Recognizing Geometric Constraints in Human Demonstrations using Force and Position Signals

Guru Subramani[1], Michael Gleicher[2], and Michael Zinn[1]

***Abstract*—This paper introduces a method for recognizing geometric constraints from human demonstrations using both position and force measurements. Our key idea is that position information alone is insufficient to determine that a constraint is active and reaction forces must also be considered to correctly distinguish constraints from movements that just happen to follow a particular geometric shape. Our techniques can detect multiple plane, arc, and line constraints in a single demonstration. Our method uses the principle of virtual work to determine reaction forces from force and position data. It fits geometric constraints locally and clusters these over the whole motion for global constraint recognition. Experimental evaluations compare our force and position constraint inference technique with a similar position only technique and conclude that force measurements are essential in eliminating false positive detections of constraints in free space.**

***Index Terms*—Learning from Demonstration, Recognition, Programming by Demonstration, Constraint Inference, Recognizing Geometry, Constraint Theory, Natural Demonstrations**

I. Introduction

GEOMETRIC constraints, such as the plane of a whiteboard during erasing or the arc of an opening door frequently occur in tasks. These motions are different from unconstrained motions as they restrict the available degrees of freedom, and require different control approaches for execution on a robot. Recognizing constraints in human demonstrations is therefore valuable in robot programming by demonstration (PbD) [1] as ultimately these demonstrations will be used for robot programming. While current PbD work uses kinesthetic demonstrations (i.e., users showing the task by moving the robot), newer approaches consider a more natural input method of recording the user's movement as they perform the task with their hands or a tool. Our goal is to provide methods for interpreting such natural demonstrations, specifically to identify the geometric constraints involved.

This paper provides an approach for recognizing geometric constraints in human demonstrations of tasks. Specifically, our methods identify plane, arc and line constraints in recorded trajectories of tool positions and forces from human movements. Our key insight is that position information alone is insufficient to determine that a constraint is active. Therefore, reaction forces must be considered to correctly distinguish constraints from movements that just happen to follow a particular geometric shape.

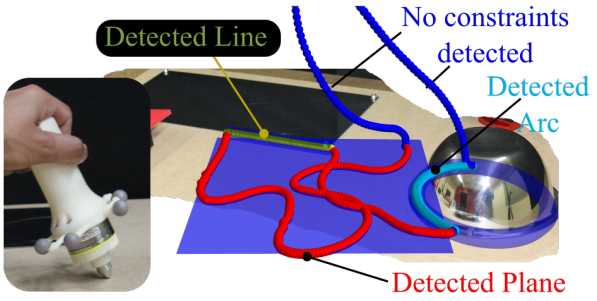

Fig. 1. **Demonstration containing all constraints in the same plane. The demonstration involves moving the Constraint Saber along the edge of the steel sheet, the planar surface and the edge of the bowl consecutively. Notice that in this demonstration, the demonstrator transitions between constraints without lifting the Constraint Saber off of the test bed. Our approach correctly associates samples to their respective constraint. Left - Shows the Constraint Saber demonstration tool.**

Constrained motion is different from unconstrained motion: reaction forces from the constraint guide the movement along a permissible path. For example, simply moving in a straight line *feels* different from drawing a line using a straight edge because the edge can be felt pushing with its reaction forces. Prior approaches to constraint recognition use only kinematic information. Therefore, they cannot adequately disambiguate between unconstrained movements that happen to have the correct shape from movements whose shape are caused by a constraint, such as distinguishing line constraints (e.g., following a straight edge or pulling a drawer) from more general planar movements that happen to be straight.

Our approach uses position and force information together to recognize the constraints active in a demonstration and exploits the physical properties of constrained motions, specifically the principle of virtual work. Force measurements are resolved into reaction forces and friction forces to build constraint frames for all movement samples where applied forces are present. Local descriptors of the constraints are identified at each time instance and clustered over the entire motion to find global constraints.

Our method recognizes plane, line, and arc constraints in a demonstration, identifying the timing and geometric parameters of each. Experimental validation of the approach was

Manuscript received: September, 10, 2017; Revised November, 27, 2017; Accepted December, 29, 2017.

This paper was recommended for publication by Editor Dongheui Lee upon evaluation of the Associate Editor and Reviewers' comments. This work was supported in part by NSF award 1208632 and the University of Wisconsin-Madison Office of the Vice Chancellor for Research and Graduate Education with funding from the Wisconsin Alumni Research Foundation.

[1]Guru Subramani and Michael Zinn are with Department of Mechanical Engineering, University of Wisconsin - Madison, Madison 53706, USA [gsubramani|mzinn]@wisc.edu

[2]Michael Gleicher is with the Department of Computer Sciences, University of Wisconsin - Madison, Madison 53706, USA gleicher@cs.wisc.edu

Digital Object Identifier (DOI): see top of this page.

performed through numerous demonstrations recorded using a tool instrumented with motion capture and force sensing, demonstrating its superiority to a method that uses position measurements alone.

II. Related Work

In robotics, geometric constraint interaction has been explored under different contexts. In situations where a robot is required to recognize constraints autonomously, Ortenzi et al. [2] have explored techniques that use the null space of allowable velocities during contact. Work exploring how robots can learn complicated contact constraints autonomously have been explored by Levine et al. [3]. However, they do not explore the recognition of constraints in human demonstrations.

Another approach to constraint recognition in human demonstrations is C-LEARN by Perez-D'Arpino [4]. In C-LEARN, constraints are learned through multiple human demonstrations. C-LEARN focuses on learning different grasp approaches and using keyframes to determine their associated permissible constrained motions such as movement in a plane or along an axis. Recent work using CHAMP, a parametric model based change point detection system which uses kinematic information only, have been proposed by Neikum et al [5]. However these methods do not incorporate force interaction in their models.

Demonstrations of trajectories under the influence of constraints, represented in the context of null spaces, have been explored by Lin et al. [6]. This method utilizes kinematic information to identify constraints but does not handle multiple types of constrained motions in the same trajectory.

Contact detection in compliant motions has been explored by Meeussen et al. [7] In their work the notion of principle contacts (PC) are identified between two different polyhedral objects. Demonstrations containing compliant motions(constrained motions) are modeled as multiple PCs and identified using particle filters. They determine the contact state of the tool with a known object in the environment over the motion. While local contact states are determined, this method does not provide estimates of the geometry of the environment.

Many methods are available to fit geometry to points such as the Hough transform [8], but these methods are not suitable for trajectories because they do not take advantage of the ordered continuous stream of points and there are not enough points to reliably fit geometry with a sample consensus based method.

To summarize, the above methods do not either incorporate force measurements or explicitly determine the geometric constraints in the motion. This paper incorporates force measurements for global explicit constraint inference.

III. Approach

Our approach uses both force and position trajectories and the principle of virtual work to recognize geometric constraints in human demonstrations. An overview is shown in figure 2. The ordered steps are as follows:

1. Preprocess position and force signals and transform force signals to the global frame of reference.
2. Extract *constraint frames* by resolving reaction force and friction force.
4. Fit local plane, line and arc constraints using windows of constraint frames.
5. Cluster similar plane, line and arc constraints over entire trajectory.
6. Disambiguate and select from plane, line and arc constraints.
7. Fit recognized constraints.

Prior to describing the details of our approach, the underlying theory behind point contact constrained motions are described next.

IV. Physics based Constraint Modeling

When forces are applied to a constrained body, it experiences reaction forces from these constraints. The principle of virtual work resolves forces into constraint reaction forces and forces along the direction of motion such as frictional forces. Constraint reaction forces must be orthogonal to the motion of the object. While conceptualizing how point contact reaction forces act on a body is intuitive, it is not trivial to explain why a particular constraint produces reaction forces in particular directions. We develop this idea below.

A. Principle of Virtual Work for Constraint Interactions

Consider an object constrained at a point $\mathbf{p}=(x,y,z)\in\mathbb{R}^3$ by a constraint $\mathbf{\Phi}:\mathbb{R}^3\Rightarrow\mathbb{R}^k$ such that $\mathbf{\Phi}(\mathbf{p})=\mathbf{0}$.[1]

Let an applied force $\boldsymbol{f}\in\mathbb{R}^3$ act on the body. If we assume that inertial forces are negligible, the reaction force $\boldsymbol{f_r}$ and friction force $\boldsymbol{f_\mu}$ are given as:

$$\boldsymbol{f}=\boldsymbol{f_r}+\boldsymbol{f_\mu} \tag{1}$$

If the velocity of the point $\mathbf{p}$ moving under constraint equations $\mathbf{\Phi}(\mathbf{p})=\mathbf{0}$ is $\boldsymbol{v}\in\mathbb{R}^3$, then from the principle of virtual work [9]:

$$\boldsymbol{f_r}\cdot\boldsymbol{v}=0 \tag{2}$$

Assuming that the friction force opposes the contact velocity, and thus is collinear to the velocity vector, force $\boldsymbol{f}$ can be resolved as friction force $\boldsymbol{f_\mu}$ and reaction force $\boldsymbol{f_r}$:

$$\boldsymbol{f_\mu}=(\boldsymbol{f}\cdot\hat{\boldsymbol{v}})\hat{\boldsymbol{v}} \tag{3}$$

$$\boldsymbol{f_r}=\boldsymbol{f}-\boldsymbol{f_\mu} \tag{4}$$

where $\hat{(\cdot)}$ represents a unit vector. Using the velocity, force and position, a constraint frame F_p is defined, with basis vectors $\hat{\boldsymbol{v}}$, $\hat{\boldsymbol{f}}_{\boldsymbol{r}}$ and $\hat{\boldsymbol{v}}\times\hat{\mathbf{f}}_{\mathbf{r}}$ at position $\boldsymbol{p}$ of the point contact.

If the variation of $\mathbf{p}$ is $\partial\mathbf{p}$ then the admissible variations of $\mathbf{p}$ under $\mathbf{\Phi}(\mathbf{p})=\mathbf{0}$ must satisfy:

$$\partial\mathbf{p}^{\mathbf{T}}\frac{\partial\mathbf{\Phi}}{\partial\mathbf{p}}=\mathbf{0} \tag{5}$$

Equation 2 can be rewritten in variational form:

$$\partial\mathbf{p}^{\mathbf{T}}\mathbf{f}_{\mathbf{r}}=\mathbf{0} \tag{6}$$

[1]We denote vectors and and matrices with bold symbols.

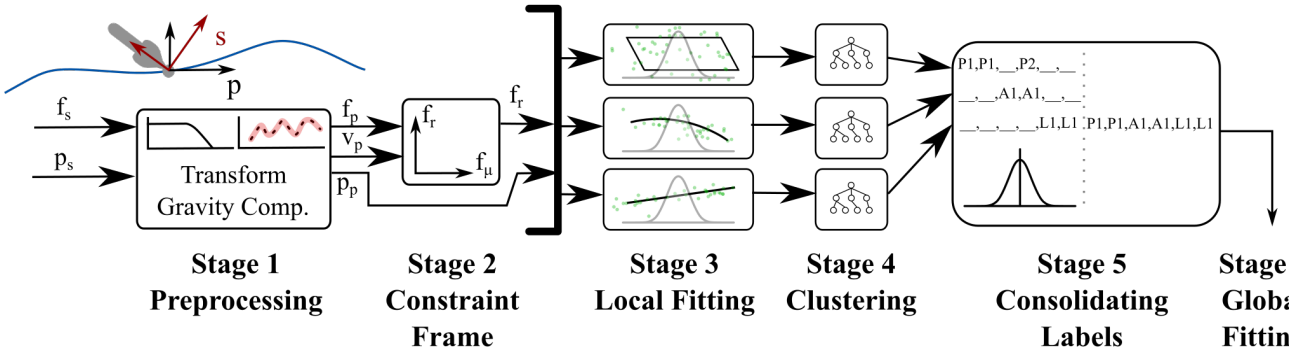

Fig. 2. **Schematic representation of our method. Stage 1 - Signals are preprocessed, Stage 2 - Constraint frames are defined where applicable over the entire signal, Stage 3 - Local geometry is fit at every constraint frame, Stage 4 - Local geomery is clustered for each constraint type, Stage 5 - Cluster labels are consolidated, Stage 6 - Geometry is fit globally.**

Using Lagrange multiplier theorem, similar to the approach used by Huag [10], Combining equations 5 and 6:

$$\frac{\partial \mathbf{\Phi}}{\partial \mathbf{p}}^T \lambda + \mathbf{f_r} = \mathbf{0} \tag{7}$$

where $\lambda \in \mathbb{R}^k$ are the Lagrange multipliers. Note: $\frac{\partial \mathbf{\Phi}}{\partial \mathbf{p}}$ is size $k \times 3$ and λ is size $k \times 1$.

Equation 7 is important because it provides a direct relationship between the constraint reaction force $\mathbf{f_r}$ and the constraint equations $\mathbf{\Phi}(\mathbf{p}) = \mathbf{0}$. As shown in the following section, any set of constraint equations can essentially be 'plugged in' equation 7 to retrieve the permissible reaction forces. Note that the Lagrange multipliers λ take any value that satisfy equation 7.

B. Reaction forces for Geometric Constraints

The geometric constraints considered are line, arc and plane constraints. When a rigid body is point constrained to one of these constraints, it experiences reaction forces and a reduction in its degrees of freedom. The geometric models used here are common and can be found in most textbooks on vector calculus [11].

1) Plane Constraint: The equation of a plane can be represented by:

$$\mathbf{\Phi}(\boldsymbol{p}) = ax + by + cz + d = 0 \tag{8}$$

where a,b,c,d $\in \mathbb{R}$

Using equation 7 and 8, the relationship between the constraint equations and its corresponding reaction forces are:

$$\frac{\partial \mathbf{\Phi}}{\partial \mathbf{p}} = \begin{bmatrix} a & b & c \end{bmatrix}$$

$$\mathbf{f_r} = \begin{bmatrix} a & b & c \end{bmatrix}^{\mathbf{T}} \begin{bmatrix} \lambda_1 \end{bmatrix}$$

The normal $\boldsymbol{n}$ of the plane defined by equation 8 is $< a,b,c >$ This immediately leads to the result:

$$\hat{\mathbf{f_r}} = \hat{\mathbf{n}} \tag{9}$$

Equation 9 shows that *the constraint reaction force of the plane is always directed along the normal to the plane.*

2) Line Constraint: The line constraint can be represented by the symmetric equations of a line:

$$\frac{x - x_0}{l_1} = \frac{y - y_0}{l_2} = \frac{z - z_0}{l_3} \tag{10}$$

where the tangent vector $\boldsymbol{t} = (l_1, l_2, l_3) \in \mathbb{R}^3$ is the direction of the line and $p = (x_0, y_0, z_0) \in \mathbb{R}^3$ is a point on the line. Rearranging equation 10:

$$\mathbf{\Phi}(\boldsymbol{p})_1 = (x - x_0)\, l_2 - (y - y_0)\, l_1 = 0, \tag{11}$$

$$\mathbf{\Phi}(\boldsymbol{p})_2 = (z - z_0)\, l_2 - (y - y_0)\, l_3 = 0 \tag{12}$$

Applying equation 7 to the line equations 11 and 12, the constrained reaction force contributed by this constraint is given by:

$$\mathbf{f_r} = \begin{bmatrix} l_2 & -l_1 & 0 \\ 0 & -l_3 & l_2 \end{bmatrix}^{\mathbf{T}} \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} \tag{13}$$

Equation 13 provides the permissible values of $\boldsymbol{f_r}$. As $\boldsymbol{f_r}$ is a sum of two linear independent vectors, *the constraint reaction force spans the plane perpendicular to the tangent of the line.*

3) Arc Constraint: The arc constraint is represented by the intersection of the sphere and plane that contain the arc.

$$\mathbf{\Phi_1}(\boldsymbol{p}) = (x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 - r^2 = 0 \tag{14}$$

$$\mathbf{\Phi_2}(\boldsymbol{p}) = ax + by + cz + d = 0 \tag{15}$$

where the center and radius of the sphere are $C = (x_0, y_0, z_0) \in \mathbb{R}^3$ and $r \in \mathbb{R}$ respectively. Using the same development:

$$\frac{\partial \mathbf{\Phi}}{\partial \mathbf{p}} = \begin{bmatrix} 2(x - x_0) & 2(y - y_0) & 2(z - z_0) \\ a & b & c \end{bmatrix}$$

$$\mathbf{f_r} = \begin{bmatrix} 2(x - x_0) & 2(y - y_0) & 2(z - z_0) \\ a & b & c \end{bmatrix}^{\mathbf{T}} \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} \tag{16}$$

Similarly to the line case, *the constraint reaction force spans a plane perpendicular to* $\mathbf{n} = \langle a,b,c \rangle$ *passing through the center of the circle.* Note: This plane containing the reaction force vector has a normal vector produced by the cross product: $\frac{\partial \mathbf{\Phi_1}}{\partial \mathbf{p}}$ and $\frac{\partial \mathbf{\Phi_2}}{\partial \mathbf{p}}$.

C. Parameterizations of Planes, Lines and Arcs

For a representation of a constraint to be useful, a uniquely identifiable one must be selected. We list the parameterizations chosen for each constraint type below:

1) Plane Constraint: The definition of the plane, given by equation 8 is parameterized by (a,b,c,d). This is inconvenient as multiple (a,b,c,d) can represent the same plane. Instead, the unit normal vector $\hat{n}$ and the distance of the plane to the origin d are used as the plane constraint representation.

2) Line Constraint: The line is parameterized with the unit tangent vector $\hat{t}$ and the point on the line $\boldsymbol{c_p} \in \mathbb{R}^3$ closest to the origin. This is given by:

$$\boldsymbol{c_p} = \boldsymbol{q} - \frac{\boldsymbol{q}\cdot\hat{\boldsymbol{t}}}{\hat{\boldsymbol{t}}\cdot\hat{\boldsymbol{t}}}\hat{\boldsymbol{t}} \tag{17}$$

where $\boldsymbol{q} \in \mathbb{R}^3$ is any point on the line.

3) Arc Constraint: The arc is parameterized with the center C, radius r and the normal vector $\hat{n}$ of the plane containing the arc.

V. Detailed Overview of Approach

Our approach uses both position and force information to identify local descriptions of the signals and cluster these over the complete task. An overview of our approach is shown in figure 2.

Stage 1: Preliminary signal preprocessing and calibration

To mitigate the effects of sensor noise and to prevent signal aliasing, the position and force measurements are re-sampled and filtered. Gravitational effects are removed by compensating for tool weight. Finally, the tool tip forces are transformed to the global coordinate frame using the orientation measurements.

Stage 2: Determining Constraint Frames across the Demonstration

The purpose of this stage is to determine where the constraints occur in the demonstration. The reaction and frictional forces are computed by using an estimate of the tool tip velocity and force measurements using equations 3 and 4. The reaction force and friction force combine to form the constraint frame F_p, a local description of the constraint at the point of contact. There are two situations where the constraint frame is not well defined:

- At locations where the velocity is approximately zero - the tool tip must be moving and its position measurements free of excessive noise.
- At locations where reaction forces do not exist - the computed reaction force is smaller than sensor noise.

Samples where the reaction force is zero correspond to the absence of a constraint and are ignored. Locations where the velocity is effectively zero are unusable because it is impossible to resolve the measured forces into reaction forces and friction forces.

Stage 3: Determining local constraint geometry

In this stage, all defined constraint types (i.e., plane, line and arc) are fit to the measured position and force data locally at each well defined constraint frame. This provides a local description of the constraint. The proposed fitting methods are all least squares approximations as the errors in our measurements are Gaussian distributed.

1) Fitting Planes: A plane is represented uniquely by a plane normal and a point on the plane. To fit a plane, the normal reaction force and the point of contact are used as the normal and the point on the plane respectively. Over a local window about the sample of interest, gaussian weighted average of the reaction force $\hat{f}_r$ represents the plane normal vector. Note that this is equivalent to applying a gaussian kernel to a signal as done in low pass filtering. Using the local estimate of the normal reaction force and a similar weighted average of the position of the tool tip, a local plane is constructed. For a unique representation of this plane, the distance of the plane d and the unit normal of the plane $\hat{f}_r$ are used.

2) Fitting lines: To fit a line, position measurements need only be used. A gaussian windowed SVD performs the fit (detail in appendix B). The final parameterization of the line is the tangent vector and a point on the line.

3) Fitting Arcs: The arc fitting procedure is accomplished using a nonlinear least squares fit(detail in appendix C). The arc parameters are the center C, the radius r and the normal of the plane $\hat{n}$ containing the arc.

The result from this stage is a stream of labels containing fit parameters for each constraint type for all well defined constraint frame samples over the complete demonstration.

Stage 4: Clustering

Local parameters of each constraint type are clustered over the entire task. The procedure clusters local constraints into larger groups and removes outliers and noise. These larger groups of similar parameters represent global constraints.

Our dataset contains unit vectors, euclidean points and simple scalars called features. Like all clustering methods, if features used to cluster are of vastly different units or type, clustering will provide poor results. Ideally providing a different distance metric for each feature type is preferable. Also, the number of clusters is unknown before hand.

To meet these clustering requirements, a custom method referred to as the *Tiered DBSCAN* clustering algorithm was developed. The conventional DBSCAN [12] [13] algorithm is augmented with a tiered approach so it is suitable for clustering datasets with features of different types including unit vectors, scalars or euclidian coordinates. Alternatively, one might try to weight these features and develop a common metric but determining effective weights for each feature type is challenging.

Tiered DBSCAN clusters in stages, using a new feature at every stage. The clusters from the previous stage are subdivided using a new feature to create sub-clusters. As DBSCAN can remove outliers effectively, many of the sub-clusters from higher tiers are rejected after successive stages of clustering.

Please see algorithm 2 in appendix A and figure 4 for further details. The two parameters for DBSCAN are *min-points* - the number of points required to form a cluster and *eps* - the minimum distance points need to be to successfully cluster. DBSCAN also allows specification of custom distance metrics apart from the L2 norm. Min-points is set considering the smallest possible number of points that would be considered a cluster. Eps is set depending on the measurement noise and associated fitting errors. The order of clustering is chosen empirically but follows intuition: features that cluster more reliably and reject noise should be preferred first. For example, clustering plane normal vectors before plane distance is better because the normal force measurement is more reliable than the plane distance estimate.

Figure 3 shows the features, corresponding metric and DBSCAN parameters at each tier for each type of geometric constraint. The result of the clustering procedure is a signal/stream of labels for each constraint type. Note that clustering for each constraint type is performed individually.

Each cluster recognized by Tiered DBSCAN represents an individual constraint. The clustering procedure creates constraint label streams over time for each constraint type. At locations where constraints were not detected, the stream labels take no value.

Stage 5: Consolidating Plane, Line and Arc labels

The clustering procedure creates three labeled data streams/signals, each corresponding to a constraint type. To consolidate these streams into a single stream, disambiguation heuristics and the *Proximity Weighted Selection Filter* are applied. The heuristics are as follows:

H1 Plane Check: For each plane cluster, the positions and forces are checked for consistency with the recognized plane. A plane using the points contained in the cluster are fit to recover a normal vector. This normal vector is checked for consistency with the average constraint reaction force, using the magnitude of the cross product between the two. This heuristic helps eliminate line constraints detected as planes, as a very straight line would most likely provide a plane inconsistent with the reaction force vector.

H2 Arc Check: Arc labels where a plane and arc are simultaneously detected are ignored. As an arc constraint will most probably have different constraint reaction forces than a plane, the plane is preferred when both are detected.

These heuristics deal with degenerate cases where the constraint is ambiguous. Note that in both these cases either constraint is physically possible but in practice the heuristic captures the real constraint.

After applying the heuristics, the Proximity Weighted Selection Filter combines these three label streams to form one stream of labels. The filter assumes that at a particular sample the probability of a label depends on the density of that label about the sample of interest. The filter uses a gaussian to weight points closer to the sample more than those further away. Please refer to algorithm 1 in appendix A.

	Feature	Metric	eps	min_samples
Plane	Unit Normal Force	Magnitude of cross product	0.02	10
	Plane Distance	L2 norm distance	0.05(m)	10
Line	Unit Tangent Vector	Magnitude of cross product	0.02	50
	Closest Point to Origin	L2 norm distance	0.02(m)	10
Arc	Center	L2 norm distance	0.01(m)	10
	Radius			
	Unit Normal of Plane	Magnitude of cross product	0.02	10

Fig. 3. **Various metrics for each feature type corresponding to each constraint. The features are clustered in decreasing order of reliability to DBSCAN clustering.**

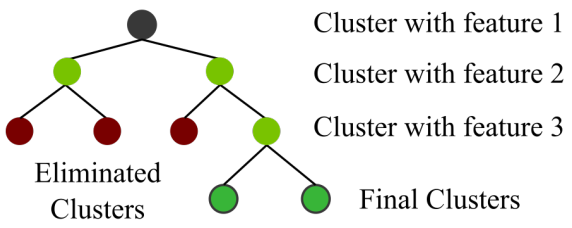

Fig. 4. **Shows the clustering procedure by Tiered DBSCAN. Clustering is performed in stages where each level represents clustering with a specific feature and its corresponding metric. This is useful for datasets with different types of features, like unit vectors and euclidian points.**

Stage 6: Fitting constraints

The points corresponding to each constraint are used to fit their respective constraint. At this point, the constraint type, the parameters associated with the constraints and the associated samples of each constraint are identified.

VI. Experimental Evaluation

The performance of the developed constraint recognition method was evaluated using a dedicated apparatus containing a constraint recognition instrument, the Constraint Saber and a custom testbed. The Constraint Saber has motion capture markers and an ATI Mini40 force torque sensor. While the force torque sensor can measure forces and moments, force measurements are only used. The testbed was designed to include multiple examples of each constraint type (i.e., plane, line and arc) with various levels of frictional resistance varied through the use of different materials and rolling element bearings.

The Constraint Saber, shown in Figure 6 contains a force torque sensor to measure forces and motion capture markers to track its position in space. The Constraint Saber interacts with the environment through a steel tool tip to guarantee point contacts with the environment and ensure that moments are not applied to the environment. Note that all items that move with the constraint saber tool tip have a notch to guarantee consistent contact. Other tools may be attached to the end of the Constraint Saber and this is explored in section VI-D.

	Experiment	TP	Missed	FP					Total	Recall	Precision
		#	#	planes #	lines #	arcs #	total #	total %	#		
Force and Position	**1.** Only plane constraints	116	1	1	1	3	5	4.27	117	**0.99**	**0.96**
	2. Only line constraints	117	0	2	0	0	2	1.71	117	**1.00**	**0.98**
	3. Only arc constraints	103	5	12	1	8	21	19.44	108	**0.95**	**0.83**
	4. Random	208	23	15	0	0	15	6.49	231	**0.90**	**0.93**
	5. Flat Tool	100	0	1	0	2	3	3.00	100	**1.00**	**0.97**
Only Position	**6.** Random	201	30	125	1	15	141	61.04	231	**0.87**	**0.59**

Fig. 5. **Experimental results shown. TP - True Positives, detects the right constraint at the correct location, FP - False Positives, FPs are situations when the method detects a constraint but none exists at the detected location, Precision indicates how well the method is robust against detecting false positives (Larger is better). Recall indicates how well the method identifies the constraints. Missed indicates the number of constraints that were not detected. Total indicates the total number of constraint in the demonstrations. Notice the position only method has a smaller precision when compared to the position and force method.**

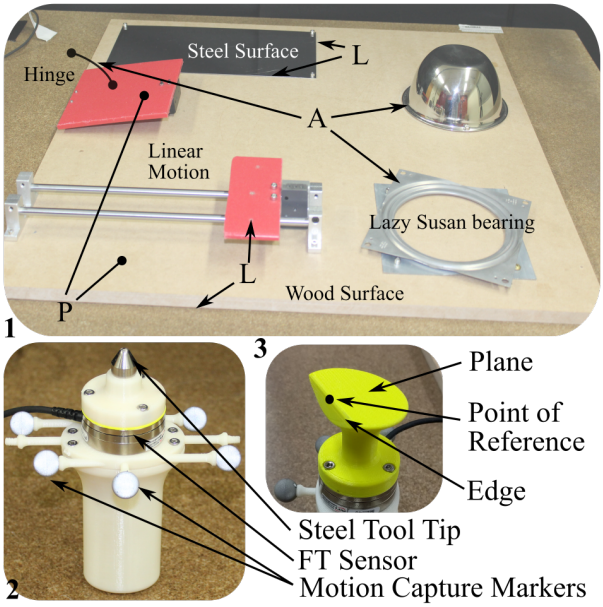

Fig. 6. **1. Shows the test bed containing plane, line and arc constraints. 2. Shows the Constraint Saber. The constraint Saber has motion capture markers to track its position and a FT sensor to measure forces. 3. An alternative tool for edge and plane constraint interaction (Flat Tool).**

In a typical demonstration, a demonstrator uses the constraint saber to interact with the available constraints on the testbed. Such demonstrations contain multiple constraint interactions of different types where transitions can occur between constraint types or between a constraint type and free space.

Experimental evaluations are performed with two demonstrators. For each evaluation, (an approximately) equal number of trials are performed by each demonstrator. An expert catalogues the number of true and false positive constraint detections: a true positive occurs when a constraint is detected correctly, i.e. the location and type of constraint are correctly identified, a false positive occurs when a constraint is detected in a location where that particular constraint does not exist. Examples include a plane constraint detected where a line constraint should be or a plane constraint detected in a location where no constraint exists. In our experiments we use four metrics:

- TP: Number of correctly detected constraints - *True Positives*
- FP: Number of incorrectly detected constraints - *False Positives*
- PPV: = TP/(TP + FP) - *Precision, Positive Predictive Value*
- Recall: = TP/(Total number of actual constraints)

Precision is an efficient metric to evaluate this kind of recognition as it describes how well the method is robust against detecting false positives. Recall provides a measure of how well constraints are identified. As we will see, detection is not as difficult as preventing a false detection.

A. Single Constraint Type Recognition

The algorithm's performance is evaluated in situations where only one type of constraint appears in a demonstration. In such a demonstration, the demonstrator uses the constraint saber to interact with a particular type of constraint such as only line constraints. This experiment contains 39, 39 and 36 trials of 3 plane, 3 line and 3 arc constraints respectively.

A summary of the results are given in figure (table) 5, row 1-3. The precision of the identified plane, line and arc constraints are 0.96, 0.98 and 0.83 respectively. The recall numbers for the same are 0.99, 1 and 0.95. The arc is the most difficult to recognize, particularly in cases where the ratio of the arc-length to radius is small or when the radial forces applied are low.

B. Multiple Constraint Type Recognition

Our approach is evaluated in situations where multiple constraint types occur in a single demonstration. See figure 1. The demonstrators interact with constraints on the test bed and are directed to perform 2 planar, 2 arc and 2 line constraints

in a demonstration. Our method's performance is evaluated by recording 39 trials containing 6 constraints each from two different demonstrators(20 and 19 each).

A summary of the results are given in figure (table) 5, row 4. The precision was 0.93 while the recall was 0.90 for this experiment. In these experiments the majority of false positives were due to arc constraints falsely detected as plane constraints.

C. Position only approach evaluation

From the experimental results described above, it is clear that the performance of our method is reasonably good. However it is important to assess its performance relative to a method that uses only position (kinematic) information. To make a fair comparison, our approach is adapted to function with only position measurements making minimal changes.

The position only approach uses the gaussian weighted windowed SVD to estimate the local plane normal vector. (Instead of the normal force vector - which is not available.) The most significant difference is due to the absence of the reaction force measurement and its use in identifying active constraints. In the position-only approach, this step is not possible and thus all samples must be considered for constraint detection. Other than these differences, the other elements of the position only approach are identical.

Using the position-only approach we perform demonstrations described in section VI-B. While the recall value (0.87) is similar to the force and position method, the precision is low (0.59). The position only method correctly labels 201 constraints out of 231 constraints, while also detecting an additional 141 false positives. This is captured by the low precision (PPV: 0.59) which is significantly less than the forces and position approach (PPV: 0.93). A summary of the results is provided in figure (table)5, row 6.

The position only approach detects plane constraints in locations where constraints do not exist. Figure 7 shows an example of incorrect detection of a planar constraint in free space by the position only approach.

In conclusion, the large false positive rate is contributed by the lack of sufficient information i.e., the absence of forces. While the position only method correctly identifies motions contained in fictitious planes in free space, these planes are not constraints as a constraint physically requires force interaction.

D. Alternative tooling

The experiments described above use a tool with a single point of contact. However, our approach can be applied to other contact geometries. To test this, we developed a flat tool that attaches to the constraint saber (Figure 6). Our methods provide constraints relative to a specific point of reference on the tool, and require knowledge of the geometry of the tool in order to provide precise positions for the inferred constraints. To test the performance of the methods with this tool an experiment was conducted with two different people performing 12 and 13 trials of 2 planar and 2 line constraints each. Accuracy for these tests was very high (Precision: 0.97, Recall: 1.0).

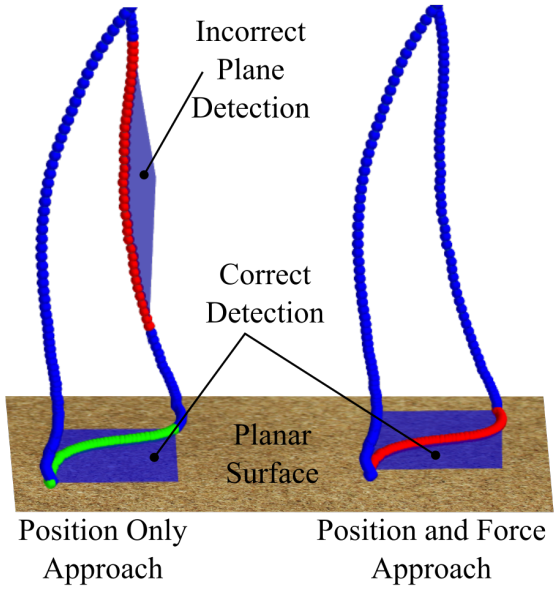

Fig. 7. **The position only approach detects planes in free space while the force and position approach does not.**

VII. Conclusion and Future Work

Our constraint recognition method is effective at identifying constraints. Reaction force information significantly improves constraint recognition when position information is insufficient to disambiguate between constraint types. Reaction force information provides valuable information about the existence of a constraint.

The approach outlined in this article uses local information over windows and tries to cluster this information globally over the entire motion. This is an effective way of processing demonstrations. It also shows a way of detecting parametric models over demonstrations like lines, planes and arcs.

The methods in this paper consider only force information, not moments. This creates a limitation in the kinds of constraints that can be recognized. Moments are important for some constraints, such as turning a door knob or pushing with a tool. Therefore, a more general solution to the constraint recognition problem would require accounting for moments. While the general approach of this paper may apply, a different mathematical framework would be required to extend our methods to fully consider moments. We consider this an interesting avenue for future work.

VIII. Acknowledgement

We thank Jordan Black for his assistance with the experimental evaluations.

APPENDIX A
ALGORITHMS

Algorithm 1 Proximity Weighted Selection Filter

Consider m discrete sequences of length n that take values from an un-ordered set $labels$ with l elements.

sig_mat is an $n \times m$ matrix of input sequences
$prob\_mat := \mathbf{0}_{n \times l}$
consolidated_signal := empty length n sequence
for $k \in 1,2...l$ **do**
  for $i \in 1,2...n$ **do**
    for $j \in 1,2...m$ **do**
      if $labels[k] == sig\_mat[i,j]$ **then**
        $prob\_mat[i,k] := prob\_mat[i,k] + 1$
  $prob\_mat[:,k] := gaussian\_filter(prob\_mat[:,k])$
for $i \in 1,2...n$ **do**
  $label\_id := max\_index(prob\_mat[i,:])$
  $consolidated\_signal[i] := labels[label\_id]$
consolidated_signal is the required filtered signal

Algorithm 2 Tiered DBSCAN

DBSCAN parameters: *min-points*, *eps* and distance *metric*.

clusters := [*data*] ▷ Single cluster containing all data
for each *feature* in *features*:
  current_cluster_set := [ ]
  for each *cluster* in *clusters*:
    sub_clusters := DBSCAN(*cluster*, *feature.metric*, *feature.eps*, *feature.min_samples*)
    append *sub_clusters* to *current_cluster_set*
  clusters := *current_cluster_set*
clusters is the list of required clusters.

APPENDIX B
WEIGHTED SVD FOR PLANE AND LINE ESTIMATES

Lines or planes can be fit using the weighted SVD technique elaborated by Shakarji's et al. [14] . Consider a set of N points $p_1, p_2 ... p_N$ where each $p_i = (x_i, y_i, z_i) \in \mathbb{R}^3$, with corresponding positive weights $w_1, w_2 ... w_i ... w_N \in \mathbb{R}$. A point on the plane (line) is given by the weighted centroid

$$\overline{p} = \frac{\sum_{i=1}^{N} w_i p_i}{\sum_{i=1}^{N} w_i}$$

and the normal (tangent) vector to the plane (line) is the right singular vector that corresponds to the smallest (largest) singular value in the singular value decomposition of the $N \times 3$ matrix given by:

$$\mathbf{M} = \begin{bmatrix} \sqrt{w_1}(\mathbf{p_1^T} - \overline{\mathbf{p}}^\mathbf{T}) \\ \sqrt{w_2}(\mathbf{p_2^T} - \overline{\mathbf{p}}^\mathbf{T}) \\ \vdots \\ \sqrt{w_N}(\mathbf{p_N^T} - \overline{\mathbf{p}}^\mathbf{T}) \end{bmatrix}$$

APPENDIX C
LEAST SQUARES REGRESSION FOR ARC ESTIMATES

Consider a set of N points as in the previous section. The procedure to fit an arc is to first find the best fit plane containing the points then to use nonlinear least squares to fit the arc in that plane and finally transform back. The best fit plane is obtained using SVD and a corresponding inverse rotation matrix $R^T, R \in SO3$ is found. The points are transformed into the plane by:

$$\tilde{\mathbf{p}} = R^T (\mathbf{p} - \overline{\mathbf{p}})$$

Consider $\tilde{\mathbf{p}}$ as a 2×1 matrix ignoring the z component. The radius r and center $C = R\tilde{C}$ are given by:

$$\arg\min_{\tilde{C},r} \sum_{i=1}^{N} w_i \left(\left\| \left(\tilde{\mathbf{p}}_\mathbf{i} - \tilde{\mathbf{C}} \right) \right\|^2 - r^2 \right)^2$$

REFERENCES

[1] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, 2009.

[2] V. Ortenzi, H.-c. Lin, M. Azad, J. A. Kuo, and M. Mistry, "Kinematics-based estimation of contact constraints using only proprioception," *2016 IEEE-RAS International Conference on Humanoid Robots (Humanoids 2016)*, pp. 1304–1311, 2016.

[3] S. Levine, N. Wagener, and P. Abbeel, "Learning Contact-Rich Manipulation Skills with Guided Policy Search," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015-June, no. June, pp. 156–163, 2015. [Online]. Available: https://arxiv.org/pdf/1501.05611.pdf

[4] C. Pérez-D'Arpino and J. A. Shah, "C-learn: Learning geometric constraints from demonstrations for multi-step manipulation in shared autonomy," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 4058–4065.

[5] S. Niekum, S. Osentoski, C. G. Atkeson, and A. G. Barto, "Online Bayesian changepoint detection for articulated motion models," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015-June, no. June, pp. 1468–1475, 2015.

[6] H. C. Lin, M. Howard, and S. Vijayakumar, "Learning null space projections," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015-June, no. June, pp. 2613–2619, 2015.

[7] W. Meeussen, J. Rutgeerts, K. Gadeyne, H. Bruyninckx, and J. De Schutter, "Contact state segmentation using particle filters for programming by human demonstration in compliant motion tasks," *Springer Tracts in Advanced Robotics*, vol. 39, pp. 3–12, 2008.

[8] P. Mukhopadhyay and B. B. Chaudhuri, "A survey of Hough Transform," *Pattern Recognition*, vol. 48, no. 3, pp. 993–1010, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.patcog.2014.08.027

[9] C. Lanczos, *The variational principles of mechanics*. Courier Corporation, 2012.

[10] E. J. Haug, *Computer aided kinematics and dynamics of mechanical systems*. Allyn and Bacon Boston, 1989, vol. 1.

[11] S. Angenent, *Third Semester Calculus*, 091st ed., 2015.

[12] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pp. 226–231, 1996. [Online]. Available: https://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[14] C. M. Shakarji and V. Srinivasan, "Theory and Algorithms for Weighted Total Least-Squares Fitting of Lines, Planes, and Parallel Planes to Support Tolerancing Standards," *Journal of Computing and Information Science in Engineering*, vol. 13, no. 3, pp. 31 008–31 011, aug 2013. [Online]. Available: http://dx.doi.org/10.1115/1.4024854