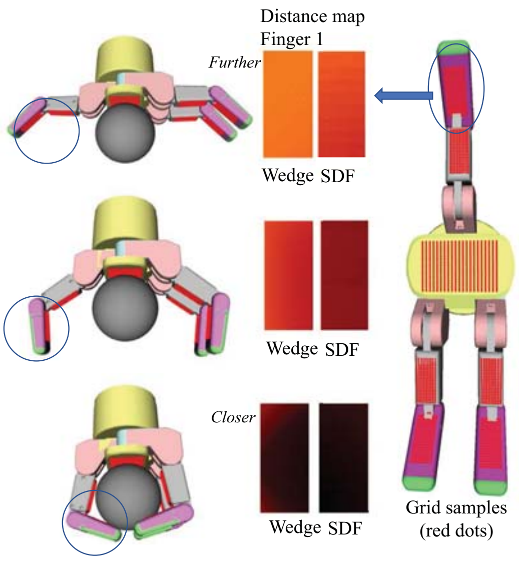

Fig. 1: Example distance maps as the hand closes. Each red region on the hand generates one map (each red dot on the finger generates one pixel, map for circled finger link shown). Orange is further away, black touching.

Existing metrics tend to be very concise (usually just a handful of numbers) and quick to compute. However, many of the most common metric approaches are based on some form of analyzing contact points, which means they cannot be used before contact and are often unstable. This instability arises in practice because of small variations in pose, object surface, or joint angles that cause changes in the number and location of these contacts. Continuous metrics based on joint angles are more stable but do not transfer well between hand morphologies and are an indirect measure of the grasp. In our representation we trade off conciseness and computational efficiency for stability and ease of including in machine learning algorithms.

Specifically, by construction our representations tend to be very stable and continuous — small changes in pose, finger pad location, or surface result in correspondingly small

*This work was supported in part by NSF grants CNS 1730126 and CNS 1659746

[1]E. Dessalene is with the department of Computer Science and Mathematics, George Mason University, 4400 University Drive, Fairfax USA `edessale@gmu.edu`

[3]Oregon State University, Collaborative Robotics and Intelligent Systems (CoRIS) `ongyi,morrowj, Ravi.Balasubramanian, Cindy.Grimm @ oregonstate.edu`

changes in distance. Additionally, distances close to zero capture contact in non-abrupt manner. This continuity is also helpful for convergence of machine learning algorithms.

The computation time of our features is linear in the number of samples needed (SDF) or samples times number of points needed to represent the object (Wedge). These features are also easy to parallelize or put on a GPU.

Of consideration is that this feature representation could be implemented directly in hardware using a set of time of flight or electrical field sensors, enabling real-time control without explicitly modeling the object.

A feature or metric should also be *predictive* — i.e., useful for evaluating grasps. We validate our features by 1) analyzing their ability to predict the expected values of the ϵ-metric and by 2) using it to predict whether or not a real-world grasp will succeed.

The ϵ-metric prediction is conducted in simulation with three different hand morphologies, ranging from simple to complex. We use simulation to generate, for each grasp, multiple ϵ-metric values by adding noise to both pose of the hand and the finger joints. We then learn the average and expected variation of the ϵ-metric . In essence, this validation test checks that our representations contain within them the information the ϵ-metric is based on. On a practical note, this learned prediction could be used in practice instead of actually generating the expected value of the ϵ-metric (which is computationally expensive).

Our real-world validation uses a database of 588 grasps tested on the Barrett Hand using four different objects. Since we know where the hand and the object are, we can calculate our features for each grasp. We demonstrate that a simple CNN classifier is capable of learning whether or not a grasp will be successful.

This paper is organized as follows: Related work is discussed in Section II. In Section III we define our features and the algorithms used to compute them. Section IV describes the setup of our ϵ-metric and real-world validation tests. Results follow in Section V.

II. RELATED WORK

Early analytical approaches to robotic grasping represented a grasp as a set of contact wrenches, relying on exact knowledge of contact locations and orientations, as well as knowledge of the center of mass of the object and friction coefficients of the object [1], [2], [3]. However, these contact representations tend to fail in the real world when exposed to both kinematic and visual uncertainty. They also suffer from lack of reproducibility, in that the contacts are difficult to consistently reproduce in reality. The ϵ-metric is a common metric that uses these contact representations to assess the quality of a force-closure grasp. However, previous works have demonstrated that the ϵ-metric is a poor indicator of grasp stability in the real world [4]. In this work, we demonstrate that our novel feature representations significantly outperform the ϵ-metric in predicting the success of a grasp in reality.

More recently, many supervised learning approaches to grasp evaluation or planning have formulated the representation of a grasp as a low-dimensional projection of a full gripper configuration at the end of grasp execution, in order to keep the grasp planning problem tractable and allow for easier manual annotation [5], [6], [7], [8]. A major drawback to these lower dimensional representations is the problem of scalability to more complicated robotic hands. While Lenz et al suggest the oriented rectangle serve as a lower dimensional representation of the opposing ends of a multi-fingered hand, this suggestion fails to scale to hands more anthropomorphic in nature. Another drawback lies in how this feature representation constrains the approach vector of the hand, forcing the robot to execute each grasp in a direction orthogonal to the plane of the image. This is a key limitation, as it possibly prevents the grasp approach vector from aligning with the principal axis of the object, a grasping strategy humans commonly employ [9].

There are a number of existing works that define alternative feature representations based off depth, similar to our feature representation proposals [10], [11], [12], [13]. Andrea ten Pas et al represent each grasp candidate as the geometry of the object along the closing plane of a 1-DOF gripper, encoding the resultant two-dimensional image using the HOG descriptor [14]. Gualteri et al compute various 2D projections of the object to be grasped along three directions: the hand approach vector, the axis of major curvature of object surface, and the vector orthogonal to the hand approach vector [10]. Kappler et al take a similar approach, computing a 2D projection of the object onto a plane defined by the intersection between the object and the hand approach vector [12]. Varley et al represents a grasp candidate as a set of simulated RGB-D patches at each palm and fingertip location [13]. Of the aforementioned feature representation, the approach in [13] is closest to ours.

III. FEATURE REPRESENTATION DEFINITIONS

We now formally define our feature representations and their corresponding implementations. Both techniques are based on distance calculations from a grid of points on the hand surface, but differ in how they represent the object. The Signed Distance Feature converts the object to a Truncated Signed Distance Function, the Wedge Distance Feature represents the object as a uniformly distributed set of points on the object's surface. This conversion only has to happen once; we transform the hand into object space to do the distance calculations.

The Signed Distance Feature is computationally efficient, but does not restrict the distances to those points on the object that the hand will actually encounter. The Wedge Distance Feature restricts the distance calculation to a "wedge" radiating from the grid point — the points on the object the hand is likely to encounter. This is computationally more expensive, and there may be no closest point in that direction.

We first discuss how the grids on the hand are generated, then the two feature representations. We close with possible extensions to the basic feature representations.

A. Generating the Hand Grid Points

This is a general-purpose algorithm (see GENERATE GRID()) for placing a grid of points on a part of the robot hand (eg a finger-tip or the palm). The algorithm essentially projects a grid onto the hand geometry.

The number of grids, their resolution, and where they are placed are user-dependent. We assume that, for each grid, the user provides the center point and two vectors that define a tangent plane with the desired width, height, and orientation of the grid. From this, a grid of points is generated in the tangent space at the desired resolution (generateGrid). Each point is then projected onto the robot geometry in the direction of the tangent plane's surface normal (morphPoints). This defines a set of points (from the robot geometry). This projection only has to happen once.

For the results presented here, we used a 20×20 grid sampling rate, one grid for each finger link, placed so that it spans the link, and one grid for the palm (Barrett and Shadow, see Figure 2). We calculated these values shortly before the fingers made contact.

Algorithm GENERATE GRID()

```
Input:
    Mesh $m_{hand}$
    $X, Y$ Resolution
    $N$ User specified points $p_l, \vec{x}_l, \vec{y}_l, l \in [1, N]$
Output:
    Grid Points $link^l_{i,j}, l \in [1, N], i \in [1, X], j \in [1, Y]$
1: for $i$ in $[1, N]$ do
2:     $g_{plane} \leftarrow$ generateGrid($m_{hand}$,$p_i$,$\vec{x}_i$,$\vec{y}_i$, $X$, $Y$)
3:     $link^l_{i,j} \leftarrow$ morphPoints($link^l_{i,j}$, $m_{hand}$, $g_{plane}$)
4: return $\{link_{i,j}\}_{1,N}$
```

B. Signed Distance Feature

The signed distance feature requires the pre-computation of a Truncated Signed Distance Field (TSDF) $d : \mathbb{R}^3 \to \mathbb{R}$ of the object. This function returns the (approximate) distance of a point $x \in \mathbb{R}^3$ to the nearest point on the surface of the object in constant time.

The TSDF is defined on a 3D volume grid placed around the object. The object is placed at the origin and the grid size in each direction is defined by the maximum span of the hand, or by the object's size in that direction plus 1/5 of the hand span, whichever is larger. This ensures that there is enough "padding" around the object to support reliable distance calculations (if the point x lies outside of the grid it is projected to the closest point on the grid).

There is a trade-off between accuracy and memory storage requirements. For this paper we used volumes with a field size of $60cm \times 60cm \times 60cm$, using a voxel size of 0.0025m. Note that standard hierarchical techniques (e.g., oct-trees) could be applied here.

To calculate the feature at each time point, we transform the grid points based on the current hand configuration (matrix transformation M_l for each link), then map the grids to object space using the hand pose M relative to the object (Transform). The full algorithm is shown in SIGNED DISTANCE FEATURES().

It is straightforward to implement this on a GPU using 3D texture lookup.

Algorithm SIGNED DISTANCE FEATURE()

```
Input:
    Meshes $m_{hand}$, $SDF_{obj}$
    Grid points $link^l_{i,j}$
    Matrices $M_l$, $M$
    SDF of object $f$
Output:
    Signed Distances $hand_{dists}$
1: $hand_{dists} \leftarrow []$
2: for $p_{i,j}$ in $link^l_{i,j}$ do
3:     $p_{i,j} \leftarrow$ Transform($M_l$, $M$ $link^l_{i,j}$)
4:     for $p$ in $p_{i,j}$ do
5:         $link_{dists}$.append($f[p_{i,j}]$)
6:     $hand_{dists}$.append($link_{dists}$)
7: return $hand_{dists}$
```

C. Wedge Distance Features

Mathematically, we define the wedge as a cone of rays emanating from the grid point, with the spread of the cone set so that the width of the cone at 1/2 of the hand span is the width of the link. In our implementation we model this as a set of 30 rays, sampled uniformly from the cone and intersected with the object. The algorithm is identical to the SDF algorithm, except we replace the f calculation with a set of ray casts (30 total in our implementation). If a ray fails to hit the object we mark that as a miss, and store a value of -1 at the location of the grid.

These features could be implemented on the GPU by rendering the object from the viewpoint of the grid points.

D. Potential alternatives and additions

The first alternative we propose is adding orientation information — how the surface normals of the grid points on the robot hand are aligned with the object (dot product). We already have the surface normals for the robot geometry. For the SDF representation the normals are easily calculated as the (normalized) gradient of the Signed Distance Function. For the Wedge representation we can return that information along with the ray intersect/store the normal with the object points.

While a temporal stream of the features was unnecessary for either of the validation tasks, utilizing a temporal stream of these features may prove useful when computing finger control strategies as in [15]. Capturing temporal data as the hand is closed is simply a matter of recording a sequence of grid values, similar to creating a video.

For the tests in this paper we found that including relative orientation information was not useful, as it increased convergence time and reduced prediction accuracy.

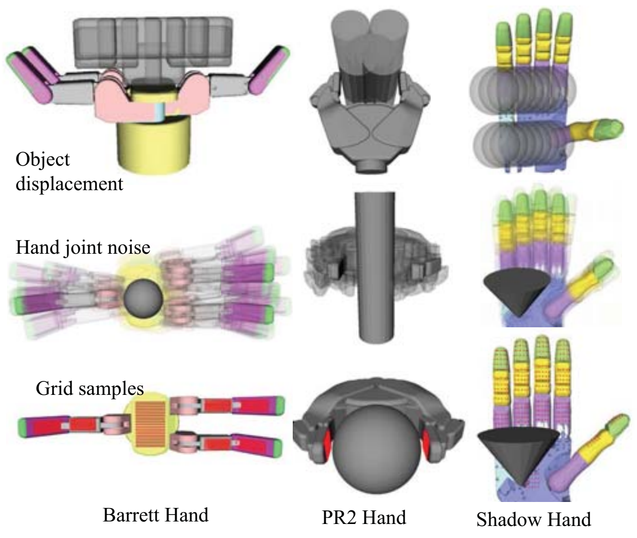

Fig. 2: Hands (Barrett, PR2, Shadow) and added noise for the validation test. Top row, moving the object, middle row, joint angle variation, bottom row, grid samples.

One additional piece of information that could be added to this representation is the *relative* locations of the grids with respect to each other (and to the object as a whole). This, combined with the surface-orientation map, would contain information similar to the current force-closure metrics.

IV. VALIDATION

In this section we describe the motivations and methods behind our validation tests.

Our first validation test focuses on behavior under kinematic noise, varying hand morphology, ability to mimic an exiting grasp metric, and suitability for machine learning. For this test we used simulation data as ground truth, since we are not using this test to show that our features are useful for predicting behavior "in the real world".

Our second validation test is a classic grasp prediction test (success/failure) using real data. We varied both the objects and the relative hand-object pose in a systematic way. Because we know the full geometric relationships of the grasp, we can calculate the features.

For both tasks we apply off-the shelf separate CNNs (with near-identical architectures) for each robotic hand to both sets of features, tested using 3-fold cross validation.

A. Validation test 1: Comparison to existing metric

We describe the hands, objects, and grasps used, the noise model used to generate variations, and the deep learning model. Note that the goal is to learn the *expected* value of the ϵ-metric — mean and standard deviation.

1) Hands, objects, and grasps: To demonstrate the hand-agnostic nature of our feature representations we conduct three experimental simulations using hands with varying kinematic structures: The fully anthropomorphic 20-DOF Shadow Hand, the opposable 4-DOF Barrett Hand, and the PR2 Gripper, (see Figure 2). We placed grids on each link: 2 for the PR2, 7 for the Barrett, and 15 for the Shadow Hand.

The objects (shown in Figure 3) used to perform our experiments were selected from a set of fundamental shapes, with three sizes for each one.

Our grasps were specified as pre-shapes (the grasps shown in Figure 2). To generate the final pose, the grasp is executed in GraspIt!, which essentially closes the fingers at equal speeds until they make contact with the object. This does not reflect "real life", because the object would normally shift when grasped, but it suffices for these tests. We calculate the ϵ-metric for the final grasp, and use the initial hand configuration (pre-noise) to compute our representations.

2) Noise generation: For all three hands we introduced noise by changing the pose of the hand relative to the object. For the Barrett and Shadow hands we also introduced noise into the joints.

For pose noise we introduced both transitional and orientation noise. The absolute amount of transitional noise depended on the hand. For orientation we added noise to the pitch, yaw, and roll, each sampled from a Gaussian with covariance of $\sigma = 0.1$ radians. The maximum translation noise varied by gripper size. For the PR2 gripper, the position of the object varied between -2.5 cm to 2.5 cm side to side, and 5 cm extending outward from the gripper. The Barrett Hand used a range between -5.0 to 5.0 cm side to side, with 5 cm outward. For the Shadow Hand, -2 cm to 2 cm side to side, with a range of 5 cm over the length of the hand. In all cases, we sampled a Gaussian with covariance set to $\sigma = 0.02$ of the available range. We used a total of 20 unique translations and 64 unique orientations for a total of 1280 pose variations.

For the joint angles we added noise from a Gaussian with covariance of σ_{JA} = 0.1 radians to each joint. We generated a total of 150 unique variations.

3) Training Data: The resulting training set consists of one grasp for each object for each of the three hands, with approximately $20 \times 64 + 150 = 1430$ variations for each grasp. For each grasp variation we calculated the corresponding values for our features and the ϵ-metric .

4) Deep Learning Model: A Convolutional Neural Network (CNN) was trained on each feature representation. We choose convolutional models due to the nature of each feature representation, as neighboring distance values for each patch of distances are locally correlated. The model consists of three Convolutional Rectified Linear Layers, followed by two dense layers. Mean Squared Error loss functions were used for each regression task.

The inputs to each channel of the CNN for the signed distance features consist of a 20x20 array of distances (one for each grid). For the wedge distance feature, each grid point generated 30 distances, for an overall input size of 600x20. The number of input channels corresponds to the number of user-specified seed points (links), resulting in 2, 7, and 15 channels for the PR2 Gripper, Barrett Hand, and Shadow Hand respectively. The mean μ_ϵ and the variance σ_ϵ^2 of the resultant distribution are the continuous outputs of the CNN.

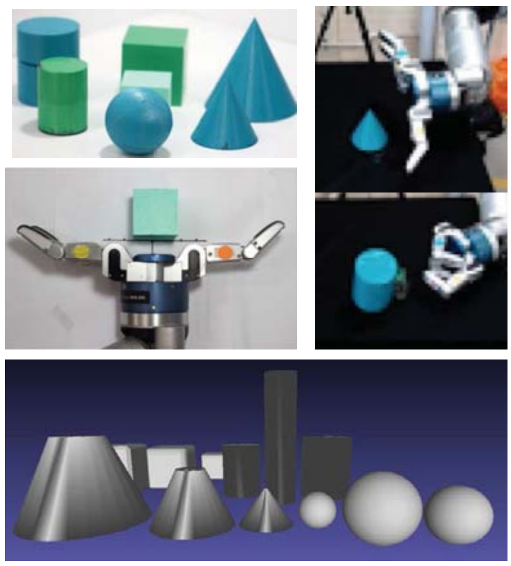
Fig. 3: Objects used in the real-world test (top) and the simulated tests (bottom). Range of grasp variation in the real world tests is shown on the top right.

The training hyperparameters are standard, consisting of 150 epochs, a batch size of 32, and a learning rate of 0.001. The architecture to the model can be seen at Figure 4. To deal with the right skewness of the ϵ-quality distributions, we normalize the ϵ-labels with a logarithmic function. All testing was done with 3-fold cross validation.

B. Validation test 2: Real-world Grasp Success Prediction

We describe the physical data capture setup, the generated grasps, and the deep learning model.

1) Physical data capture: We used a 7-DOF WAM arm equipped with the Barrett Hand to generate our grasp test, with the objects shown in Figure 3. A Constrained Bi-directional RRT (CBiRRT) [16] path planner is used to move the end effector to the starting pose for each grasp.

To ensure consistent placement of the object we used a north-south magnet on the bottom of the object. A matching magnet under the table was automatically lifted into place to “snap” the object into place, then pulled away before the grasp test.

After grasp execution, the WAM arm lifts the object and performs a rotation of the wrist joint for 15 seconds. Only grasps that pass this shake test are marked as successful.

We recorded the Barrett hand joint angles (sampled at 0.2). The signed distance and wedge distance features are computed based off of the joint angles collected at the last sample before contact is made with the object.

2) Grasp generation and objects: The 7 objects we used are shown in Figure 3. To generate our grasp test set we uniformly varied the translation and orientation of the hand prior to closing the fingers. We applied 12 translation samples at a uniform height with respect to the object and six orientations with ± 0.3 radians of roll, ± 0.3 radians yaw induced orientations, and 0.3 and 0.6 yaw induced orientations, for a total of approximately 84 variations per object. (exact numbers in Table I, note that we had 2 each of the cubes, cylinders and cones, but only one sphere). Some of the grasp-object variations were discarded due to symmetries or missing data capture. We generated a total of 572 grasps, of which 210 were successful.

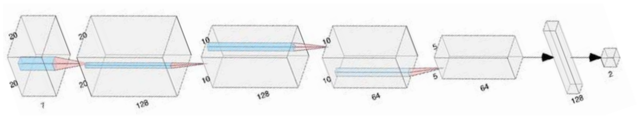

Fig. 4: The structure of our CNN. We use three convolutional layers, followed by two dense layers.

3) Training Data: We use two methods to expand our real-world data set: Adding noise to the pose and fine tuning a model trained off GraspIt! simulations.

Adding noise: We utilize data augmentation to scale from our base dataset of size 572 grasps to a dataset of 7056 grasps. We inject Gaussian noise into the pose of the hand, with $\sigma_{XYZ} = 0.005$ m and $\sigma_{RPY} = 0.05$ radians.

Fine Tuning: We pre-train on the simulation data from the previous validation task.

4) Deep Learning Model: We use the same model architecture described in the previous task (Section IV-A.4). Our only change is to switch to a binary output representing the success or failure of a grasp.

Utilizing pre-training data: After training the model on the simulation data mentioned above, the Convolutional Rectified Linear Layers are frozen and the fully connected layers are retrained using just the real-world data.

V. RESULTS

A. Validation test 1: ϵ-metric

Table II illustrates the performance of our ϵ-metric approximator. For reference, the average epsilon values over the entire dataset of grasps executed were μ_ϵ = 0.0239, 0.0193, and 0.0383, for the Barrett Hand, the PR2 Gripper, and the Shadow Hand, respectively.

B. Validation test 2: Real-world Grasp Success Prediction

Prediction results are summarized in Table I. We try four different learning models, and also show results with the standard ϵ_{GWS} model. The first model uses the predicted ϵ-metric values. The remaining three models evaluate the effect of using just the original 588 grasps, adding noise, and adding noise and pre-training.

1) The Epsilon Metric: For comparison we include the ϵ-metric prediction results for the success of each grasp executed in the real world. These values were calculated in GraspIt! from the predicted grasp closure. Following the work of Allen [4] and DexNet [17], we choose an ϵ_{GWS} threshold of 0.002, where if the computed ϵ value is greater than 0.002 the grasp is marked as successful.

SDF/Wedge	Real-Life Attempts		Prediction Accuracy				
Shapes	**Successes**	**Failures**	ϵ_{GWS}	**GQ Model 1**	**GQ Model 2**	**GQ Model 3**	**GQ Model 4**
Cubes	70	82	65%	61%/63%	64%/57%	74%79%	**80%/84%**
Cones	17	152	80%	82%/76%	80%/54%	83%/93%	**83%/92%**
Cylinders	106	75	53%	55%/59%	57%/49%	77%/92%	**84%/94%**
Ellipses	17	53	42%	40%/43%	63%/59%	76%/79%	**83%/82%**

TABLE I: Model prediction accuracy for our four learned models (real world data) by object type. Results are given as SDF/Wedge percentage successful prediction. Left: Number of successful/failed trials per shape.

TABLE II: Simulation Mean Absolute Error

SDF/Wedge	MAE μ_ϵ	MAE σ^2_ϵ
Shadow Hand	3.91×10^{-3}/ 2.57×10^{-3}	1.19×10^{-4}/ 3.61×10^{-5}
PR2 Gripper	9.2×10^{-4}/ 1.03×10^{-3}	2.86×10^{-5}/ 7.42×10^{-5}
Barrett Hand	1.45×10^{-3}/ 3.29×10^{-3}	3.19×10^{-5}/ 9.2×10^{-6}

2) GQ Model 1: This model is similar to the Epsilon metric, except we replace the calculated ϵ value from the ϵ_{GWS} model with our learned one (Section IV-A.4). This is a measure of how well our (learned) ϵ noise model works as a substitute.

3) GQ Model 2: We train the model *just* on the data collected from real life, mapping the feature representations collected from the *near contact stage* to the outcome of the grasp. The training dataset in this case was only 572 grasps.

4) GQ Model 3: In this model we train on the real-world data with the additional noise samples (Section IV-B.3).

5) GQ Model 4: In this model we utilize both the pre-training (Section IV-B.4) and the additional noise samples.

VI. Discussion

The accuracy rates for the various approaches described in the results section are shown in Table I. We make the observation that the performance of the wedge distance features correlates strongly with the amount of training data available, as noticed in Model 2, with an average success rate of 53%. This is reasonable, as the feature representation itself is significantly less locally correlated than the Signed Distance Features. It also suffers from severe sparsity, in that wedges from the hand that do not intersect with the object are meaningless, meaning more data is required to learn something meaningful. While we represent ray 'miss' distances with a value of -1, we propose alternative wedge feature representations with variable-size wedge angles such that each cone projected from an initial point does not extend past the surface of the object.

We observe Model 4 on average outperforms all other models. The fact the data acquired from simulation can so easily be transferred into the real world implies our feature representations are far more robust to domain adaptation than other features extracted from RGB images.

From an analysis of the trials, we note the main failures of the ϵ_{GWS} to be the lack of consideration for the inherent instability of the ellipsoid and cylinders, as despite the correct initial contacts being made on the shapes, the speed at which the fingers closed resulted in the object toppling. An approach to solve this would be to use the temporal stream of our features to train a model capable of predicting whether the feature gradients are likely to result in unstable object behaviors.

VII. Conclusion

In this work we have proposed two novel feature representations for near-contact grasping evaluation and validated them both in simulation (metric stability) and the real world (grasp prediction).

References

[1] J. K. Salisbury and B. Roth, "Kinematic and force analysis of articulated mechanical hands," *Journal of Mechanisms, Transmissions, and Automation in Design*, vol. 105, no. 1, pp. 35–41, 1983.

[2] C. Ferrari and J. Canny, "Planning optimal grasps," pp. 2290–2295. [Online]. Available: http://ieeexplore.ieee.org/document/219918/

[3] V.-D. Nguyen, "Constructing force-closure grasps," *The International Journal of Robotics Research*, vol. 7, no. 3, pp. 3–16, 1988.

[4] J. Weisz and P. K. Allen, "Pose error robust grasping from contact wrench space metrics," in *ICRA*, 2012, pp. 557–562.

[5] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," in *ICRA*, 2016, pp. 3406–3413.

[6] J. Redmon and A. Angelova, "Real-time grasp detection using convolutional neural networks," in *ICRA*, 2015, pp. 1316–1322.

[7] S. Kumra and C. Kanan, "Robotic grasp detection using deep convolutional neural networks," in *IROS*, 2017, pp. 769–776.

[8] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 705–724, 2015.

[9] R. Balasubramanian, L. Xu, P. D. Brook, J. R. Smith, and Y. Matsuoka, "Physical human interactive guidance: Identifying grasping principles from human-planned grasps," *Springer Tracts in Advanced Robotics*, vol. 95, no. 4, pp. 477–500, 2014.

[10] M. Gualtieri, A. ten Pas, K. Saenko, and R. Platt, "High precision grasp pose detection in dense clutter," in *IROS*, 2016, pp. 598–605.

[11] A. ten Pas and R. P. Jr., "Localizing handle-like grasp affordances in 3d point clouds," in *ISER*, ser. Springer Tracts in Advanced Robotics, vol. 109. Springer, 2014, pp. 623–638.

[12] D. Kappler, J. Bohg, and S. Schaal, "Leveraging big data for grasp planning," in *ICRA*, 2015, pp. 4304–4311.

[13] J. Varley, J. Weisz, J. Weiss, and P. Allen, "Generating multi-fingered robotic grasps via deep learning," in *IROS*, 2015, pp. 4415–4420.

[14] A. ten Pas and R. Platt, "Using geometry to detect grasp poses in 3d point clouds," in *Robotics Research*. Springer, 2018, pp. 307–324.

[15] Y. Ong, J. Morrow, Y. Qui, K. Gupta, C. Grimm, and R. Balasubramanian, "Near-contact grasping strategies from awkward poses: When simply closing your fingers is not enough," in *IROS 2019, under review*.

[16] D. Berenson, S. S. Srinivasa, D. Ferguson, and J. J. Kuffner, "Manipulation planning on constraint manifolds," in *ICRA*, 2009, pp. 625–632.

[17] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Y. Goldberg, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," *CoRR*, vol. abs/1703.09312, 2017.