ABSTRACT

Title of Dissertation: A FRAMEWORK FOR DEXTEROUS MANIPULATION THROUGH TACTILE PERCEPTION

Kanishka Ganguly
Doctor of Philosophy, 2022

Dissertation Directed by: Prof. Yiannis Aloimonos and Dr. Cornelia Fermüller
Department of Computer Science

A long-anticipated, yet hitherto unfilled goal in Robotics research has been to have robotic agents seamlessly integrating with humans in their natural environments, and performing useful tasks alongside humans. While tremendous progress has been made in allowing robots to perceive visually, and understand and reason about the scene, the act of manipulating said environment still remains a challenging and incomplete task.

For robotic agents to have capabilities where they can perform useful tasks in environments that are not specifically designed for their operation, it is crucial to have dexterous manipulation capabilities guided by some form of tactile perception. While visual perception provides a large-scale understanding of the environment, tactile perception allows fine-grained understanding of objects and textures. For truly useful robotic agents, a tightly coupled system comprising both visual and tactile perception is a necessity.

Tactile sensing hardware can be classified on a spectrum, organized by form-factor on one

end to sensing accuracy and robustness on the other. Most off-the-shelf sensors available today trade off one of these features for the other. The tactile sensor used in this research, the BioTac SP, has been selected for its anthropomorphic qualities, such as its shape and sensing mechanism while compromising on quality of sensory outputs. This sensor provides a sensing surface, and returns 24 tactile points of data at each timestamp, along with pressure values.

We first present a novel method for contact and motion estimation through visual perception, where we perform non-rigid registration of a human performing actions and compute dense motion estimation trajectories. This is used to compute topological scene changes, and is refined to get object and contact segmentation. We then ground these contact points and motion trajectories to an intermediate action-graph, which can then executed by a robot agent.

Secondly, we introduce the concept of *computational tactile flow*, which is inspired by fMRI studies on humans where it was discovered that the same parts of the brain that react to optical motion stimulus also react to tactile stimulus. We mathematically model the BioTac SP sensor, and interpolate surfaces in two- and three dimensions, on which we compute tactile flow fields. We demonstrate the flow fields on various surfaces, and suggest various useful applications of tactile flow.

We next apply tactile feedback to a novel controller, that is able to grasp objects without any prior knowledge about the shape, material, or weight of the objects. We apply tactile flow to compute slippage during grasp, and adjust the finger forces to maintain stable grasp during motion. We demonstrate success on transparent and soft, deformable objects, alongside other regularly shaped samples.

Lastly, we take a different approach to processing tactile data, where we compute *tactile events* taking inspiration from neuromorphic computing literature. We compute spatio-temporal

gradients on the raw tactile data, to generate *event surfaces*, which are more robust and reduces sensor noise. This intermediate surface is then used to track contact regions over the BioTac SP sensor skin, and allows us to detect slippage, track spatial edge contours, and magnitude of applied forces.

A FRAMEWORK FOR DEXTEROUS MANIPULATION THROUGH TACTILE PERCEPTION

by

Kanishka Ganguly

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2022

Advisory Committee:
Prof. Yiannis Aloimonos, Chair/Advisor
Dr. Cornelia Fermüller, Co-Advisor
Prof. Miao Yu, Dean's Representative
Prof. Dinesh Manocha
Prof. Nitin J. Sanket

Acknowledgments

First and foremost, I would like to express my gratitude to my academic advisors, Prof. Yiannis Aloimonos and Dr. Cornelia Fermüller for their guidance and generous support throughout my time as a student. They have been a constant source of knowledge and inspiration on my journey through the field of robotics and perception, and have helped me grow into my own as a researcher and a student. This dissertation would not have been possible without them.

I would also like to extend my thanks to all members of my doctoral dissertation committee, Prof. Dinesh Manocha, Prof. Nitin J. Sanket and Prof. Miao Yu, who have taken the time to provide constructive feedback on my work.

I also deeply grateful to everyone at the Perception and Robotics Group, for having supported me throughout this journey. To my old friends, Aleksandrs Ecins, Konstantinos Zampogiannis, Moschoula Pternea, Gregory Kramida, Anton Mitrokhin, Behzad Sadrfaridpour, Snehesh Shreshta, Nitin Sanket, Chahat Deep Singh, Chethan Parameshwara, Peter Sutor, Michael Maynord, and Chinmaya Devaraj, I will always remember the good times we shared on *our wing* of A.V. Williams and all the discussions and laughs that ensued. May the inside jokes live on beyond our shared geographic locations.

To my newest compatriots, Levi Burner, Pavan Mantripragada, Angelos Mavrogiannis, and Eadom Dessalene, I am glad to have shared this time with you all, and I wish you all the very best of luck in your careers ahead. I'm sure our paths shall cross again in the near future.

Lastly, I want to thank my family for being there through thick and thin, and for providing the moral support I needed to get through this endeavour. This achievement belongs to them just as much as me. To my parents Roshmi and Krishnendu, to whom I owe this and every other success, thank you for the inspiration and having faith in me even when I doubted myself. To my brother Ratul, I am glad I won't have to raise my hand when they ask for a doctor on an airplane, that responsibility lies with you. To Shruti, thank you for the last decade, for standing steadfastly beside me every step of the way.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

2D	Two Dimensional
3D	Three Dimensional
CNN	Convolutional Neural Network
DMP	Dynamic Motion Primitives
DOF	Degrees of Freedom
DVS	Dynamic Vision Sensor
FBTS	Fluid-Based Tactile Sensor
FEM	Finite Element Modelling
fMRI	Functional Magnetic Resonance Imaging
FSR	Force Sensitive Resistor
ICP	Iterative Closest Points
IMU	Inertial Measurement Unit
MLP	Multi-Layer Perceptron
MRF	Markov Random Field
RANSAC	Random Sample Consensus
RGBD	Red-Green-Blue (image) and Depth
ROS	Robot Operating System
SDF	Signed Distance Field
SE(3)	Special Euclidean Group in 3 Dimensions
SNR	Signal-to-Noise Ratio

Chapter 1: Extracting Contact and Motion from Manipulation Videos

When we physically interact with our environment using our hands, we touch objects and force them to move: contact and motion are defining properties of manipulation. In this work, we present an active, bottom-up method for the detection of actor-object contacts and the extraction of moved objects and their motions in RGBD videos of manipulation actions. At the core of our approach lies non-rigid registration: we continuously warp a point cloud model of the observed scene to the current video frame, generating a set of dense 3D point trajectories. Under loose assumptions, we employ simple point cloud segmentation techniques to extract the actor and subsequently detect actor-environment contacts based on the estimated trajectories. For each such interaction, using the detected contact as an attention mechanism, we obtain an initial motion segment for the manipulated object by clustering trajectories in the contact area vicinity and then we jointly refine the object segment and estimate its 6DOF pose in all observed frames. Because of its generality and the fundamental, yet highly informative, nature of its outputs, our approach is applicable to a wide range of perception and planning tasks. We evaluate our method on a number of input sequences and present a comprehensive robot imitation learning example, in which we demonstrate the crucial role of our outputs in developing action representations/plans from observation.

1.1 Introduction

A manipulation action, by its very definition, involves the handling of objects by an intelligent agent. Every such interaction requires physical contact between the actor and some object, followed by the exertion of forces on the manipulated object, which typically induce motion. When we open a door, pick up a coffee mug, or pull a chair, we invariably touch an object and cause it (or parts of it) to move. This obvious observation demonstrates that *contact* and *motion* are two fundamental aspects of manipulation.

Contact and motion information alone are often sufficient to describe manipulations in a wide range of applications, as they naturally encode crucial information regarding the performed action. Contact encodes *where* the affected object was touched/grasped, as well as *when* and for how long the interaction took place. Motion conveys *what* part of the environment (i.e. which object or object part) was manipulated and *how* it moved.

The ability to automatically extract contact and object motion information from video either directly solves or can significantly facilitate a number of common perception tasks. For example, in the context of manipulation actions, knowledge of the spatio-temporal extent of an actor-object contact automatically provides action *detection/segmentation* in the time domain, as well as *localization* of the detected action in the observed space [1, 2]. At the same time, motion information bridges the gap between the observation of an action and its semantic grounding. Knowing what part of the environment was moved effectively acts as an attention mechanism for the manipulated *object recognition* [3, 4], while the extracted motion profile provides invaluable cues for *action recognition*, in both "traditional" [1, 2, 5–20] and deep learning [21] frameworks.

Robot imitation learning is rapidly gaining attention. The use of robots in less controlled

workspaces and even domestic environments necessitates the development of easily applicable methods for robot "programming": autonomous robots for manipulation tasks must efficiently *learn* how to manipulate. Exploiting contact and motion information can largely automate robot replication of a wide class of actions. As we will discuss later, the detected contact area can effectively bootstrap the grasping stage by guiding primitive fitting and grasp planning, while the extracted object and its motion capture the trajectory to be replicated as well as any applicable kinematic/collision constraints. Thus, the components introduced in this work are essential for building complex, hierarchical models of action (e.g., behavior trees, activity graphs) as they appear in the recent literature [22–28].

In this work, we present an unsupervised, bottom-up method for estimating from RGBD video the contacts and object motions in manipulation tasks. Our approach is fully 3D and relies on dense motion estimation: we start by capturing a point cloud model of the observed scene and continuously warp/update it throughout the duration of the video. Building upon our estimated dense 3D point trajectories, we use simple concepts and common sense rules to segment the actor and detect actor-environment contact locations and time intervals. Subsequently, we exploit the detected contact to guide the motion segmentation of the manipulated object and, finally, estimate its 6DOF pose in all observed video frames. Our intermediate and final results are summarized in Table 1.1.

It is worth noting that we do not treat contact detection and object motion segmentation/estimation independently: we use the detected contact as an *attention mechanism* to guide the extraction of the manipulated object and its motion. This *active* approach provides an elegant and effective solution to our motion segmentation task. A passive approach to our problem would typically segment the whole observed scene into an *unknown* (i.e. to be estimated) num-

ber of motion clusters. By exploiting contact, we avoid having to solve a much larger and less constrained problem, while gaining significant improvements in terms of both computational efficiency and segmentation/estimation accuracy.

The generality of our framework, combined with the highly informative nature of our outputs, renders our approach applicable to a wide spectrum of perception and planning tasks. In Section 1.3, we provide a detailed technical description of our method, while in Section 1.4, we demonstrate our intermediate results and final outputs for a number of input sequences. In Section 1.5, we present a comprehensive example of how our outputs were successfully used to facilitate a robot imitation learning task.

1.2 Related Work

We focus our literature review on recent works in four areas that are most relevant to the our twofold problem, and the major processes/components upon which we build. We deliberately do not review works from the action recognition literature; while our approach may very appropriately become a component of a higher-level reasoning solution, the scope of this work is the extraction of contacts, moving objects, and their motions.

Scene flow. Scene flow refers to the dense 3D motion field of an observed scene with respect to a camera; its 2D projection onto the image plane of the camera is the optical flow. Scene flow, analogously to optical flow, is typically computed from multi-view frame pairs [29]. There have been a number of successful recent works on scene flow estimation from RGBD frame pairs, following both variational [30–34] and deep learning [35] frameworks. While being of great relevance in a number of motion reasoning tasks, plain scene flow cannot be directly

integrated into our pipeline, which requires *model-to-frame* motion estimation: the scene flow motion field has a 2D support (i.e. the image plane), effectively warping the 2.5D geometry of an RGBD frame, while we need to appropriately warp a *full* 3D point cloud model.

Non-rigid registration. The non-rigid alignment of 3D point sets can be viewed as a generalization of scene flow, in the sense that the estimated motion field is supported by a 3D point cloud: the goal is to estimate point-wise transformations (usually rigid) that best align the point set to the target geometry under certain global prior constraints (e.g., 'as-rigid-as-possible' [36]). The warp field estimation is performed either by iterating between correspondence estimation and motion optimization [37–40], or in a correspondence-free fashion, by aligning volumetric SDFs (Signed Distance Fields) [41]. For this work, and due to lack of publicly available solutions, we have implemented a non-rigid registration algorithm very similar to [39] and [40] (Section 1.3.2).

Contact detection. A CNN-based method for grasp recognition is introduced in [42]. A 2D approach for detecting "touch" interactions between a caregiver and an infant is presented in [43]. To the best of our knowledge, there is no prior work on explicitly determining the spatiotemporal extent of human-environment contact.

Motion segmentation. A very large volume of works on motion segmentation have casted the problem as subspace clustering of 2D point trajectories, assuming an affine camera model [44–54]. In [55], an active approach for the segmentation and kinematic modeling of articulated objects is proposed, which relies on the robot manipulation capabilities to induce object motion. In [56], object segmentation is performed from two RGBD frames, one before and one after the manipulation of the object, by rigidly aligning and 'differencing' the two views and robustly estimating rigid motion between the 'difference' regions. The same method is used in [30], where scene flow is used to obtain motion proposals, followed by an MRF inference step. In [57], joint

tracking and reconstruction of multiple rigidly moving objects is achieved by combining two segmentation/grouping strategies with multiple surfel fusion [58] instances. A naive integration of a generic motion segmentation algorithm for the extraction of the manipulated object into our pipeline would be suboptimal in multiple ways. For instance, given the fact that there may exist an unknown number of other object motions that are irrelevant to the manipulation, we would be solving an unnecessarily hard problem. For the same reason, we would have little control over the segmentation granularity, which could cause the manipulated object to be over/under-segmented. Instead, we leverage the detected contact and bootstrap our segmentation by an informed trajectory clustering approach that is similar to [59].

1.3 Our Approach

1.3.1 Overview

We present an automated system that, given a video of a human performing a manipulation task as input, *detects* and *tracks* the parts of the environment that participate in the manipulation. More specifically, our system is able to visually detect physical contact between the actor and their environment, and, using contact as an attention mechanism, eventually segment the manipulated object and estimate its 6DOF pose in every observed video frame. Our pipeline, as well as the interactions of the involved processes, are sketched in Fig. 1.1 and followed by a more detailed description. An in-depth discussion of our core modules is provided in the following subsections.

The input to our system is an RGBD frame sequence, captured by a commodity depth sensor, of a human actor performing a task that involves the manipulation of objects in their

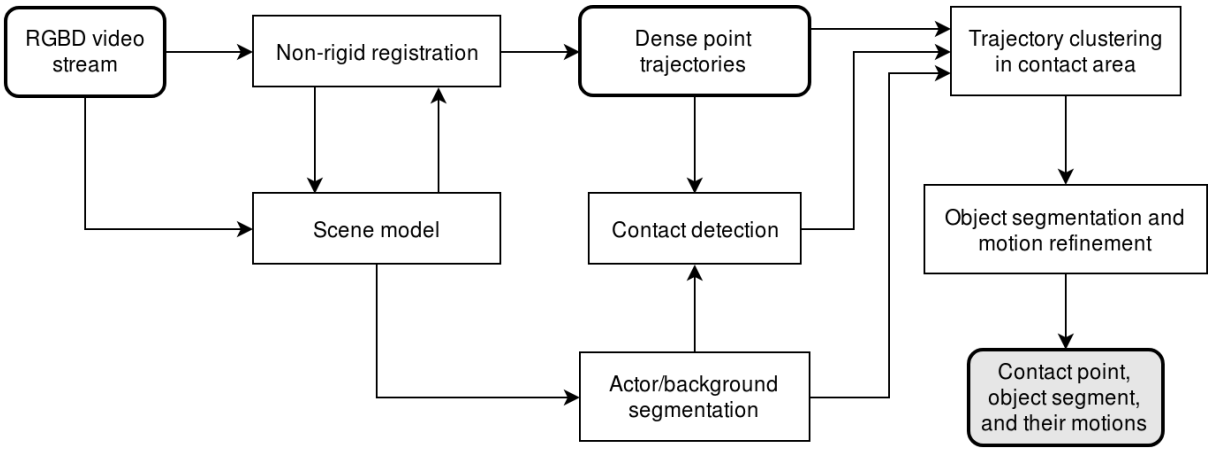


Figure 1.1: A high-level overview of our modules and their connections in the proposed pipeline.

environment. We assume that the input depth images are registered to and in sync with their color counterparts. Using estimates of the color camera intrinsics (e.g., from the manufacturer provided specifications), all input RGBD frames are back-projected to 3D point clouds (colored, with estimated surface normals), on which all subsequent processing is performed.

At the core of our method lies non-rigid point cloud registration, described in detail in Section 1.3.2. An initial point cloud model of the observed scene is built from the first observed frame and is then consecutively transformed to the current observation based on the estimated *model-to-frame* warp field at every time instance. This process generates a dense set of point trajectories, each associated with a point in the initial model. In order to keep the presentation clean, we opted to obtain the scene model from the first frame and keep it fixed in terms of its point set. Non-rigid reconstruction techniques for updating the model over time [39, 40] can be easily integrated to our pipeline if required.

To perform actor/background segmentation, we follow the semi-automatic approach described in Section 1.3.3. The obtained binary labeling is propagated to the whole temporal extent of the observed action via our estimated dense point trajectories, and enables us to easily detect

human-environment *contacts* as described in Section 1.3.4.

Given the dense scene point trajectories, the actor/background labels, and the (hand) contact interaction locations and time intervals, our final goal is, for each detected interaction, to *segment* the manipulated object and re-estimate its *motion* for every time instance, assuming it is rigid (i.e. fully defined by a 6DOF pose). Our contact-guided motion segmentation approach for this task is described in Section 1.3.5.

In Table 1.1, we summarize our proposed system's expected inputs, final outputs, and some useful generated intermediate results.

Table 1.1: List of the inputs, intermediate results, and final outputs of our proposed system.

Input	**Intermediate results**	**Final outputs**
RGBD video of manipulation	• **Dense 3D point trajectories** for the whole sequence duration • **Actor/background labels** for all model points at all times	• **3D trajectories** of detected actor-environment **contact points** • Manipulated **object segments** and their **6DOF poses** for every time point

1.3.2 Non-rigid registration

As described in the previous subsection, whenever a new RGBD frame (point cloud) becomes available, our scene model is non-rigidly warped from its previous state (that corresponds to the previous frame) to the new (current) observation. Since parts of the scene model may be invisible in the current state (e.g., because of self-occlusion), we cannot directly apply a traditional scene flow algorithm, as that would only provide us with motion estimates for (some of) the currently visible points. Instead, we adopt a more general approach, by implementing a non-rigid Iterative Closest Point (ICP) algorithm, similar to [38–40].

As is the case with rigid ICP [60], our algorithm iterates between a correspondence search

step and a warp field optimization step for the given correspondences. Our correspondence search typically amounts to finding the nearest neighbors of each point in the current frame to the model point cloud in its previous state. Correspondences that exhibit large point distance, normal angle, or color difference are discarded. Nearest neighbor searches are done efficiently by parallel kd-tree queries.

In the following, we will focus on the warp field optimization step of our scheme. It has been found that modeling the warp field using locally affine [38] or locally rigid [39] transformations provides better motion estimation results than adopting a simple translational local model, due to better regularization. In our implementation, for each point of the scene model in its previous state, we compute a full 6DOF rigid transformation that best aligns it to the current frame.

Let $X = \{x_i\}$ be the set of scene model points in the previous state that need to be registered to the point set $Y = \{y_i\}$ of the current frame, whose surface normals we denote by $Y^n = \{n_i\}$. Let $S = \{s_i\} \subseteq \{1, \ldots, |X|\}$ and $D = \{d_i\} \subseteq \{1, \ldots, |Y|\}$ be the index sets of corresponding points in X and Y respectively, such that (x_{s_i}, y_{d_i}) is a pair of corresponding points. Let $T = \{T_i\}$ be the unknown warp field of rigid transformations, such that $T_i \in SE(3)$ and $|T| = |X|$, and $T_i(x_i)$ denote the application of T_i to model point x_i. Local transformations are parameterized by 3 Euler angles (α, β, γ) for their rotational part and 3 offsets (t^x, t^y, t^z) for their translational part, and are represented as 6D vectors $T_i = \begin{bmatrix} \alpha_i & \beta_i & \gamma_i & t_i^x & t_i^y & t_i^z \end{bmatrix}^T$.

Our goal at this stage is to estimate a warp field T, of $6|X|$ unknown parameters, that maps model points in S as closely as possible to frame models in D. We formulate this property as the minimization of a weighted combination of sums of point-to-plane and point-to-point squared

distances between corresponding pairs:

$$E_{data}(T) = \sum_{i=1}^{|S|} \left(n_{d_i}^T \left(T_{s_i}(x_{s_i}) - y_{d_i} \right) \right)^2 + w_{point} \sum_{i=1}^{|S|} \left\| T_{s_i}(x_{s_i}) - y_{d_i} \right\|^2 . \tag{1.1}$$

Pure point-to-plane metric optimization generally converges faster and to better solutions than pure point-to-point [61] and is the standard trend in the state of the art for both rigid [58, 62] and non-rigid [39, 40] registration. However, we have found that integrating a point-to-point term (second term in Eqn. 1.1) with a small weight (e.g., with $w_{point} \approx 0.1$) to the registration cost improves motion estimation on surfaces that lack geometric texture.

The set of estimated correspondences is only expected to cover a subset of X and Y, as not all model points are expected to be visible in the current frame, and the latter may suffer from missing data. Furthermore, even for model points with existing data terms (correspondences) in Eqn. 1.1, analogously to the aperture problem in optical flow estimation, the estimation of point-wise transformation parameters locally is under-constrained. These reasons render the minimization of the cost function in Eqn. 1.1 ill-posed. To overcome this, we introduce a "stiffness" regularization term that imposes an as-rigid-as-possible prior [36] by directly penalizing differences between transformation parameters of neighboring model points in a way similar to [38]. We fix a neighborhood graph on X, based on point locations, and use $\mathcal{N}(i)$ to denote the indices of the neighbors of point x_i to formulate our stiffness term as:

$$E_{stiff}(T) = \sum_{i=1}^{|X|} \sum_{j \in \mathcal{N}(i)} w_{ij} \left\| T_i - T_j \right\|^2 , \tag{1.2}$$

where $w_{ij} = \exp\left(- \left\| x_i - x_j \right\|^2 / (2\sigma_{reg}^2) \right)$, σ_{reg} controls the radial extent of the regularization

neighborhoods, and '$-$' denotes regular matrix subtraction for the 6D vector representations of the local transformations. Our complete registration cost function is a weighted combination of costs Eqns. 1.1 and 1.2:

$$E(T) = E_{data}(T) + w_{stiff} E_{stiff}(T) \tag{1.3}$$

To minimize Eqn. 1.3, we use the same linear approximation of local transformations as in [63]. Specifically, for small Euler angles, the rotation matrix associated with each local transformation can be approximated by a matrix whose entries only consist of linear functions of the unknown angles. Under this assumption, *all* vector differences involved in the summands of our cost function can be written as linear expressions of the unknowns. Therefore, Eqn. 1.3 can be approximated in the form:

$$E(T) \approx \|A \operatorname{vec}(T) - b\|^2, \tag{1.4}$$

where matrix A and vector b encode all data and regularization terms and $\operatorname{vec}(T)$ is our vector of unknowns (the vertical concatenation of all T_i into a single $6|X| \times 1$ column vector). Minimizing Eqn. 1.4 is a standard linear least-squares problem, which we solve using Cholesky factorization. Because of the above approximation, more than one linearization-minimization iterations may be required for convergence. However, since our warp field optimization is anyway part of an iterative procedure, a single iteration is sufficient in practice.

In Fig. 1.2, we show two sample outputs of our algorithm in an RGBD frame pair non-rigid alignment scenario. Our registration module accurately estimates deformations even for complex motions of significant magnitude.

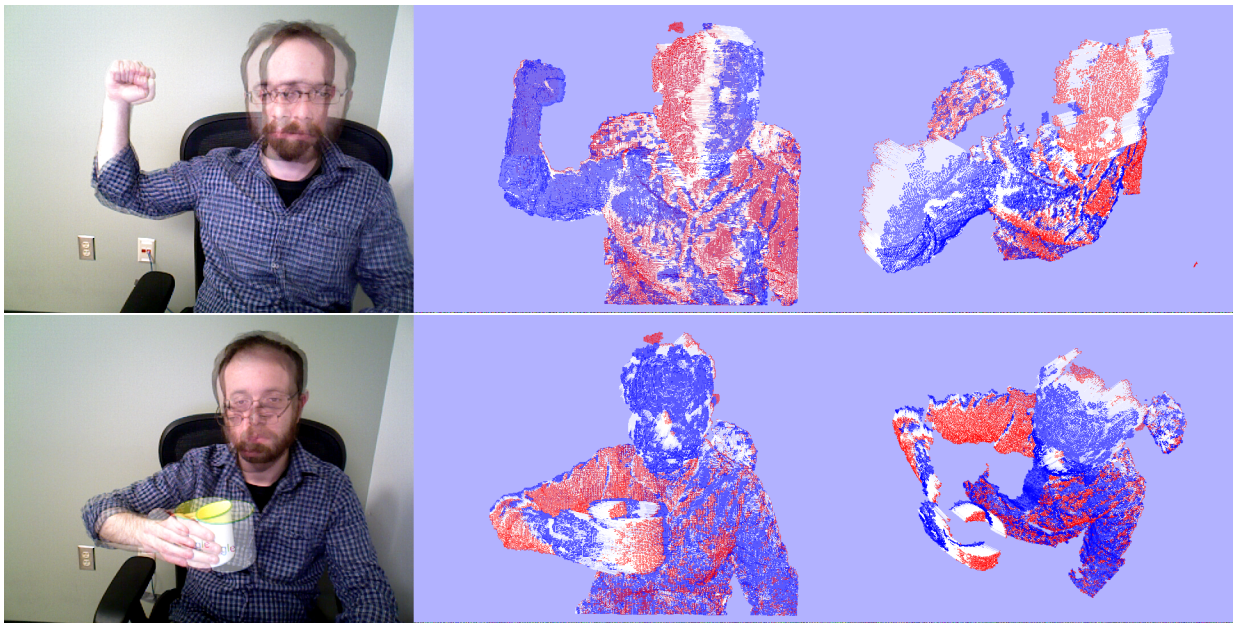


Figure 1.2: Non-rigid registration: displacement vectors are depicted as white lines, aligning the source (red) to the target (blue) geometry.

1.3.3 Human actor segmentation

We follow a semi-automatic approach to perform actor/background segmentation that relies on simple point cloud segmentation techniques.

We construct a proximity graph over the scene model points in the initial state, in which each node is a model point and two nodes are connected if and only if their Euclidean distance falls below a predefined threshold. Assuming that the actor is *initially* not in contact with any other part of the scene (i.e. the minimum distance of an actor point to a background point is at least our predefined distance threshold) and the observed actor points are not too severely disconnected in the initial state, the actor points will be exactly defined by one connected component of this proximity graph. The selection of the correct (actor) component can be automated by filtering all the extracted components based on context-specific criteria (e.g., rough size, shape, location, etc.) or by picking the component whose image projection exhibits maximum overlap

with the output of a 2D human detector [64, 65]. Equivalently, we may begin by selecting a seed point known to belong to the actor and then perform region growing on the model point cloud until our distance threshold is no longer satisfied. Again, the selection of the seed point can be automated by resorting to standard 2D means (e.g., by picking the point with the strongest skin color response [66, 67] within a 2D human detector output [64, 65]).

We believe that the assumptions imposed by our Euclidean clustering based approach for the actor segmentation task are not too restricting, as the main setting we focus on (representing human demonstrations for robot learning) is reasonably controlled in the first place.

We note that, since we opted to keep the scene model point set fixed and track it throughout the observed action, the obtained segmentation automatically becomes available at all time points.

1.3.4 Contact detection

The outputs of the above two processes are a dense set of *point trajectories* and their respective actor/background *labels*. Given this information, it is straightforward to reason about *contact*, simply by examining whether the minimum distance between parts of the two clusters is small enough at any given time. In other words, we can easily infer both *when* the actor comes into/goes out of contact with part of the environment and *where* this interaction is taking place.

Some of the contact interactions detected using this criterion may, of course, be semantically irrelevant to the performed action. Since semantic reasoning is not part of our core framework, these cases have to be handled by a higher lever module. However, under reasonably controlled scenarios, we argue that it is sufficient to simply assume that the detected contacts are established by the actor *hands*, with the goal of manipulating an *object* in their environment.

1.3.5 Manipulated object motion/segmentation

Knowing the dense scene point trajectories, labeled as either actor or background, as well as the contact locations and intervals, our next goal is to infer what part of the environment is being manipulated, or, in other words, which object was moved. We assume that every contact interaction involves the movement of a *single* object, and that the latter undergoes *rigid* motion. In the following, we only focus on the *background* part of the scene around the contact point area, ignoring the human point trajectories. We propose the following two-step approach.

First, we bootstrap our segmentation task by finding a coarse/partial mask of the moving object, using standard unsupervised clustering techniques. Specifically, we cluster the point trajectories that are labeled as background and lie within a fixed radius of the detected contact point at the beginning of the interaction into two groups. We adopt a spectral clustering approach, using the 'random walk' graph Laplacian [68] and a standard k-means last step. Our pairwise trajectory similarities are given by $s_{ij} = \exp\left(-(d_{max} - d_{min})^2/(2\sigma^2)\right)$, where d_{min} and d_{max} are the minimum and maximum Euclidean point distance of trajectories i and j over the duration of the interaction, respectively. This similarity metric enforces similar trajectories to exhibit relatively constant point-wise distances, i.e. promotes clusters that undergo rigid motion. From the two output clusters, one is expected to cover (part of) the object being manipulated. Operating under the assumption that only interaction can cause motion in the scene, we pick the cluster that exhibits the largest average motion over the duration of contact as our object segment candidate.

In the above, we restricted our focus within a region of the contact point, in order to *1)* avoid that our binary classification is influenced by other captured motions in the scene that are not related to the current interaction, and *2)* make the classification itself more computationally

tractable. As long as these requirements are met, the choice of radius is not important.

Subsequently, we obtain a refined, more accurate segment of the moving object by requiring that the latter undergoes a rigid motion that is at every time point consistent with that of the previously found motion cluster. Let B^t denote the background (non-actor) part the scene model point cloud at time t, for $t = 0, \ldots, T$, and $\hat{M}^t \subseteq B^t$ be the initial motion cluster state at the same time instance. For all $t = 1, \ldots, T$, we robustly estimate the rigid motion between point sets $\hat{M}^0$ and $\hat{M}^t$ (i.e. relative to the first frame), using the closed form solution of [69] under a RANSAC scheme, and then find the set of points in *all* of B^t that are consistent with this motion model between B^0 and B^t. If we denote this set of motion inliers by I^t (which is a set of indices of points in B^t), we obtain our final object segment for this interaction as the intersection of inlier indices for all time instances $t = 1, \ldots, T$:

$$I \equiv \bigcap_{t=1}^{T} I^t \tag{1.5}$$

The subset of the background points indexed by I, as well as the per-frame RANSAC motion (pose) estimates of this last step, are the final outputs of our pipeline for the given interaction.

1.4 Experiments

We provide a qualitative evaluation of our method for video inputs recorded in different settings, covering three different scenarios: *1)* a tabletop object manipulation that involves flipping a pitcher, *2)* opening a drawer, and *3)* opening a room door. All videos were captured from a static viewpoint, using a standard RGBD sensor.

For each scenario, we depict (in Fig. 1.3, 1.4, and 1.5, respectively) the scene model

point cloud state at three time snapshots: one right before, one during, and one right after the manipulation. For each time point, we show the corresponding color image and render the tracked point cloud from two viewpoints. The actor segment is colored green, the background is red, and the detected contact area is marked by blue. We also render the point-wise displacements induced by the estimated warp field (from the currently visible state to its next) as white lines (mostly visible in areas that exhibit large motion). The outputs displayed in these figures are in direct correspondence with the processes described in Sections 1.3.2, 1.3.3, and 1.3.4.

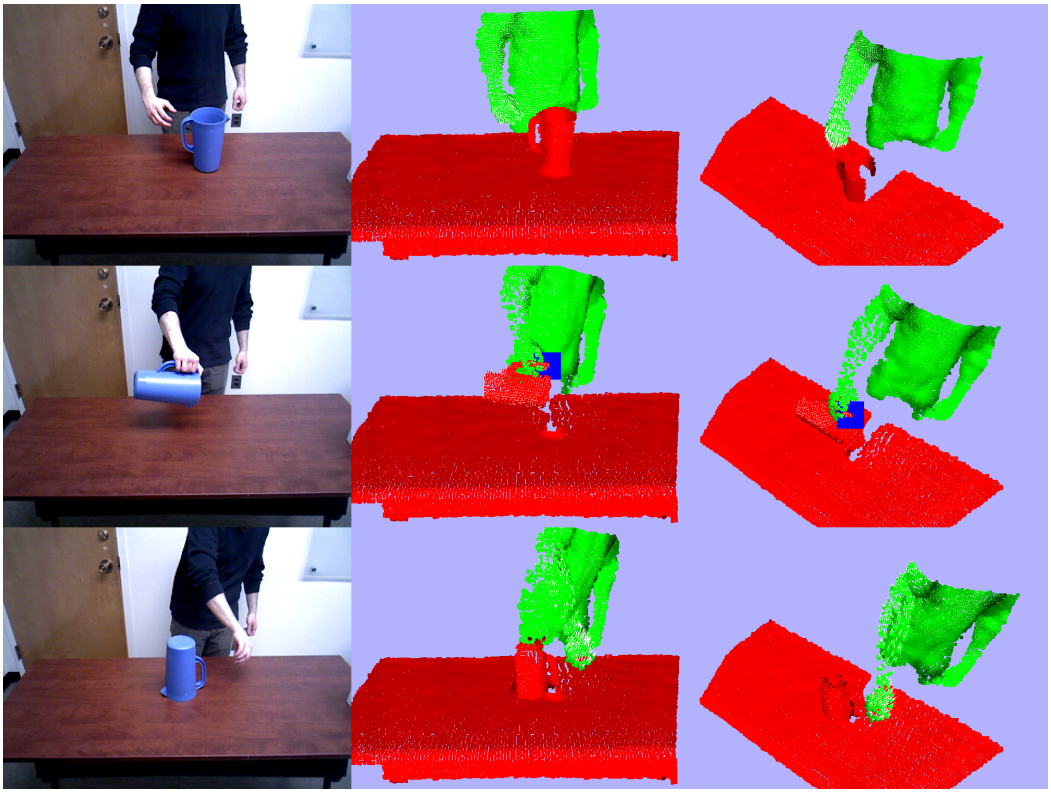


Figure 1.3: Flipping a pitcher: scene tracking, labeling, and contact detection.

Next, we demonstrate our attention-driven motion segmentation and 6DOF pose estimation of the manipulated object. In Fig. 1.6, we render the background part of the scene model at its initial state with the actor removed and show the two steps of our segmentation method described in Section 1.3.5. In the middle column, the blue segment corresponds to the initial

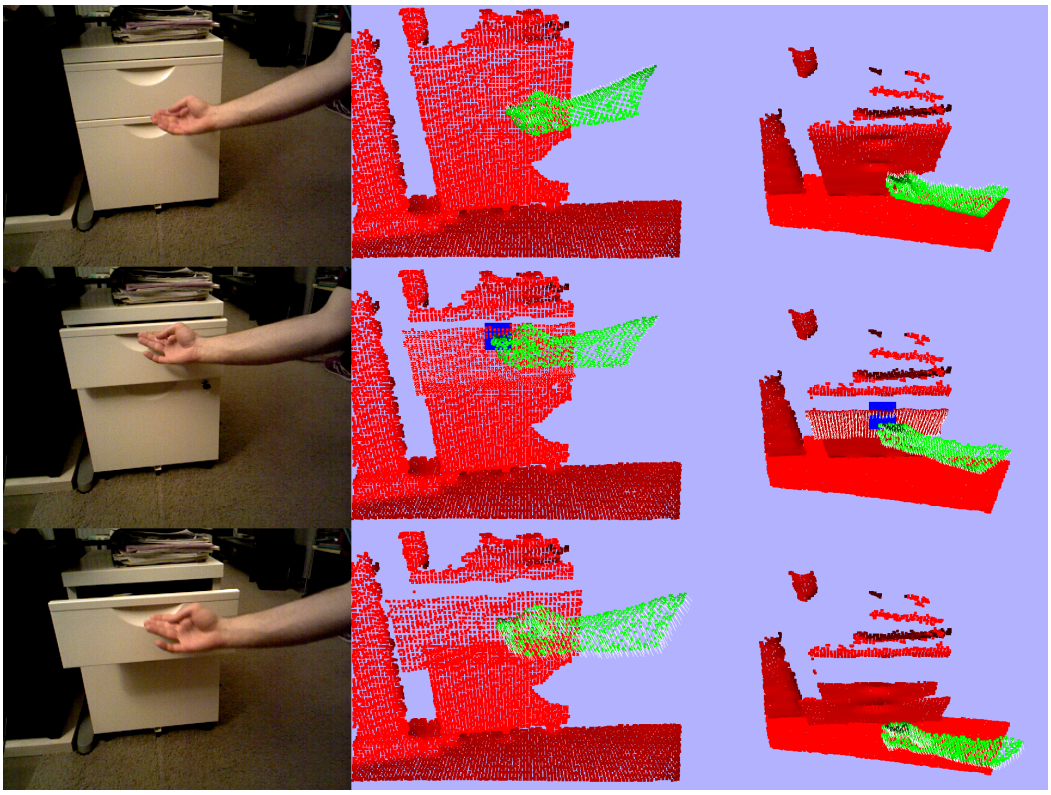


Figure 1.4: Opening a drawer: scene tracking, labeling, and contact detection.

motion segment, obtained by clustering trajectories in the vicinity of the contact point, which was propagated back to the initial model state and is highlighted in yellow. In the left column, we show the refined, final motion segment. We note that, because of our choice of the radius around the contact point in which we focus out attention in the first step, the initial segment in the first two cases is the same as the final one. In Fig. 1.7, we show the estimated rigid motion (6DOF pose) of the segmented object. To more clearly visualize the evolution of object pose over time, we attach a local coordinate frame on the object, at the location of the contact point, whose axes were set as the principal components of the object point set.

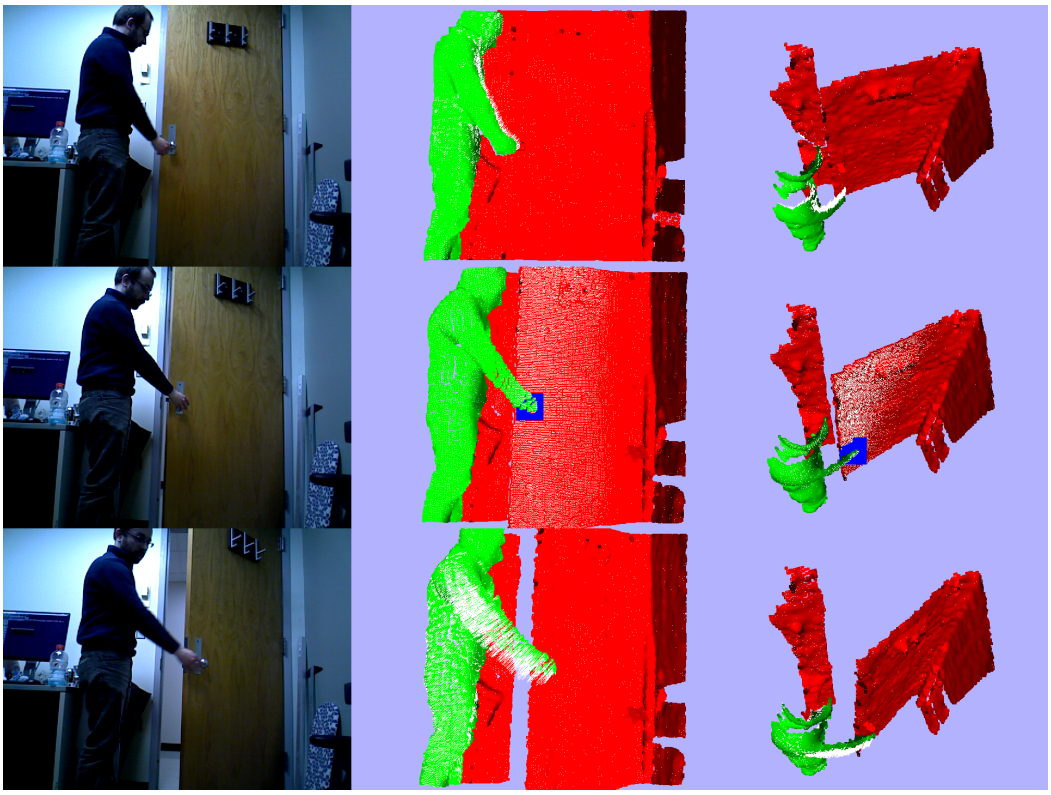
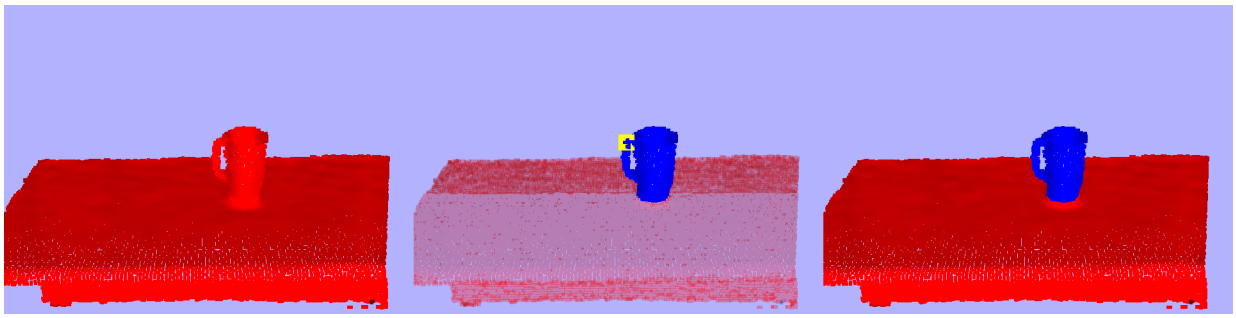


Figure 1.5: Opening a door: scene tracking, labeling, and contact detection.

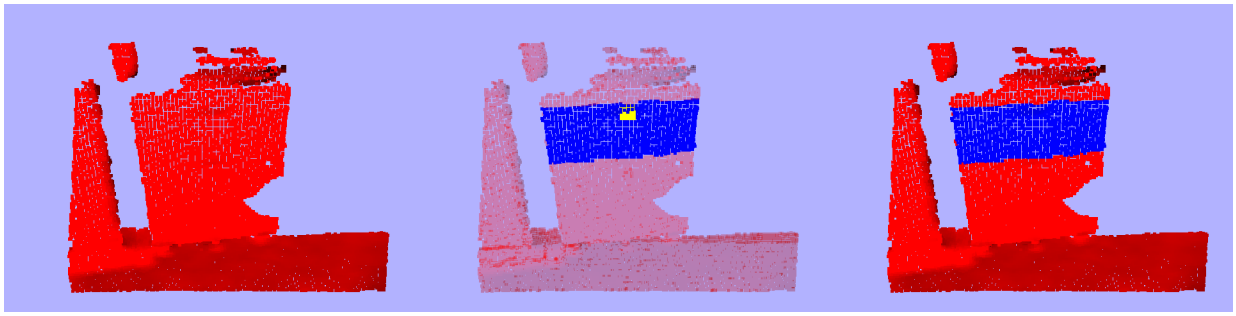
1.5 Application: Replication from Observation by a robot

For any human-environment task to be successful, there is a well-defined process involved, demarcated into phases depending on human-environment contact and consequent motion. This allows us to generate a graph representation for actions, such as that shown in Fig. 1.8, for the task of opening a refrigerator. Given this general representation of tasks, we demonstrate how our algorithm allows grounding of the *grasp* and *release* parts, based on contact detection, and also of the feedback loop for opening the door, based on motion analysis of segmented objects. Such a representation, featuring a tight coupling of planning and perception, is crucial for robots to observe and replicate human actions.

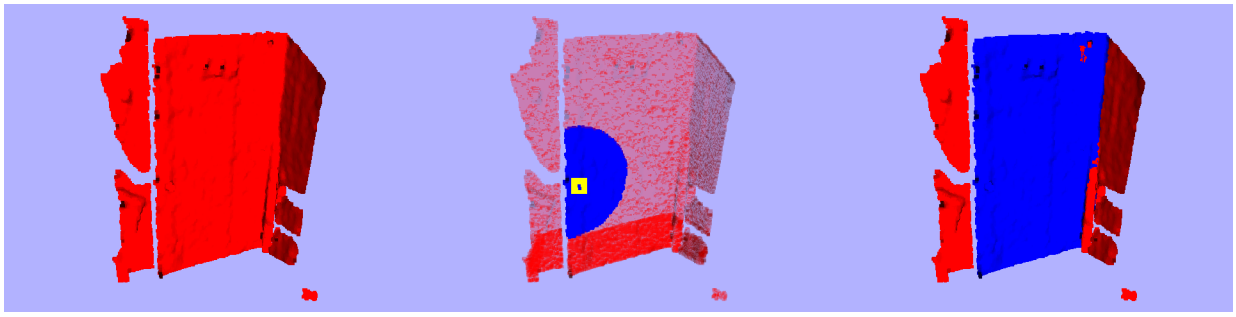
We now present a comprehensive application of our method to a real-world task, where



(a) Flipping a pitcher.

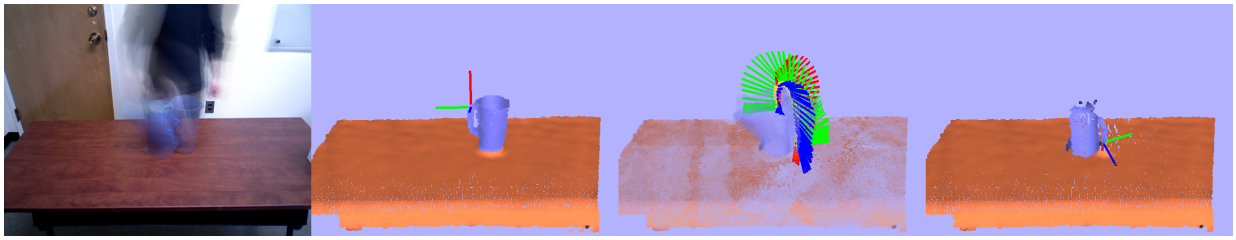


(b) Opening a drawer.

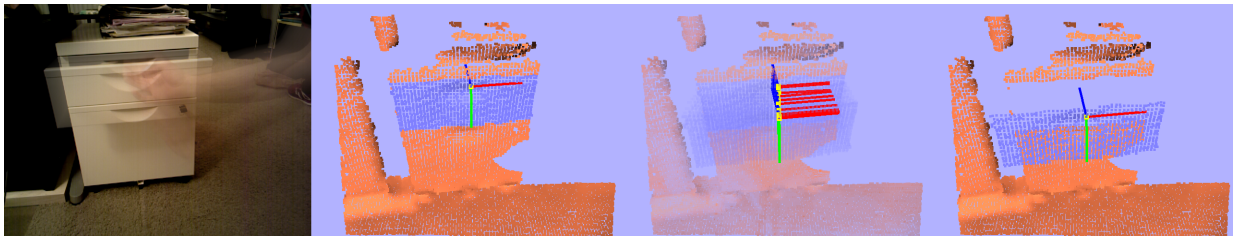


(c) Opening a door.

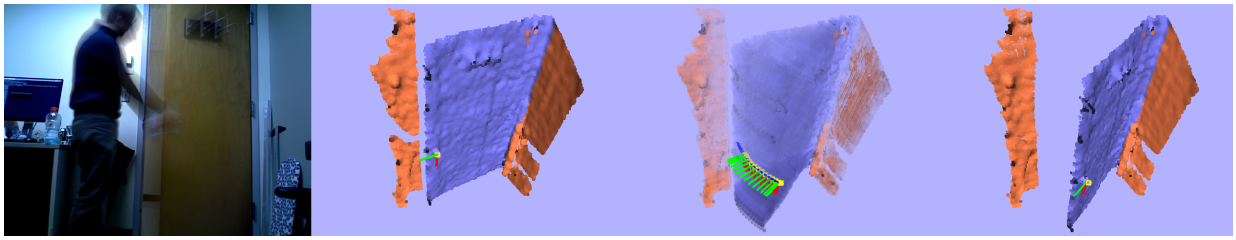
Figure 1.6: Motion segmentation of the manipulated object. First column: scene background points (the actor is removed). Second column: initial motion segment (blue) obtained by spectral clustering of point trajectories around contact area (yellow). Third column: Final motion segment.



(a) Flipping a pitcher.



(b) Opening a drawer.



(c) Opening a door.

Figure 1.7: Estimated rigid motion of the manipulated object. A coordinate frame is attached to the object segment (blue) at the contact point location (yellow). First column: temporal accumulation of color frames for the whole action duration. Second column: object state before manipulation. Third column: object trajectory as a series of 6DOF poses. Fourth column: object state after manipulation.

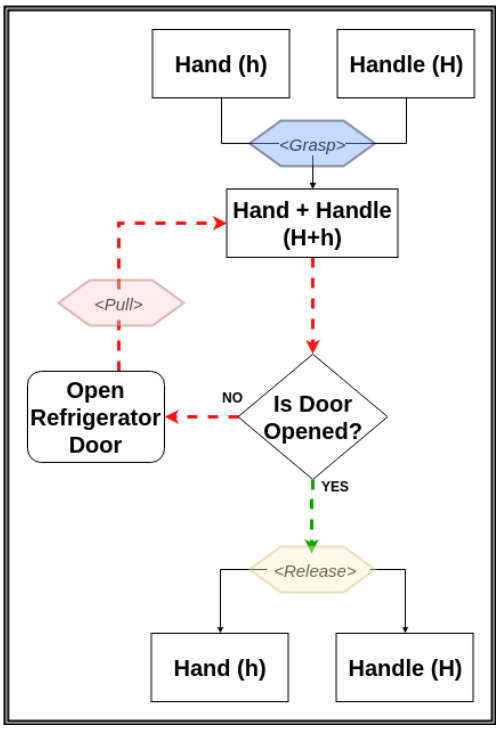


Figure 1.8: High-Level representation of opening a refrigerator door.

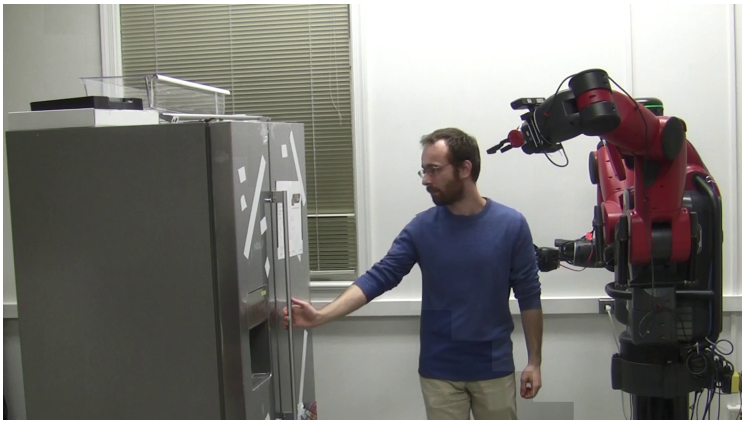


Figure 1.9: Robot observing a human opening a door.

a robot observes a human operator opening a refrigerator door and learns the process for replication. This can be seen in Fig. 1.9, where a RGBD sensor mounted to the robot's manipulator is used for observation. This process involves the segmentation of the human and the environment from the observed video input, analyzing the contact between the human agent and the environment (the refrigerator handle in this case), and finally performing 3D motion tracking

and segmentation on the action of opening the door, using our methods elucidated in Section 1.3. These analyses, and the corresponding outputs, are then converted into an intermediate graph-like representation, which encodes both semantic labeling of regions of interest, such as doors and handles in our case, as well as motion trajectories computed from observing the human agent. The combination of these allow the robot to understand and generalize the action to be performed even in changing scenarios.

We present a detailed explanation of each step involved in the process of a robot's replication of an action by observing a human. This entire process is visually described in Fig. 1.10, which separates our application into three phases, namely *preprocessing*, *planning* and *execution*.

1.5.1 Preprocessing Stage

The preprocessing stage is responsible for taking the contact point, object segments and their motion trajectories, as described in Fig. 1.1, and converting them into robot-specific trajectories for planning and execution. A visualization of this input can be seen in Fig. 1.11, where **(a)** depicts the RGB frame of the human performing the action. Subfigure **(b)** shows the contact point, highlighted in yellow, along with an initial object frame. Subfigure **(c)** demonstrates a dynamic view of the motion trajectory and segmentation of the door, along with the tracked contact point axes across time. Subfigure **(d)** shows the final pose of the door, after opening has finished.

In this stage, we exploit domain knowledge to semantically ground contact points and object segments, in order to assist affordance analysis and common-sense reasoning for robot manipulation, since that provides us with task-dependent priors. For instance, since we know

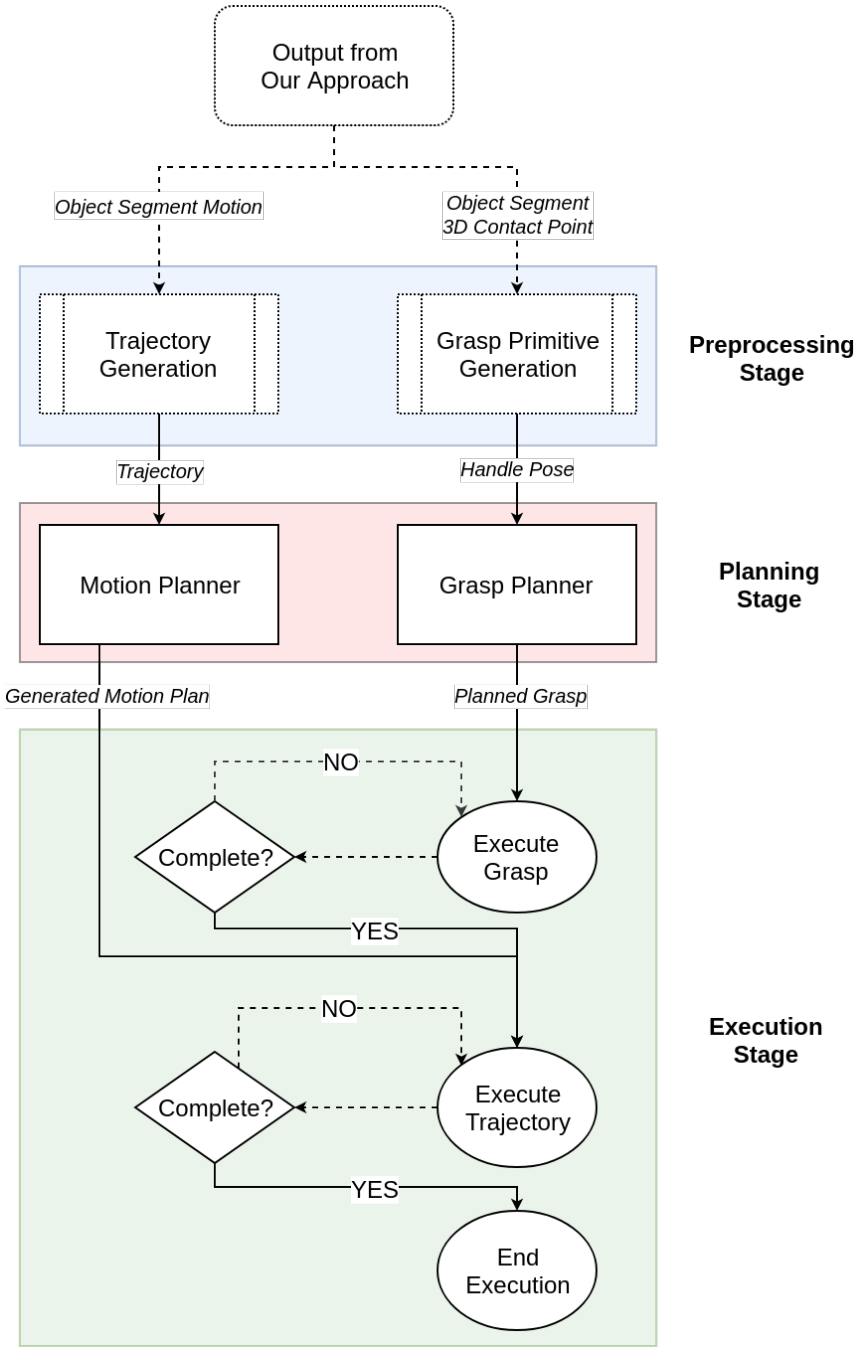
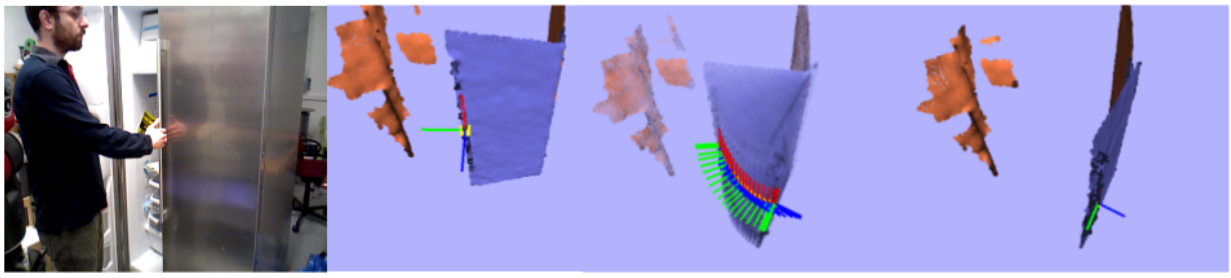


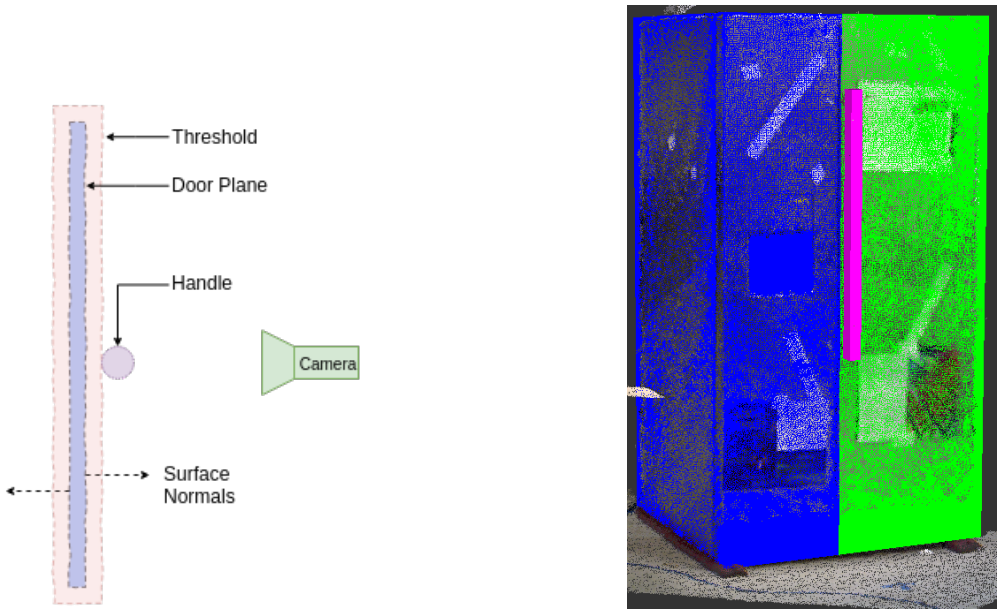
Figure 1.10: State transition diagram of our process.

that our task involves opening a refrigerator door, we can make prior assumptions that the contact point between the human agent and the environment will happen at the handle and any consequent motion will be of the door and handle only.



(a) (b) (c) (d)

Figure 1.11: Input to the *preprocessing* stage from our algorithm.



(a) Diagram depicting refrigerator handle detection.

(b) Point cloud of refrigerator with detected handle and door.

Figure 1.12: Handle detection.

1.5.1.1 Door Handle Detection

These priors allow us to robustly fit a plane to the points of the door (extracted object) using standard least squares fitting under RANSAC and obtain a set of points for the door handle (plane outliers). We then fit a cylinder to these points, in order to generate a grasp primitive with a 6 DOF pose, for robot grasp planning. The estimated trajectories of the object segment, as mentioned in Table 1.1, are not directly utilized by the robot execution system, but must instead

be converted to a robot-specific representation before replication can take place. Our algorithm outputs a series of 6DOF poses P_i for every time point $t_i \in T$. These are then converted to a series of robot-usable poses for the planning phase.

1.5.2 Planning Stage

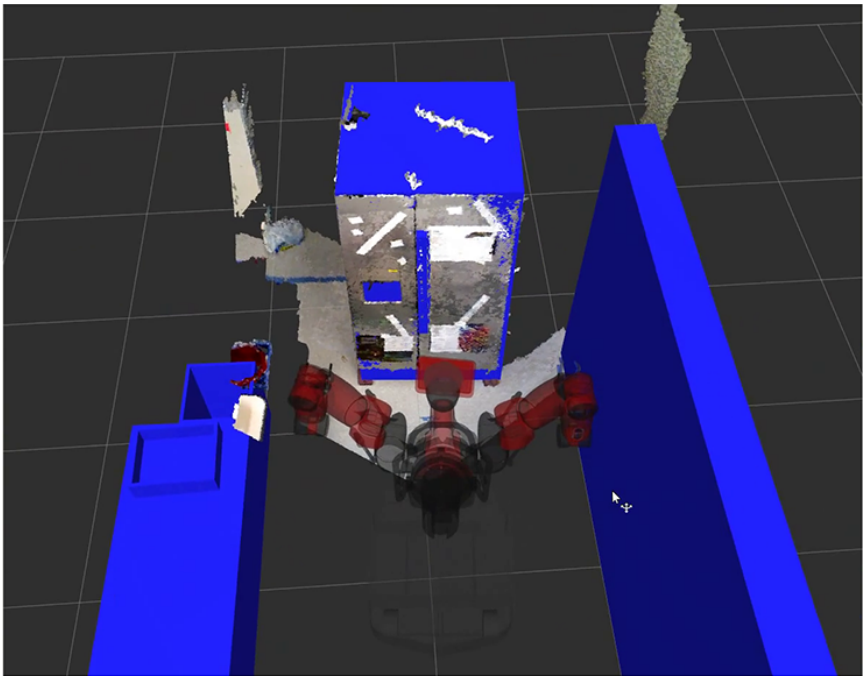


Figure 1.13: Visualization of planning stage

The outputs from the preprocessing stage, namely the robot-specific 6DOF poses of the handle and the cylinder of specified radius and height depicting the handle are passed in to the *planning* stage of our pipeline, for both grasp planning and trajectory planning. The Robot Visualizer (rviz) [70] package in ROS allows for simulation and visualization of the robot during planning and execution, via real-time feedback from the robot's state estimator. It also has point cloud visualization capabilities, which can be overlaid over primitive shapes. We use this tool for the planning stage, with the Baxter robot and our detected refrigerator.

1.5.2.1 Grasp Planning

Given a primitive shape, such as a block or cylinder, we are able to use the MoveIt! Simple Grasps [71] package to generate grasp candidates for a parallel gripper (such as one mounted on the Baxter robot). The package integrates with the "MoveIt!" library's pick and place pipeline to simulate and generate multiple potential grasp candidates, i.e. approach poses. There is also a grasp filtering stage, which uses task and configuration specific constraints to remove kinematically infeasible grasps, by performing feasibility tests via inverse kinematics solvers. At the end of the grasp planning pipeline, we have a set of candidate grasps, sorted by a grasp quality metric, of which one is chosen for execution in the next stage.

1.5.2.2 Trajectory Planning

The ordered set of the poses over time obtained from the *preprocessing* stage is then used to generate a Cartesian path, using the Robot Operating System's "MoveIt!" [72] motion planning library. This abstraction allows us to input a set of poses through which the end-effector must pass, along with parameters for path validity and obstacle avoidance. "MoveIt!" then uses inverse kinematics solutions for the specified manipulator configuration combined with sampling-based planning algorithms, such as Rapidly-Exploring Random Trees [73], to generate a trajectory for the robot to execute.

1.5.3 Execution Stage

The *execution* stage takes as input the grasp and trajectory plans generated in the *planning* stage and executes the plan on the robot. First, the generated grasp candidate is used to move the

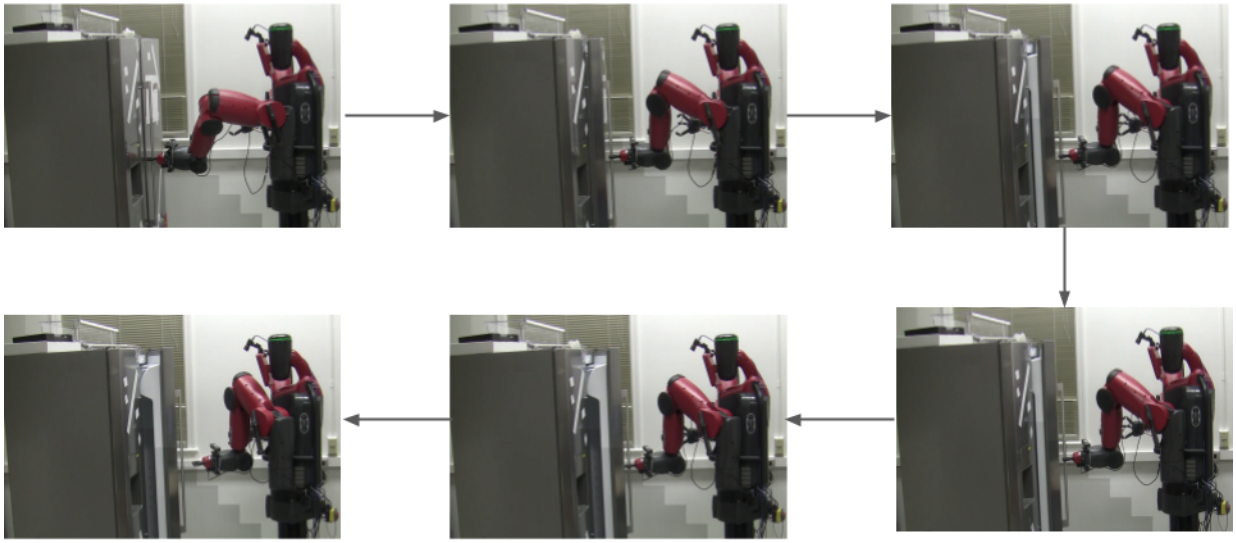


Figure 1.14: Robot replicating human by opening refrigerator

end-effector to a pre-grasp pose and the parallel gripper is aligned to the cylindrical shape of the handle. The grasp is executed based on a feedback control loop, with the termination condition decided by collision avoidance and force feedback. Upon successful grasp of the handle, our pipeline transitions into the trajectory execution stage, which attempts to follow the generated plan based on feedback from the robot's state estimation system. Once the trajectory has been successfully executed, the human motion replication pipeline is complete. This execution process is demonstrated by the robot in Fig. 1.14, beginning with the robot grasping the handle in the top-leftmost figure and ending with the robot releasing the handle in the bottom-leftmost figure, with intermediate frames showing the robot imitating the motion trajectory of the human.

In future work, we plan to implement a dynamic motion primitives [74] (DMP) based approach, which will allow more accurate and robust tracking of trajectories by the robot.

1.6 Conclusions

In this work, we have introduced an active, bottom-up approach for the extraction of two fundamental features of an observed manipulation, namely the contact points and motion trajectories of segmented objects. We have demonstrated the success of our algorithm on a diverse set of video inputs and described in detail its crucial role in a robot imitation scenario. Owing to its general applicability and the manipulation-defining nature of its output features, our method can effectively bridge the gap between observation and the development of action representations and plans.

There are many possible directions for future work. At a lower level, we plan to integrate *dynamic reconstruction* into our pipeline to obtain a more complete model for the manipulated object; at this stage, this can be achieved by introducing a step of static scene reconstruction before the manipulation happens, after which we run our algorithm. We also plan to extend our method so that it also can handle *articulated* manipulated objects, as well as objects that are *indirectly* manipulated (e.g., via the use of tools).

On the planning end, one of our future goals is to release a software component for the fully automated replication of *door opening* tasks (Section 1.5), given only a single demonstration. This module will be hardware agnostic up until the final execution stage of the pipeline, such that the generated plan to be imitated can be handled by any robot agent, given the specific manipulator and end-effector configurations.

Chapter 2: Computational Tactile Flow for Anthropomorphic Grippers

Grasping objects requires tight integration between visual and tactile feedback. However, there is an inherent difference in the scale at which both these input modalities operate. It is thus necessary to be able to analyze tactile feedback in isolation in order to gain information about the surface the end-effector is operating on, such that more fine-grained features may be extracted from the surroundings. For the tactile perception of the robot, inspired by the concept of the tactile flow in humans, we present the *computational tactile flow* to improve the analysis of the tactile feedback in robots using a Shadow Dexterous Hand.

In the computational tactile flow model, given a sequence of pressure values from the tactile sensors, we define a virtual surface for the pressure values and define the tactile flow as the optical flow of this surface. We provide case studies that demonstrate how the computational tactile flow maps reveal information on the direction of motion and 3D structure of the surface, and feedback regarding the action being performed by the robot.

2.1 Introduction

The concept of tactile flow, as it pertains to human manipulation and sensing, has been studied extensively both from a neurological as well as psychological perspective [75, 76]. The part of the brain responsible for receiving and processing tactile feedback is called the somatosensory

cortex and is subdivided into three main areas, known as Brodmann's areas 3a, 3b, 1 and 2. It is important to understand how the human brain processes and decodes tactile information in order to replicate it on robotic end-effectors. As our understanding of the human brain has developed, so has the understanding of its shortcomings. This has led to a rise in research on tactile/haptic illusions [75, 77], in which the misperceptions of the tactile signals received by the brain are leveraged to "fool" the mind into perceiving sensations that are not in sync with reality. This allows a better understanding of how certain parts of the brain react to different input signals.

We believe that the tactile flow provides a rich understanding of the frictional forces between surfaces of the hand and the object. The friction interaction can be categorized into static and dynamic interactions. In static interaction, the gripper does not move relative the object but it performs an action on or with the object such as opening the lid of a jar or pushing and screwing a screw in a wall. In the dynamic interaction, the gripper and the object have a relative motion with respect to each other, mostly for detection of the surface.

Most of the existing research on the tactile flow has been performed on human subjects [75, 77, 78] but has mostly considered the dynamic friction interaction. We use that as an inspiration for some of our case studies and present other case studies to cover the static friction interaction. It is important to attempt to replicate human studies on robotic systems, if we are to create robots that function at a level similar to humans. This is especially true for anthropomorphic grippers, such as the Shadow Dexterous Hand we use, since effective grasping using robotic systems is still far from being a reality. To the best of our knowledge, this is the first paper to simulate tactile flow on an electro-mechanical sensor, the BioTac, and we attempt to replicate some common and representative tactile experiments on the robot. We perform our experiments on specially designed experimental surfaces, which capture a variety of scenarios. We are able to use this

computational tactile flow to discover the direction of motion of the finger, detect different surface properties such as width and height of textures, recover 3D information about surface textures and also angles of these textures for the dynamic interaction. Moreover, we are able to detect some of the tactile flow patterns for various manipulation actions.

2.2 Related Work

The concept of tactile flow is not new and has been discussed and studied at length over the years. It has been found that tactile flow in humans is a highly sensitive and important source of proprioception and can override other cues to self-motion [78]. Harris, et al. conclude in this work that due to the sensitivity of tactile flow, they act as an "emergency override" and even minimal cues are enough to promote stability in a subject. This demonstrates the importance of having good tactile sensory mechanisms in robotic systems, if they are to become pervasive in day to day applications.

In a pilot study conducted by Ricciardi et al. [79], fMRI studies of the human brain have discovered that visual and tactile flow both result in activation of the V5/MT cortex, suggesting that a similar process occurs in the human brain when decoding motion cues from either visual or tactile cues. They define tactile flow as the "flow associated with displacement of iso-stress curves on the surface of contacting fingertips." This quantifies to our ability to perceive relative motion as well as changes in pressure on the surfaces in contact with the fingertip(s). In our presented work, we use this as the basis of our approach and show that conventional optical flow algorithms can indeed be used to discover tactile flow, using the BioTac [80] sensors. We call this process and our output, computational tactile flow using robotic sensors.

On a more practical level, tactile feedback remains crucial to human grasping and manipulation. In Chapter 7.3 of [81], the authors discuss the crucial nature of tactile slip, which encodes information about the relative motion between the skin and the surface. We demonstrate that our computational tactile flow also encodes this information and can be used as control feedback for dynamic robot grasping. Similarly, applications such as robot-assisted surgery can greatly benefit from tactile feedback, since it results in reduced (and more accurate) grasping forces on objects [82] and thus improve control over the robotic system.

We also take note of the concept of "tactile illusions" or "haptic illusions" [75, 77] which leverages the aforementioned property of similar visual and tactile activation regions, in order to induce misrepresentations caused by the dynamic tactile stimulation of the fingertips, and they have been shown to be similar to how optical illusions work and can be explained by tactile flow perception, which is the analog to optical flow. In future work, we would like to consider how haptic illusions affect our computation of simulated tactile flow and how we may find mechanisms to compensate for them.

2.3 Computational Tactile Flow

As it discussed in Sections 2.1 and 2.2, people construct tactile flows for their tactile sensory stimuli. In this section, we propose a method for computing the tactile flow for a robotic gripper equipped with tactile sensors.

2.3.1 Robotic Hardware

We are using Shadow Dexterous Hand equipped with BioTac SP [80] tactile sensors on its five fingers. Each of the BioTac SP sensors has multimodal sensory capabilities. They sense contact forces, micro-vibrations, and temperature flux. All of the sensory electronics are attached to a rigid core which receives the sensory information from a soft elastomeric skin through an incompressible conductive fluid. The attached sensors include (i) a hydro-acoustic pressure sensor for measuring the pressure of the whole fluid, (ii) a thermistor for measuring the vibrations and heat flux, and, (iii) 24 distributed electrodes, called taxels (tactile pixels), for measuring the changes of their electrical impedance in response to the deformation of the skin. Note that the impedance values approximate the pressure that the fluid between the skin and core applies at each electrode. Fig. 2.1 shows the BioTac SP tactile sensor with a layout of the electrode positions. The taxel 3-dimensional positions are not given by the manufacturer but provided in [83]. We use the same data in this work.

2.3.2 Smoothing the Data

The measured values of the sensor is noisy as it can be seen in Fig. 2.2a. Therefore, before performing any interpretation on the data, we smooth the data using Kalman method, as shown in Fig. 2.2b. First, we apply a Kalman filter with zero velocity model. We define the vector for the sensor readings of the 24 taxels at each time step t as

$$\boldsymbol{p}(t) = \begin{bmatrix} p(\boldsymbol{x}_1(t)) & \dots & p(\boldsymbol{x}_{24}(t)) \end{bmatrix}^T . \tag{2.1}$$

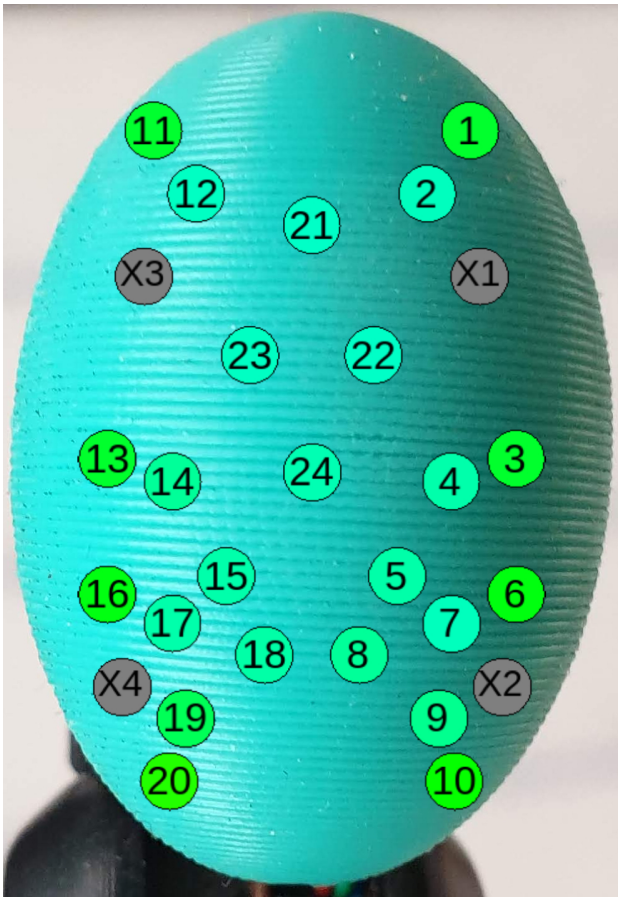


Figure 2.1: The BioTac SP sensor and layout of the taxels.

We choose the state transition matrix as $\Phi = \boldsymbol{I}_{24}$, where $\boldsymbol{I}_{24}$ is the 24×24 identity matrix. We choose $\boldsymbol{R} = 0.005\boldsymbol{I}_{24}$ and $\boldsymbol{Q} = 0.00015\boldsymbol{I}_{24}$ as the initial covariances of the measurement and process noises. The filter iteratively updates the data to $\boldsymbol{p}(t|t)$ for the measurement $\boldsymbol{p}(t)$ as

$$\boldsymbol{p}(t|t-1) = \Phi\boldsymbol{p}(t-1|t-1)$$

$$\boldsymbol{S}(t|t-1) = \Phi\boldsymbol{S}(t-1|t-1)\Phi^T + \boldsymbol{Q}$$

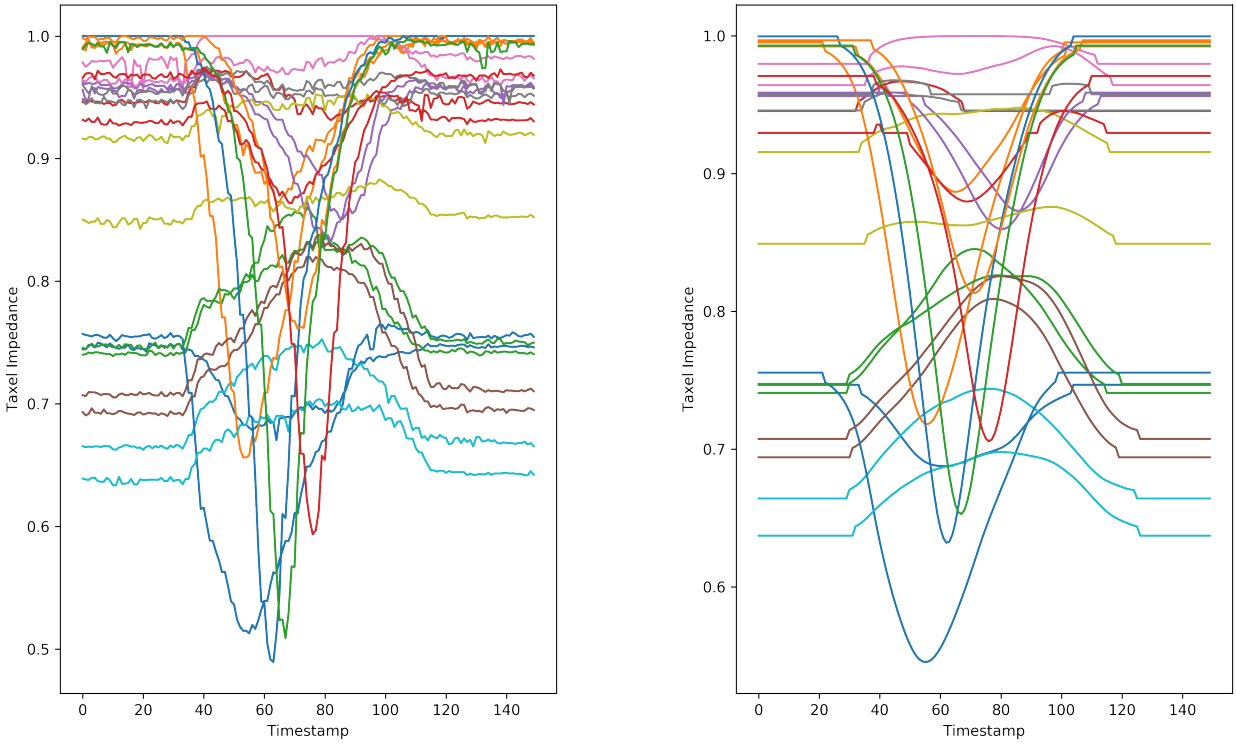
$$\boldsymbol{K}(t) = \boldsymbol{S}(t|t-1)[\boldsymbol{S}(t|t-1) + \boldsymbol{R}]^{-1}$$

$$\boldsymbol{S}(t|t) = (\boldsymbol{I}_{24} - \boldsymbol{K}(t))\boldsymbol{S}(t|t-1)$$

$$\boldsymbol{p}(t|t) = \boldsymbol{p}(t|t-1) + \boldsymbol{K}(t)(\boldsymbol{p}(t) - \boldsymbol{p}(t|t-1)),$$

where $\boldsymbol{S}$ is the state covariance matrix. We record the data over the time period $T = \{1 \dots N_T\}$. After applying the filter, we smooth the data given the whole observed data to $\boldsymbol{p}(t|T)$ by backward iterations as

$$
\begin{aligned}
\boldsymbol{L}(t) &= \boldsymbol{S}(t-1|t-1)\boldsymbol{\Phi}^T \boldsymbol{S}(t|t-1)^{-1} \\
\boldsymbol{S}(t-1|T) &= \boldsymbol{S}(t-1|t-1) + \\
&\quad \boldsymbol{L}(t)(\boldsymbol{S}(t|T) - \boldsymbol{S}(t|t-1))\boldsymbol{L}(t)^T \\
\boldsymbol{p}(t-1|T) &= \boldsymbol{p}(t-1|t-1) + \\
&\quad \boldsymbol{L}(t)(\boldsymbol{p}(t|T) - \boldsymbol{p}(t|t-1)).
\end{aligned}
$$



(a) Raw data

(b) Smoothed data

Figure 2.2: Sample trajectories of the sensor values.

2.3.3 Tactile Data Interpolation

Taxels provide observed information only on their specific locations on the BioTac sensors. To simulate the tactile flow on the sensor, we need the taxel values over the whole surface of the sensor. We can find these values by interpolating the data on the surface of the sensor. The 3D model of the sensor is not given but, using the least square error method, we realized that a half ellipsoid fits the 24 taxel positions well. Fig. 2.3 shows a 3D visualization of the 24 taxels and the fitted half-ellipsoid.

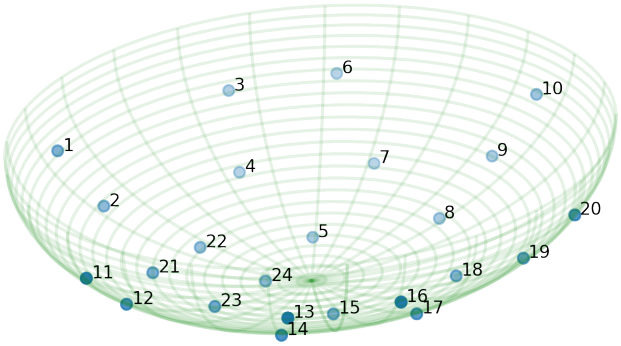


Figure 2.3: The taxels and the half ellipsoid.

For the interpolation of taxel values at $\boldsymbol{x}$, $\boldsymbol{x}_i \in \mathbb{R}^3$ on the core of the sensor, first note that a 3D point $\boldsymbol{x}$ on the ellipsoid can be presented by two parameters. The relation between the 3D representation, $\boldsymbol{x}$, and its corresponding 2D representation, $\boldsymbol{\theta} = [\theta\, \phi]^T$, can be written as

$$\begin{aligned} x &= a \sin\theta \cos\phi \\ y &= b \sin\theta \sin\phi \\ z &= c \cos\theta \end{aligned} \tag{2.2}$$

where a, b and c are the parameters of the half-ellipsoid, and

$$0 \leq \theta \leq \pi \quad \text{and} \quad \pi \leq \phi \leq 2\pi.$$

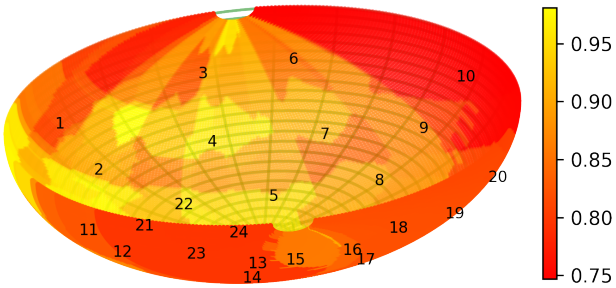
Therefore, there is a mapping $f : [0, \pi] \times [\pi, 2\pi] \rightarrow \mathbb{R}^3$ such that $f(\boldsymbol{\theta}) = \boldsymbol{x}$. The interpolation problem of taxel values can be approximated using a Gaussian kernel as

$$k(\boldsymbol{\theta}, \boldsymbol{\theta}_i) = \exp\left(-\frac{1}{2\sigma^2} d(\boldsymbol{\theta}, \boldsymbol{\theta}_i)\right), \tag{2.3}$$

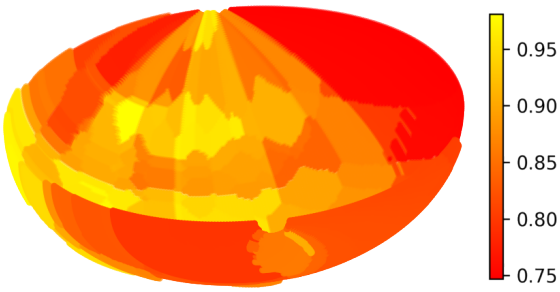
where $d(\boldsymbol{\theta}, \boldsymbol{\theta}_i)$ is the shortest distance between $\boldsymbol{x} = f(\boldsymbol{\theta})$ and $\boldsymbol{x}_i = f(\boldsymbol{\theta}_i)$ on the surface of sensor (called geodesic) and σ is the kernel parameter. The fitted ellipsoid on the surface is a tri-axial ellipsoid with a parameter for each axis. The problem of finding the minimum distance between two points on a tri-axial ellipsoid does not have a closed form analytical solution. Here, we approximate the geodesic distance numerically by integrating the Euclidean distances of a set of consecutive points on the surface which are between $\boldsymbol{x}$ and $\boldsymbol{x}_i$. The taxel values at each point, $p(\boldsymbol{\theta})$ can be estimated as

$$\hat{p}(\boldsymbol{\theta}) = \frac{\sum_{i=1}^{24} k(\boldsymbol{\theta}, \boldsymbol{\theta}_i) p(\boldsymbol{\theta}_i)}{\sum_{i=1}^{24} k(\boldsymbol{\theta}, \boldsymbol{\theta}_i)}, \tag{2.4}$$

where $p(\boldsymbol{\theta}_i)$ is the measured impedance value of the i-th electrode and $\hat{p}(\boldsymbol{\theta})$ is the estimated value of $p(\boldsymbol{\theta})$. Fig. 2.4a shows a sample of the contour plot of the interpolation of the impedance values.

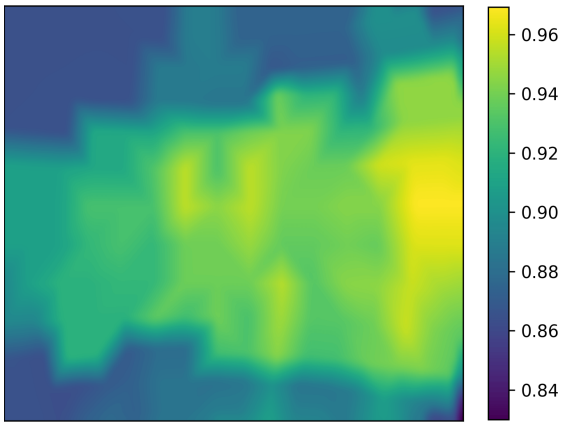


(a) A sample plot of the interpolated impedance values on the half ellipsoid.

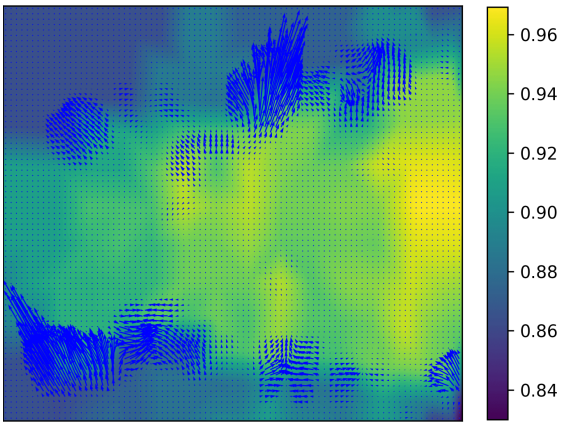


(b) The tactile surface.

Figure 2.4: Gaussian interpolated surfaces of the BioTac SP sensor



(a) Tactile frame



(b) Tactile flow

Figure 2.5: A sample tactile frame of projection in $x - y$ plane and its corresponding tactile flow.

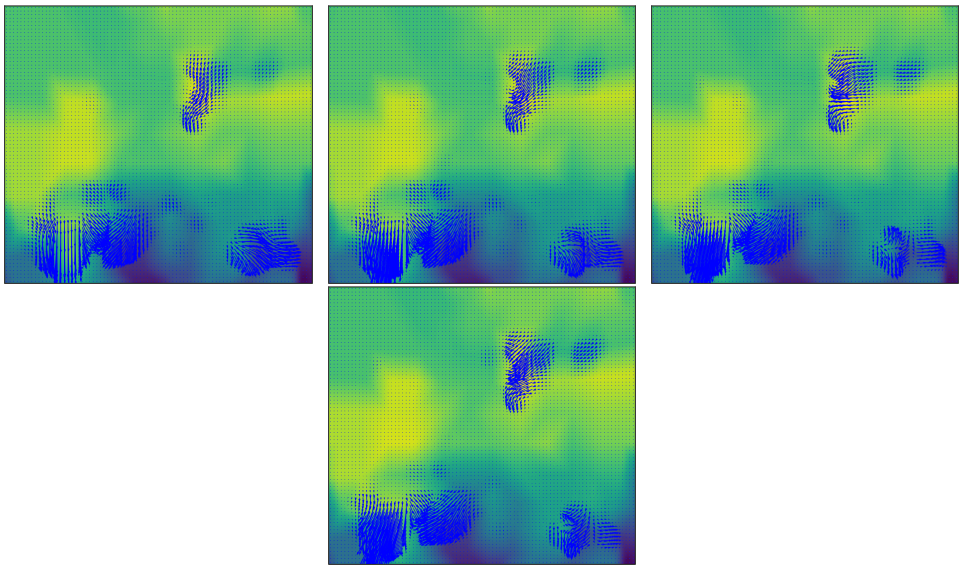


Figure 2.6: A sequence of tactile flows when the robot moves across a bump.

2.3.4 Tactile Flow Calculation

The impedance values over the surface of the sensor can be interpolated using (2.3). These values are directly proportional to the normal pressure at taxel positions. We can draw the normal vectors of the taxels with their impedance value as their length. The endpoint of these vectors can construct a new surface as shown in Fig. 2.4b. This surface is analogous to a 3-dimensional surface waving in the space and we call this surface the *tactile surface*.

Any local changes in the impedance values on the sensor result in motion in the corresponding neighborhood of the tactile surface. Using this analogy, we define the computational tactile flow as the optical flow for the motion of the tactile surface. Here, we calculate optical flows using projections of the tactile surface on various camera planes. We name a projection of the tactile surface on a camera plane as a *tactile frame*. Depending on the camera position, we can have different tactile frames. For example, if the camera is fixed at the top of the surface and faced down toward it, the optical flow perceived by this camera is the tactile flow in $x-y$ plane. Similarly, we can find the tactile flow projected in other planes. For instance, we can project the left and right half of the tactile surface to the $y-z$ plane and find two other tactile flows in the left and right of the sensor.

The impedance values, $p(\boldsymbol{x})$, do not remain constant and $\frac{d}{dt}p(\boldsymbol{x}) \neq 0$ but the intensity of each point on the surface, $I(x, y)$, remains unchanged for spatial and time perturbations and thus for each projection of the tactile surface, the optical flow is calculated using the oft-cited Horn and Schunck equation [84]

$$\frac{dI}{dt} = \frac{\partial I}{\partial x}v_x + \frac{\partial I}{\partial y}v_y + \frac{\partial I}{\partial t} = 0, \tag{2.5}$$

where $v = [v_x \quad v_y]^T$ is the tactile flow. In this work, we estimate the tactile flow based on Farnebäck polynomial expansion method [85]. This method approximates all neighborhoods of both frames as second-order polynomials and estimates the displacements by observing the polynomial expansion coefficients [85]. Fig. 2.5a shows a sample tactile frame of the projection of the tactile surface on the plane beneath the sensor (on $x - y$ plane). Fig. 2.5b shows the tactile flow for this frame. Here, the Shadow Hand is moving on a flat surface.

2.4 Experiments

2.4.1 Experimental Setup

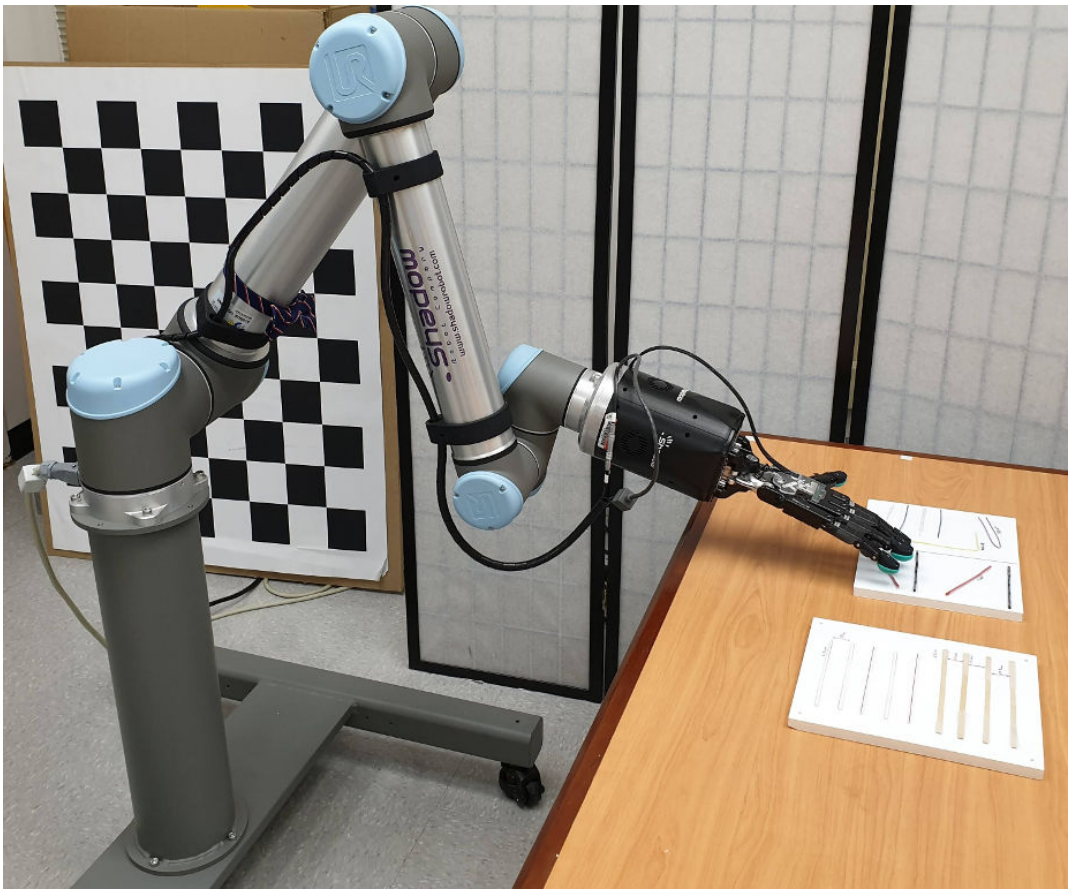
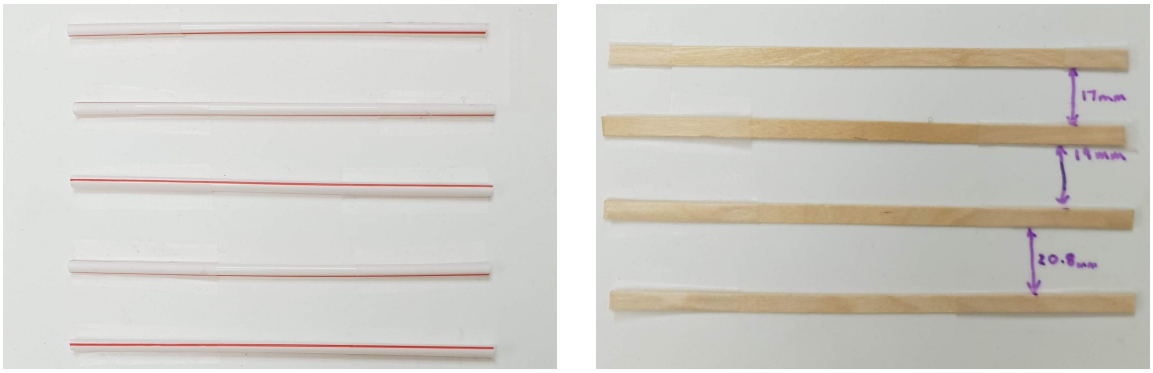


Figure 2.7: Our experimental setup showing the manipulator with the end-effector

As mentioned earlier in Sec. 2.1, tactile flow can be classified under two main classes, namely *static* and *dynamic*. Static tactile flow encodes flow patterns in situations when the sensor and the object remain almost stationary in contact, but the skin deforms due to static friction between the surfaces in contact. Dynamic tactile flow encodes information in scenarios where the finger has already overcome the static frictional forces and is moving over a surface. In both cases, the flow patterns encode information about the direction of motion, the relative magnitude of forces being applied as well as latent shape information. To demonstrate our approach, we perform several experiments on our custom-designed surfaces in order to best validate our hypotheses on using tactile flow and overall pressure to discover information about the surface.

Our hardware setup, shown in Fig. 2.7 includes a UR10 robotic manipulator, on which is mounted the Shadow Dexterous Hand as the end-effector. There is a table placed in front of the manipulator and end-effector setup, on which are mounted the various experiment boards. The robot is controlled through Robotic Operating System (ROS). The robot's position, as well as the end-effector's fingers, are tracked using the *tf* [86] library provided in ROS.



(a) Straws (b) Sticks

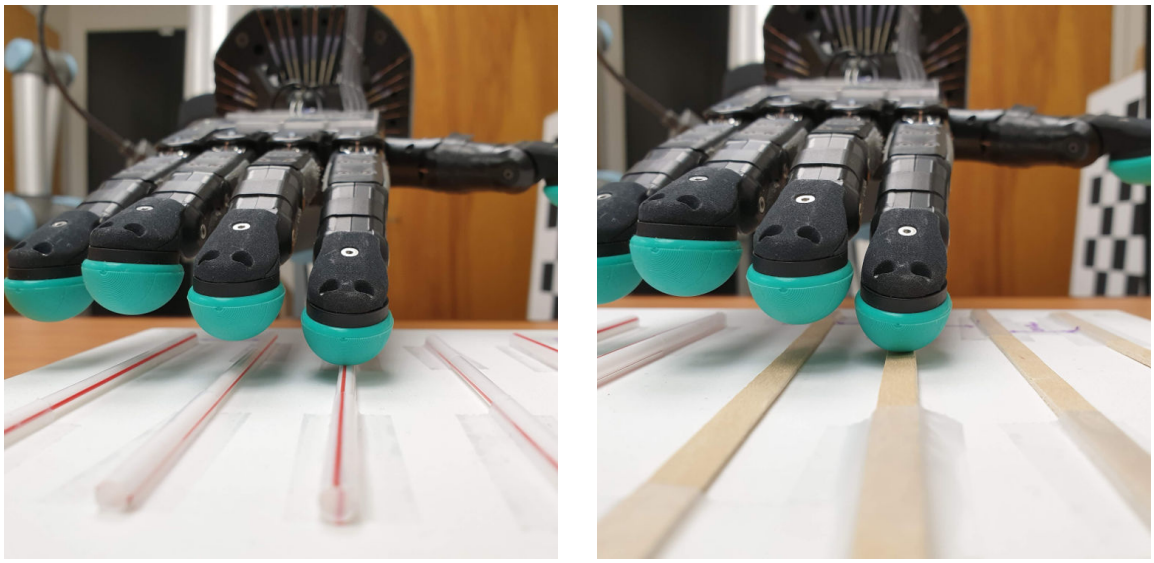
Figure 2.8: Experimental textured surfaces

For our experimental surfaces, we designed several boards on which we mounted different textured objects such as wires of varying thickness and wooden sticks. Examples of our experi-

mental surfaces can be seen in Fig. 2.8.

2.4.2 Dynamic Tactile Flow

The surfaces shown in Figs. 2.8a and 2.8b respectively have been carefully chosen to reflect different circumstances a finger might encounter. The straws surface is used to detect high and low surface differences and distinguish between angles of approach and departure. This is facilitated by the fact that at the time of contact, the finger "lifts off" the flat surface and only the part of the finger in contact with the straw surface is "activated". By comparing the computed flow, we are able to detect the direction from which the finger moved over the straw as well as measure the relative height of the straw by looking at the pressure peaks. This is demonstrated in Fig. 2.9a.



(a) Finger moving over the straw (b) Finger moving over the stick

Figure 2.9: Finger moving over different textured surfaces

A similar experiment is performed with the "sticks surface", which is designed to have a lower height difference to the ground surface but is wider than the straws. This is done to compare

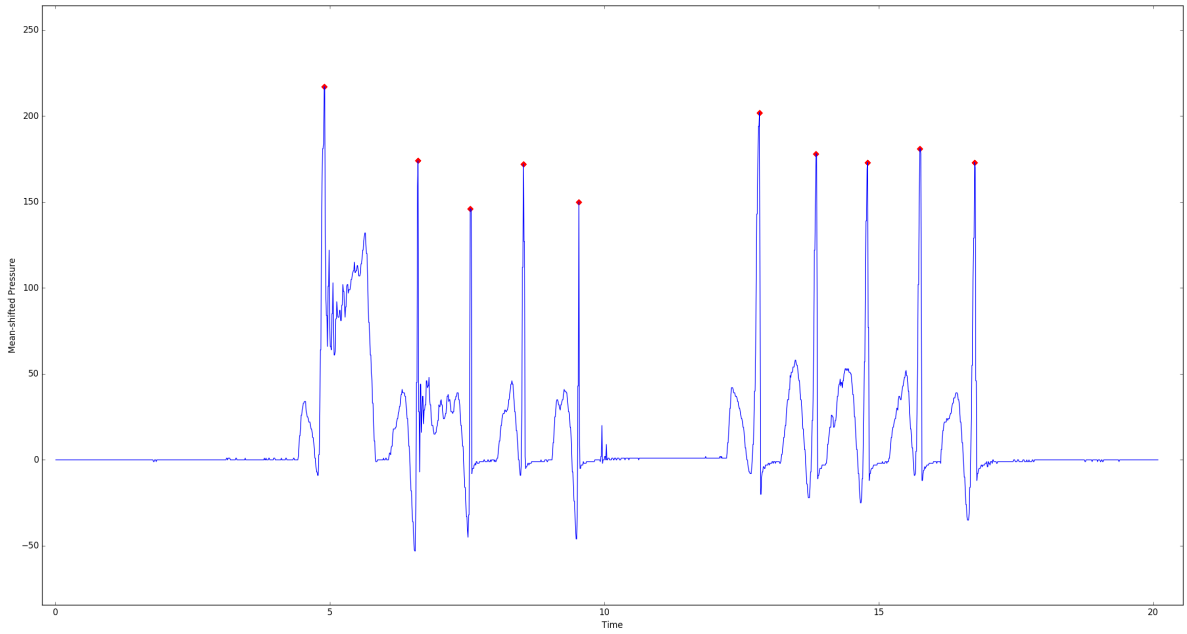
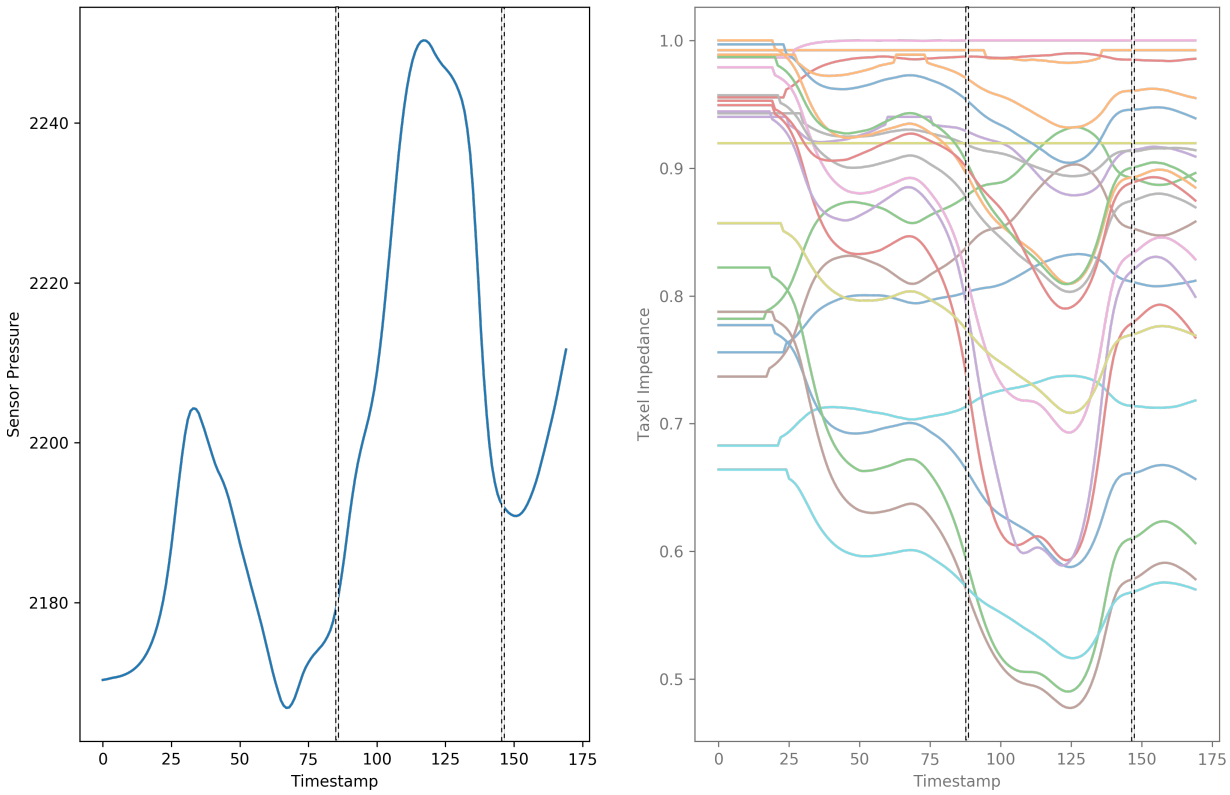


Figure 2.10: Peaks in Mean-Shifted Pressure

and contrast the readings in pressure and tactile flow between the two and demonstrate the utility of tactile flow in detecting the direction of motion over different surfaces. In this case, we are also able to estimate the width of the stick by analyzing the pressure plot.

The BioTac sensors provide both pressure and impedance readings, and we utilize both in our experiments. While the impedance values are used to construct the "tactile surface" described above, the pressure readings are useful to find regions of interest during the finger's movement over textured surfaces. We can use the peaks from the pressure readings in order to identify where significant events, such as bumps, approaching or departing an edge, etc. occurs and use those as priors for selecting frames for flow computation. This is facilitated by the fact that all our data is time-synchronized using a common clock. One such example is shown in Fig. 2.10 for the *straws* sequence.

2.4.2.1 Sticks Sequence



(a) Sensor Pressure (b) Taxel Impedances

Figure 2.11: The pressure and impedance plots for the finger moving over sticks

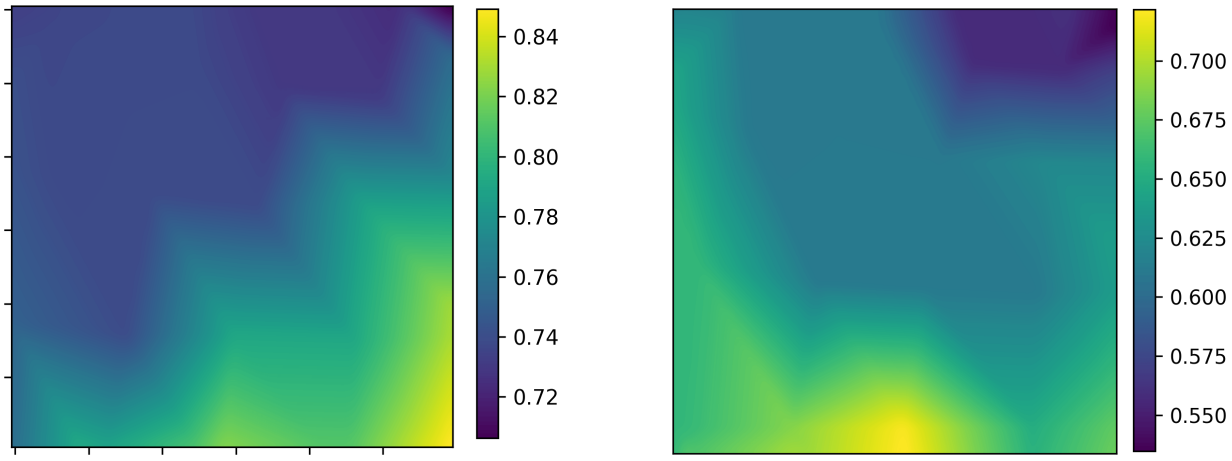
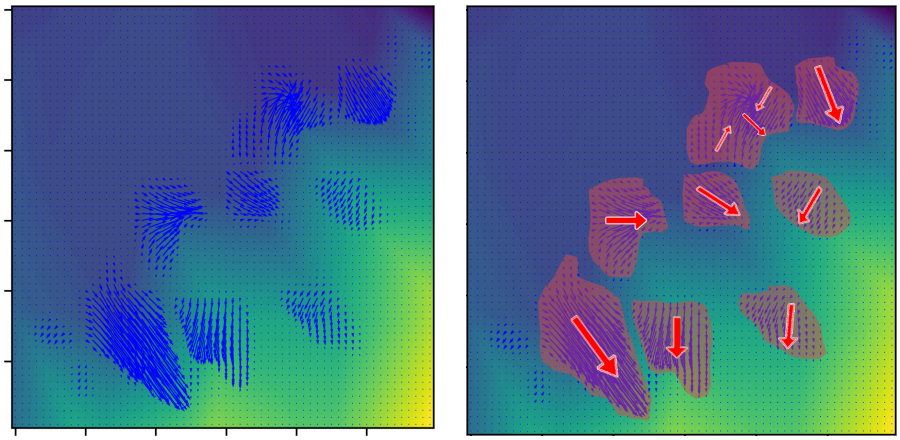
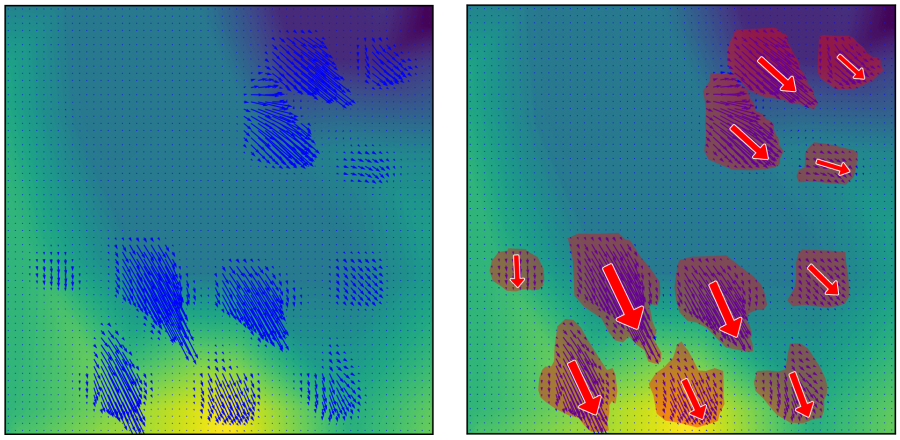


Figure 2.12: 2D projections of 3D taxel impedances for flow computation

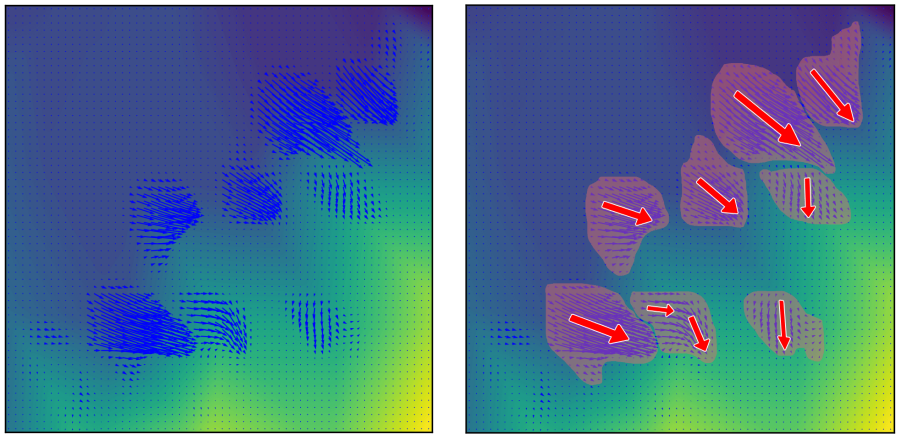
As can be seen in Fig. 2.11, the bands point us towards the region of interest (sequence of



(a) Before



(b) During



(c) After

Figure 2.13: Three tactile flow images and corresponding aggregated flow for the Sticks sequence

frames) which has been obtained from the pressure plot (Fig. 2.11a). The corresponding bands in the impedance plot show similar peaks and troughs, and we can thus focus on computing tactile

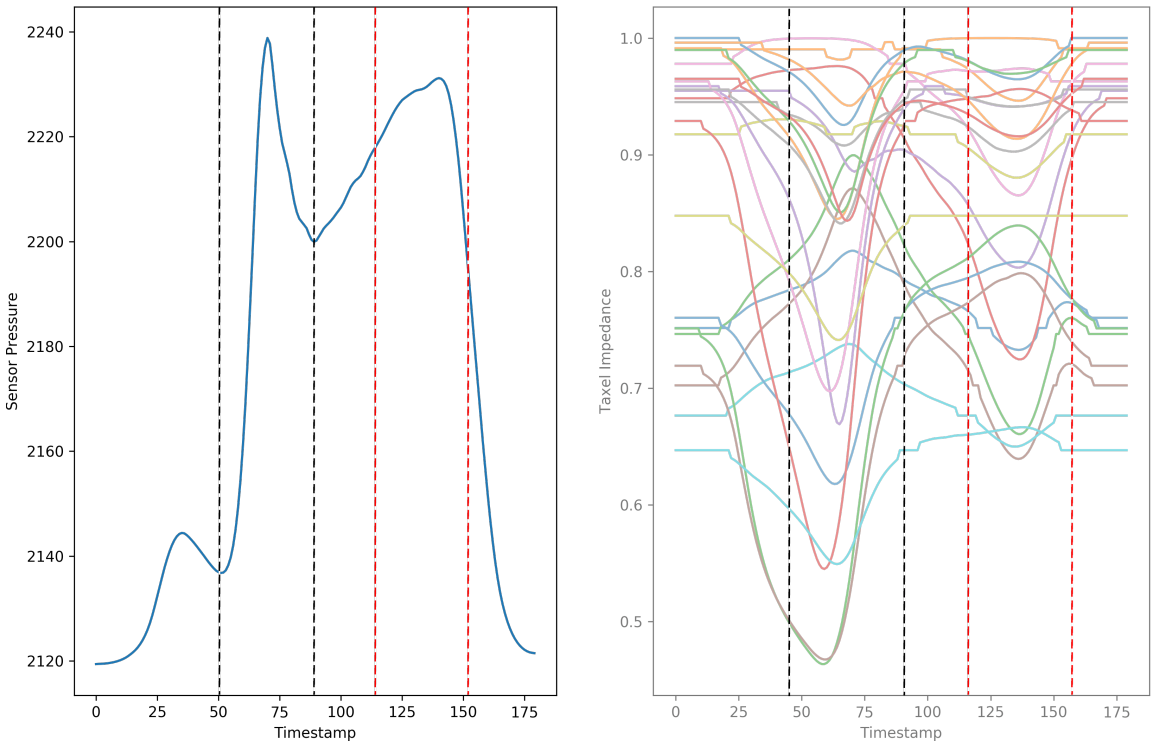
flow in those specific regions only. As described in our method, Fig. 2.12 shows two sample projections from the 3D ellipsoidal model to a 2D image. For our flow computations, we pick three distinct regions, namely one just before contact as the finger approaches the stick, one region as the finger is moving over the stick and one just after the finger has left the stick and is moving away. The computed flow in these respective regions are representative of different "classes" of motion and facilitate texture and motion classification that may be performed from the tactile flow data. This is elucidated in Fig.2.13. Alongside, we also show aggregated flow directions, computed from the generated flow fields. This is done to better visualize the direction of motion and easier parsing of the tactile flow data.

2.4.2.2 Straws Sequence

Similar to the *Sticks* sequence, for the *Straws* sequence, the time-synchronized taxel impedances and the pressure values are shown in Fig.2.14. As before, we use the pressure peaks to find regions of interest for which we compute the tactile flow sequences. This is illustrated in Fig. 2.15. These sequences also correspond to before, during and after the finger moves on the straw. The aggregated flow field directions are also provided.

2.4.3 Static Tactile Flow

In this part, we discuss our experiments and results on computing *static* tactile flow. These are the flow patterns experienced when the finger and the object in contact is not "actively" moving but is held in place by static frictional forces. Such patterns may occur during operating a tool, such as a screwdriver, or when opening a jar for example. In our experiments, we demon-

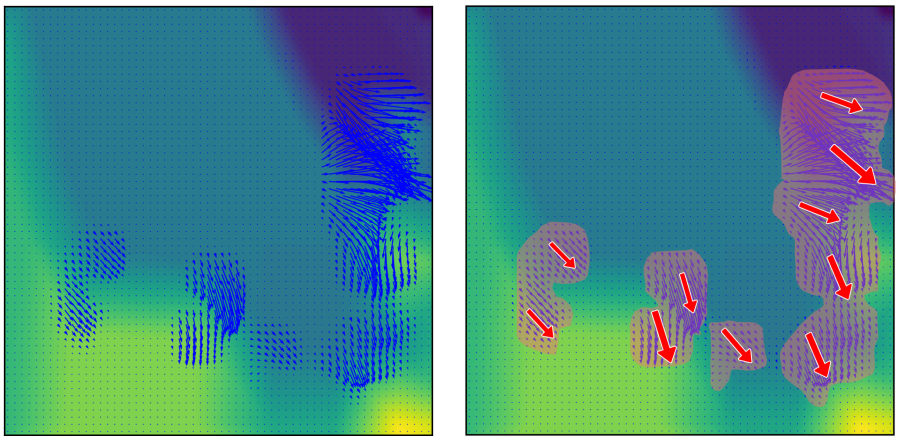


(a) Sensor Pressure (b) Taxel Impedances

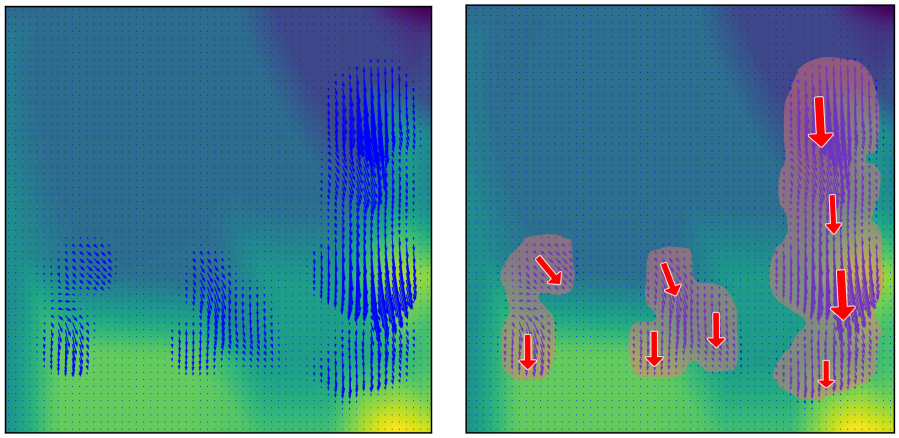
Figure 2.14: The pressure and impedance plots for the finger moving over straws

strate a simple instance of the robot grasping an empty bottle into which water is slowly poured from a height. As the weight in the bottle changes, the frictional forces also increase until the fingers are unable to exert sufficient forces to prevent the object from slipping or moving. Note that the robot is completely passive during this operation, i.e. there is no active reactionary force on the bottle to counteract these increasing frictional forces.

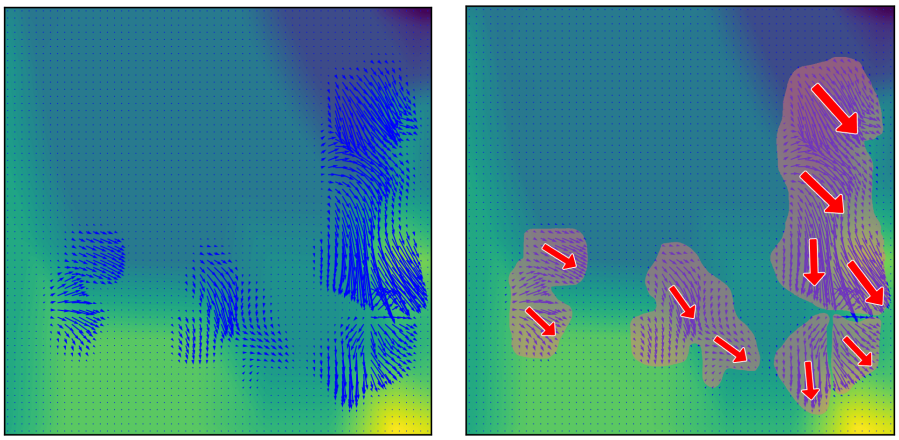
The combined taxel and impedance plots are shown in Fig. 2.16, where we have used the pressure peaks and troughs to find the regions of interest as before. Each of these segments are color coded to demarcate the important events in the experimentation procedure. The red segment is when the water has just been poured into the bottle, increasing the pressure felt by the fingers. The trough in the green segment is when the pressure is reduced due to inital slippage of the bottle, as it comes loose in the grasp. The magenta and cyan segments represent some rotational



(a) Before



(b) During



(c) After

Figure 2.15: Three tactile flow images and corresponding aggregated flow for the Straws sequence

motion of the bottle, and some downward motion respectively. Lastly, the yellow segment is when the bottle has started slipping completely.

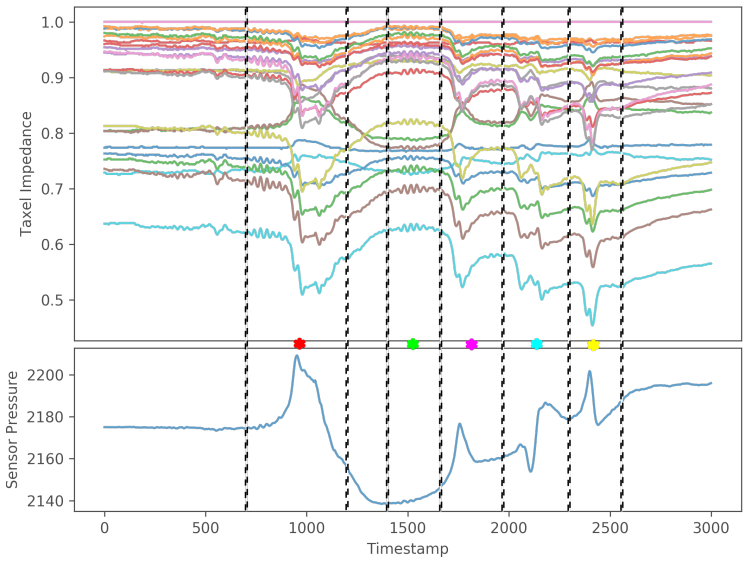


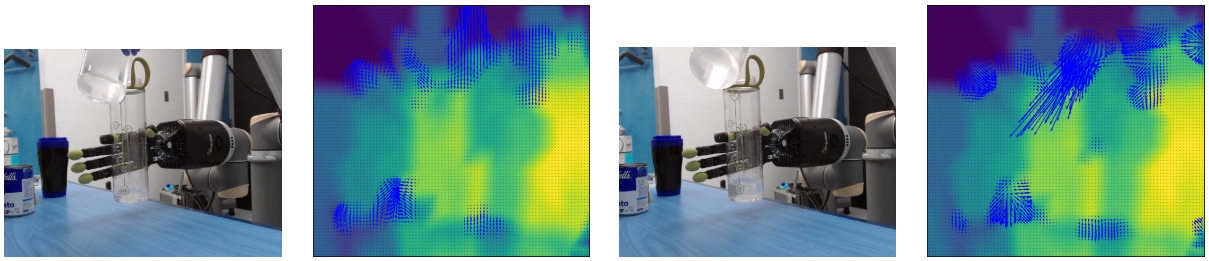
Figure 2.16: Combined taxel impedance and pressure plot

The corresponding scene configuration and tactile flow sequences are shown in Fig. 2.17, with the timestamps denoted accordingly.

2.5 Conclusions

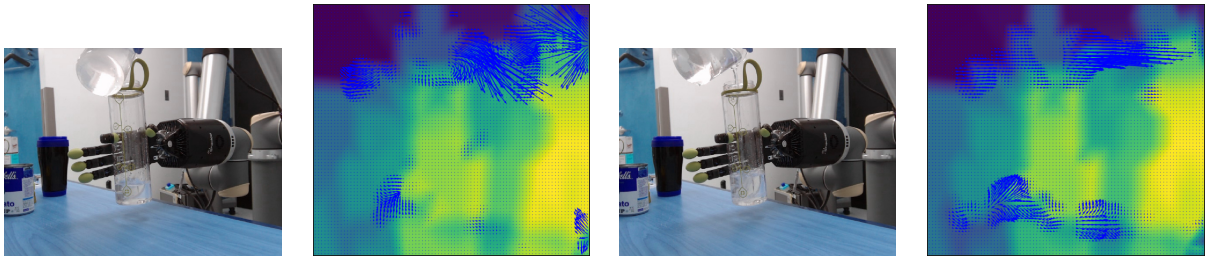
In this work, we present the theoretical premise for computing tactile flow using BioTac sensors. We also present a few experiments to validate our claims and methods. However, the practical usefulness of computing tactile flow demands another study and will be discussed in further detail as part of follow-up works.

We believe that, like humans, tactile flow is an essential component of robust grasping and proprioception, at par with conventional sensory mechanisms like visual or inertial means that are in use today. In order to achieve general grasping, it is necessary to have an understanding of and appropriate reaction to the forces in play between the object and the fingers. Our method of computational tactile flow provides the foundation for such an understanding, and in future work we aim to present a dynamic grasping mechanism based on feedback from these tactile flow data.



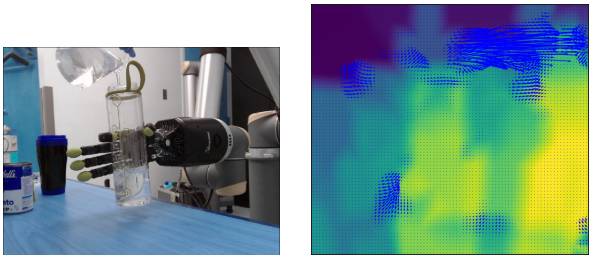
(a) Frame 970

(b) Frame 1500



(c) Frame 1750

(d) Frame 2250



(e) Frame 2390

Figure 2.17: Scene configuration and tactile flow sequence for grasping while water is being poured

Our plan for upcoming research is to use this computed tactile flow in order to facilitate better, more robust grasping strategies. We hope to use tactile flow as feedback in a grasping pipeline, in order to obtain dynamic and robust grasps, similar to how humans perform grasping without constant visual feedback.

Also, we consider tactile flow patterns to be representative of motion types in a given task. Thus, flow patterns can provide a way to encode motion primitives for a given task, and that can be emulated by the robot when trying to replicate a particular task. This gives rise to a general learning paradigm for grasping of novel objects, based on task goals. Latest machine learning

paradigms combined with conventional control algorithms can prove to be highly effective in learning these flow patterns, and reproducing them with the added benefit of generalizing to various scenarios.

Chapter 3: Grasping in the Dark: Zero-Shot Object Grasping Using Tactile Feedback

Grasping and manipulating a wide variety of objects is a fundamental skill that would determine the success and wide spread adaptation of robots in homes. Several end-effector designs for robust manipulation have been proposed but they mostly work when provided with prior information about the objects or equipped with external sensors for estimating object shape or size. Such approaches are limited to many-shot or unknown objects and are prone to estimation errors from external estimation systems.

We propose an approach to grasp and manipulate previously unseen or zero-shot objects: the objects without any prior of their shape, size, material and weight properties, using only feedback from tactile sensors which is contrary to the state-of-the-art. Such an approach provides robust manipulation of objects either when the object model is not known or when it is estimated incorrectly from an external system. Our approach is inspired by the ideology of how animals or humans manipulate objects, i.e. by using feedback from their skin. Our grasping and manipulation revolves around the simple notion that objects slip if not grasped stably. This slippage can be detected and counteracted for a robust grasp that is agnostic to the type, shape, size, material and weight of the object. At the crux of our approach is a novel tactile feedback based controller that detects and compensates for slip during grasp. We successfully evaluate and demonstrate our

proposed approach on many real world experiments using the Shadow Dexterous Hand equipped with BioTac SP tactile sensors for different object shapes, sizes, weights and materials. We obtain an overall success rate of 73.5%.

3.1 Introduction

Robotic agents, and their respective research fields, have generally proven useful in structured environments, crafted specifically for them to operate. We as robotics researchers envision in the near future, robots performing various tasks in our homes. For such robots to be successful when deployed "in the wild", they have to grasp and manipulate objects of various shapes, sizes, materials and weight which may or may not be present in their database.

Having the ability to grasp robustly and repeatedly is the primary way by which robots can affect their environment, and is the first step to performing more complicated and involved tasks. Robust grasping involves reliable perception of the object form (inference of pose, type of object and other properties) before grasping and a robust and continuous feedback loop to ensure that object does not slip (or fall) during grasping and manipulation.

In this work, we focus on the latter since it is required for grasping previously unseen or zero-shot objects, i.e. objects of unknown size, shape, material and weight, utilizing tactile feedback. Our method is inspired by the amazing grasping abilities of animals and humans [87, 88] for novel objects, and how they are able to "grasp in the dark" (with their eyes closed). To showcase that our method does not rely on an external perception input but rather only relies on tactile proprioception, we also demonstrate our method working on transparent objects that cannot be sensed robustly using traditional perception hardware, such as cameras or LIDARs.

3.1.1 Problem Formulation and Contributions

A gripper is equipped with tactile sensors and an object of unknown shape, size, material and weight is placed in front of the gripper. The problem we address is as follows: *Can we grasp and manipulate an unseen and unknown object (zero-shot) using only tactile sensing?*
We postulate that a robust grasp is achieved when enough force is applied to an object such that it is *just sufficient* to counteract gravity, thus suspending the object in a state of static friction experienced between the object and the finger. Our framework allows for robust grasping of previously unseen objects or zero-shot of varied shapes, sizes, and weights under the absence of visual input, due to our reliance solely on tactile feedback. Humans are capable of this feat from an early age [89], and it is an important ability to possess in scenarios with an absence of visual perception due to occlusions or the object not being present in the robot's knowledge-base. A summary of our contributions are:

- We propose a tactile-only grasping framework for unseen or zero-shot objects.
- Extensive real-world experiments showing the efficacy of the proposed approach on a variety of common day-to-day objects of various shapes, sizes, textures and rigidity. We also include challenging objects such as a soft-toy and a transparent cup demonstrating that our approach is robust.

3.2 Related Work

Most of the research in tactile-related grasping can be broadly divided into three categories, those that rely on purely tactile input, those that use vision-based approaches, and those that

perform end-to-end learning of grasping in simulation and attempt to transfer them to a real robot. We discuss some of the recent works done in all three categories, and their respective pros and cons.

3.2.0.1 Tactile Grasping

In 2015, [90] presented their force estimation and slip detection for grip control using the BioTac sensors where they try to classify "slip events" by looking at force estimation for the fingers. They proposed a grip controller, which helps adapt the grasp if slip is detected. They only used the pressure sensor data and only considered 2 or 3 fingers, and compared it with an IMU placed on the object itself.

In 2016, *BiGS: Biotac Grasp Stability* dataset [91] was released, which equipped a Barrett three-fingered hand with BioTac sensors and measured grasp stability on a set of objects, classified into cylindrical, box-like and ball-like geometries. In 2018, [92] presented their work on non-matrix tactile sensors, such as the BioTac, and how to exploit the local connectivity to predict grasp stability. They introduced the concept of "tactile images" and used only single readings of the sensor to achieve high rate of detection compared to multiple sequential readings. In 2019, TactileGCN [83] was presented. The authors used a graph CNN to predict grasp stability and they used the BioTac sensor data to construct the graph, but used only three fingers to grasp. They employed the concept of "tactile images" to convert grasp stability into an image classification problem. Their approach only deals with static grasps and does not consider the dynamic interaction between objects and the fingers. Another work [93] tackled this problem by extracting features from high-dimensional tactile images and infer relevant information to improve grasp

quality. But their approach is restricted to flat, dome- and edge-like shapes. The work by [94] used FingerVision [95] sensor mounted on a parallel gripper to generate a set of tactile manipulation skills, such as stirring, in-hand rotation, and opening objects with specified force. However, FingerVision is only appropriate for demonstraing proof of concept since it has a large form factor and is not robust. In 2020, a new tactile sensor "DIGIT" is presented in [96] that learns to manipulate small objects with a multi-fingered hand from raw, high-resolution tactile readings. In [97], the authors use the BioTac sensor as a way to stabilize objects during grasp using a grip force controller. The underlying assumption is that the shape of the object is known a-priori and repeatability with different shapes and sizes remains an ongoing challenge.

3.2.0.2 Visual Input (with Tactile Input) Grasping

In [98], the authors demonstrate a data set of slow-motion actions (picking and placing) organized as manipulation taxonomies. In [99], an end-to-end action-conditioned grasping model is trained in a self-supervised manner that learns re-grasping from raw visuo-tactile data, where the robot receives tactile input intermittently. The work in [100] leverages the innovation in machine vision, optimization and motion generation to develop a low-cost glove-free teleoperation solution to grasping and manipulation.

3.2.0.3 Simulation Based Grasping

Perhaps the most popular and famous papers in this category are [101, 102] from OpenAI, in which the authors demonstrate a massively parallel learning environment for the Shadow Dexterous Hand, and learn in-hand manipulation of a cube, and the solving of a Rubik's cube entirely

in simulation after which they are able to transfer said learning onto a physical robot. While impressive, their transfer learning approach requires near-perfect information about the joint angles of the Hand, as well as visual feedback regarding the position of the object. Levine, *et al*. [103] performs large-scale data collection and training on 14 manipulators for learning hand-eye coordination, directly going from pixel-space to task-space. In [104], a model-free deep reinforcement learning which can be scaled up to learn a variety of manipulation behaviors in the real world has been proposed, using general purpose neural networks. A State-Only Imitation Learning (SOIL) is developed in [105], by training an inverse dynamics model to predict action between consecutive states. The research problems attempted using perception and learning has seen limited progress due to the fact that vision does not provide any information regarding contact forces, regularly fails to reconstruct the scene due to occlusion or that the material properties of the object and the process of learning is time consuming, requires large amounts of data, and sometimes does not transfer to a real robot [106].

3.3 Overview

Our approach to solving the problem of grasping zero-shot objects is to define different execution stages for the robot and execute them accordingly. Figs. 3.1 and 3.2 shows the implementation and plots of these actions. An overview for controlling the robot to execute each of these actions are described in the rest of this section. Our proposed execution approach is implemented on a combination of the Shadow Dexterous Hand (we will call this ShadowHand, for brevity) equipped with BioTac SP tactile sensors (we will call this BioTac, for brevity) attached to a UR-10 robotic manipulator. An overview of our approach is as follows:

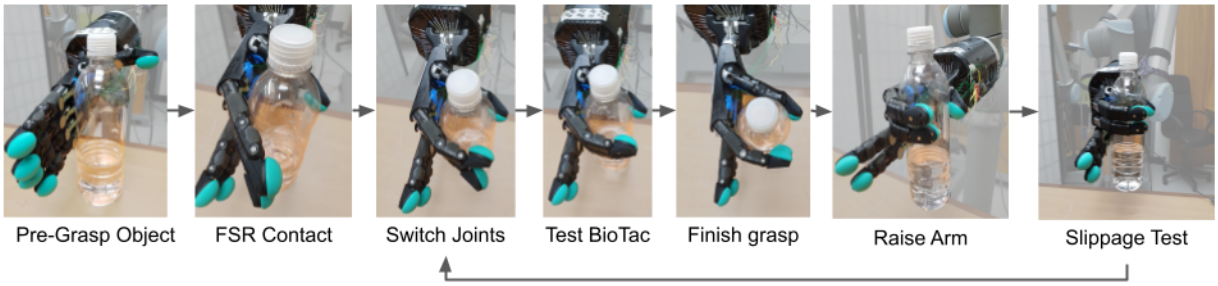


Figure 3.1: Grasp pipeline demonstration.

- *FSR Contact*: Control each finger such that its proximal phalanges (the phalanges nearest to the palm) reaches the object.
- *Switch Joints*: Control each finger's configuration such that its distal phalanges (the finger-tip) reaches the object.
- *Raise Arm*: Move the robotic arm configuration upwards while controlling the robotic hand's configuration to prevent object from slipping.

Before understanding our grasping framework, i.e. slip detection followed by a control policy for slip compensation, it is important to understand the basic structure of the ShadowHand. This will help the reader gain an intuition about the formulation of our control policy. The hardware setup is explained in the Sec. 3.4 followed by our software pipeline in Sec. 3.5.

3.4 Hardware Setup

3.4.1 Kinematic Structure of the Shadow Dexterous Hand

The ShadowHand has four fingers and an opposable thumb. Each of the fingers have four joints while the thumb has five joints. A representation of motion is shown in Fig. 3.3.

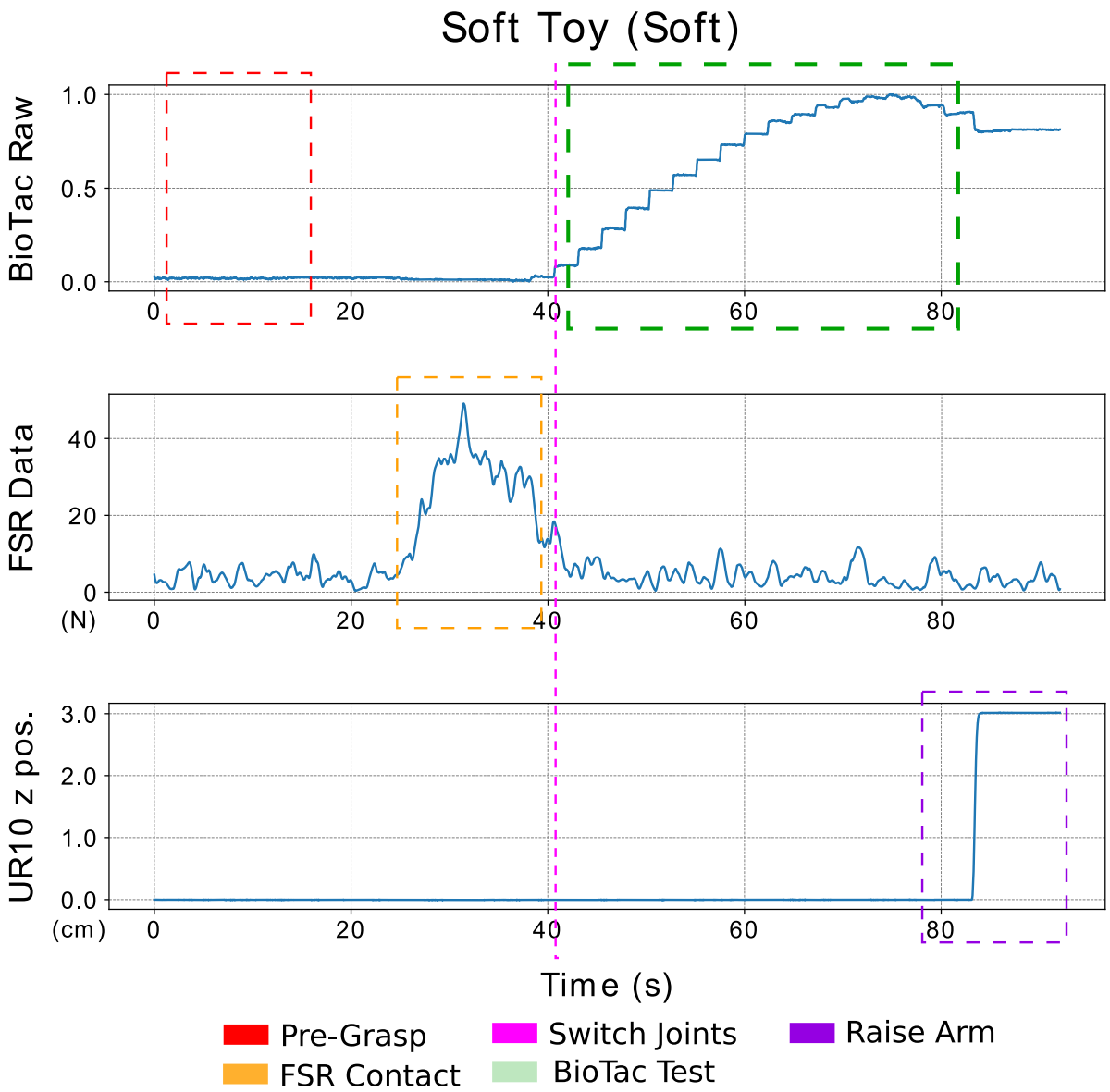


Figure 3.2: Variation of BioTac and FSR data as the soft toy is grasped and lifted from the table.

Fig. 3.3a demonstrates one finger and the joints specification it follows. Each finger has three links, also called phalanges, with one joint in between. From the top of the finger to the base, these are called the distal, middle and proximal phalanges respectively.

The fingers can be controlled by sending joint position values. The joint J_3 has two controllable ranges of motion, along the *sagittal* and *transverse* axes. This joint also has a minimum and

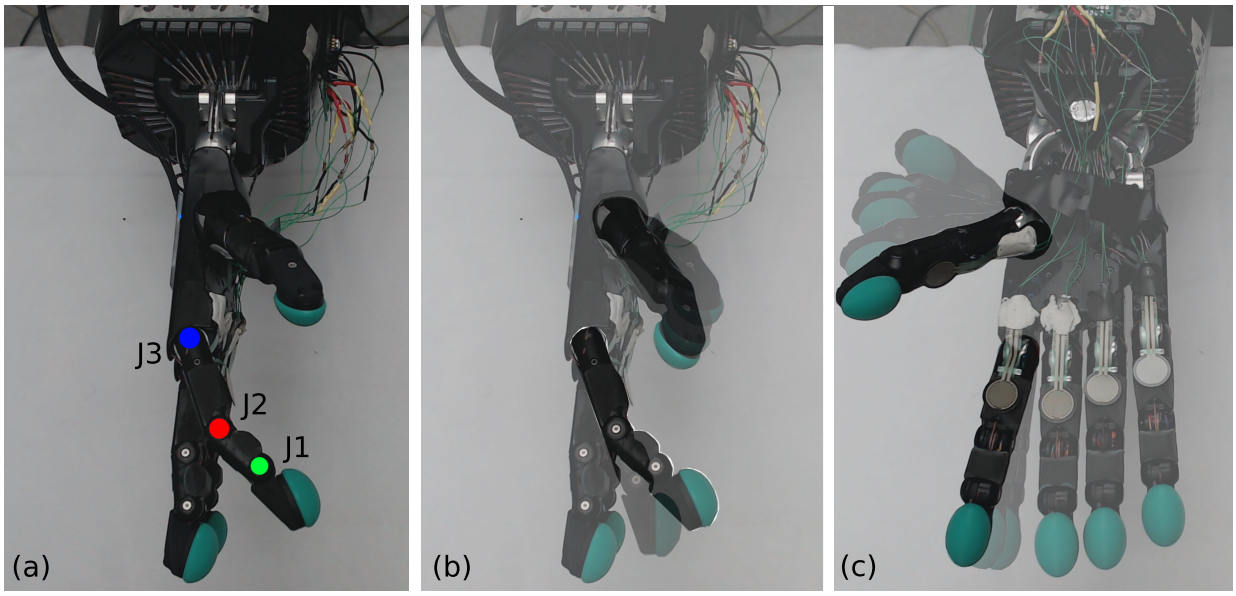


Figure 3.3: ShadowHand (a) joint nomenclature and (b, c) finger and thumb joints with their limit positions.

maximum range of 0° to 90° respectively. The joints J_1 and J_2 are different in that, similar to the human hand, they are coupled internally at a kinematic level and *do not move independently*. They individually have a range of motion between 0° to 90°, but are underactuated. This means that the angle of the middle joint, i.e. J_2 is always greater than or equal to the angle of the distal joint, i.e. J_1 which allows the middle phalanx to bend while the distal phalanx remains straight.

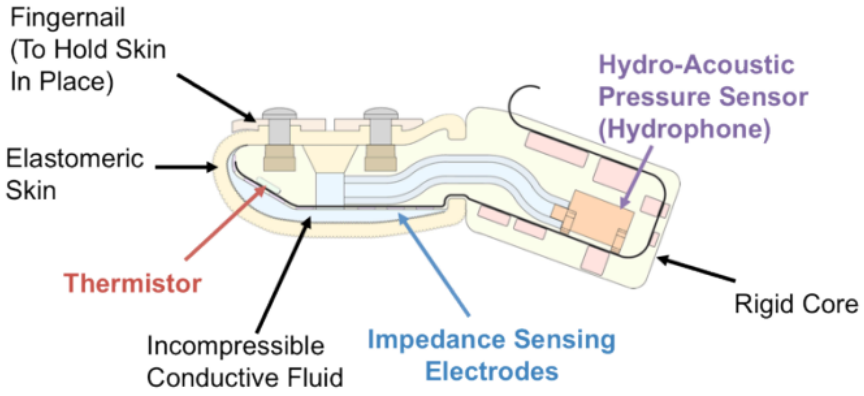
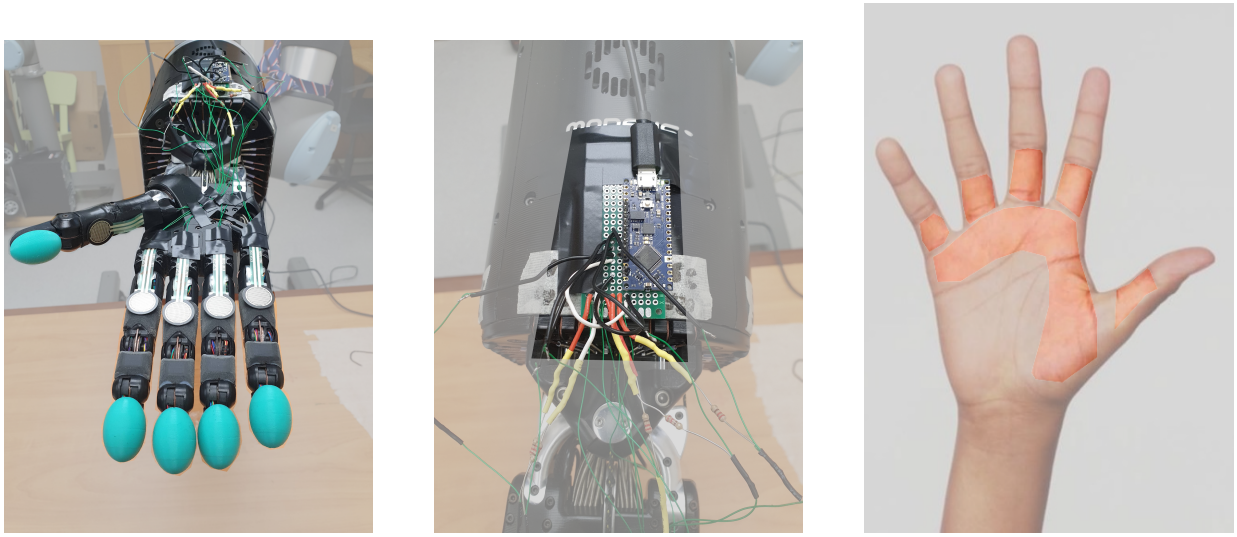


Figure 3.4: Cross-section of the BioTac sensor.

Currently available commercial tactile sensors lie on a spectrum spanning from accuracy on one end to form factor on the other. These sensors can either have high accuracy while sacrificing anthropomorphic form factor or can be designed similar to human fingers, while having a relatively poor accuracy at tactile sensing. The choice of the sensor depends on the task at hand, which in our case is to grasp zero-shot objects. To this end, we select the ShadowHand equipped with the BioTac sensors in an effort to be biomimetic.

Using a combination of impedance sensing electrodes, hydro-acoustic pressure sensors and thermistors, the BioTac sensor is capable of sensing three of the most important sensory inputs that one needs for grasping, namely deformation and motion of stimuli across the skin, the pressure being applied on the finger and temperature flux across the surface. The internal cross-section of the BioTac SP is shown in Fig. 3.4.



(a) FSRs (b) Connections (c) Contact Regions

Figure 3.5: (a) FSRs, (b) FSR connection to Arduino Nano, and (c) Regions of contact when grasping.

The human hand can grasp objects of various shapes, sizes and masses without having seen them previously. This ability to grasp previously unseen objects in the absence of visual cues is

possible only due to the presence of tactile sensing over a large surface area, through the skin. In Fig. 3.5c, the highlighted parts show the primary regions of contact when grasping is performed. These regions make first contact with the object being grasped and apply the most amount of force, due to the large surface area. To mimic similar tactile characteristics on the ShadowHand, we equip it with additional sensors at the base of each finger and the thumb.

We utilize Force Sensitive Resistors (FSRs) for this purpose, which are flexible pads that change resistance when pressure is applied to the sensitive area (See Fig. 3.5a). These FSRs work on the principle of a voltage divider circuit, Fig. 3.5b, and have a voltage drop inversely proportional to the resistance of the FSR.

We calibrate our sensors using a ground-truth force measurement unit, for 0N to 50N of force, using simple regression. We map a series of readings from the FSR to the corresponding force values in Newtons on the force measurement unit, and fit a regression line to these points. This is sufficient to measure contact forces between the fingers and an object during grasping.

3.5 Software Pipeline

3.5.1 Grasp Controller

Our pipeline, defined in Algos. 1, 2, starts at the pre-grasp pose where the fingers are fully extended and the thumb is bent at the base to a 70° angle, which is optimal for grasping most objects due to having the maximum volume coverage by the trajectories of the finger tips. We explain the distinct parts of our pipeline in the following sections.

3.5.1.1 Initialization

The controller begins by performing a tare operation using 50 readings of each BioTac sensor and computing their respective means. Successive readings are then min-max normalized, within an adjustable threshold of ± 200 of this mean, to ensure that each sensor's biases are taken into account, as well as to provide a standardized input to the control loop.

3.5.1.2 Hand-Object Contact

Once the initialization process is complete and baseline readings have been established, the hand controller begins actuating the J_3 joints of all the fingers and J_4 of the thumb. This is done by sending the appropriate joint control commands. The current joint values are obtained from the ShadowHand, checked against the maximum joint limits of each finger (90° for J_3), and increased by a small angle $\delta\theta$. The J_3 and J_4 joints of the fingers and thumb respectively are moved until it registers a contact with the object, as measured by the FSR readings. This establishes an initial reference point for the hand to begin refining the grasp, and the controller switches to the control policy for the coupled joints.

3.5.1.3 Preliminary Grasp

At this stage, since the base of each finger and thumb have made initial contact with the object, the control policy switches to the coupled joints so that the fingers can begin to "wrap around" the object. We now activate the coupled J_1 and J_2 joints (controlled using a virtual J_0 joint defined in Eq. 3.1) of the fingers and the J_1 joint on the thumb. In this stage, the $\delta\theta$ is inversely proportional to the the normalized sensor data from the BioTac sensor.

$$J_0 = \begin{cases} J_2 & \text{if} \quad J_2 \in [0, 90]^\circ \\ \\ J_1 & \text{if} \quad J_2 > 90^\circ \end{cases} \tag{3.1}$$

Intuitively, this means that when there is little or no contact between the fingers and the object, the controller sends out larger joint angle targets, causing the fingers to move larger distances. Once contact is made, the controller moves the fingers at progressively smaller increments, thus allowing for a more stable and refined grasp.

We use a Beziér curve to generate an easing function that maps our normalized BioTac sensor data to a normalized angle (in radians), between the joint limits of the respective joint. This mapping is then converted into a usable control output $\delta\theta \in [\theta_{\text{min}}, \theta_{\text{max}}]$.

The Beziér curve is generated by the parametric formula

$$\theta_\beta = S^2_{\text{BioTac}} \times (\kappa_1 - (\kappa_2 \times S_{\text{BioTac}})) \tag{3.2}$$

where κ_1 and κ_2 are the Beziér control points, S_{BioTac} is the instantaneous normalized BioTac reading and θ_β is the mapped Beziér curve output. We then compute control output $\delta\theta$ as follows

$$\delta\theta = B_1 + \frac{(\theta_\beta - A_1) \times (B_2 - B_1)}{A_2 - A_1} \tag{3.3}$$

where $[A_2, A_1] \in [0, 1]$ and $[B_2, B_1] \in [\theta_{\text{min}}, \theta_{\text{max}}]$.

We set a termination threshold $\tau_{\text{termination}} = 0.1$ on the BioTac sensor values such that the Hand controller stops executing as soon as a minimal level of contact is detected. Once all the fingers

and the thumb have reached the preliminary grasp state, we exit the control loop.

Algorithm 1: Implementation of the initial grasp controller

Procedure `Reset`**:**
  repeat
    Move Hand to pre-grasp pose
  until *pre-grasp reached*
Baseline = mean_{50} of BioTac
repeat
  $J_3 = J_3 + \delta\theta$
  Actuate J_3
until *Until FSR contact*
Switch to Coupled Joints Controller
Compute control output based on Eqs. 3.2, 3.3
repeat `/* Until fingertip touches object */`
  while $J_1 + J_2 \leq 180°$ **do**
    $J_2 = J_2 + \delta\theta$
    if $J_2^t - J_2^{t+1} < 0.1$ **then** `/* Joint limit */`
      $J_1 = J_1 + \delta\theta$
      Actuate J_1
    else
      Actuate J_2
    end
  end
until *Until* $P_{dc}^t - P_{dc}^{t+1} \geq \tau_{termination}$

3.5.2 Slippage Detection

One of the crucial aspects of our proposed grasping pipeline is the ability to detect, and react to objects slipping between the fingers during grasp and move. This reactive nature of our controller allows for precise force applications on zero-shot objects, i.e., objects of unknown masses, sizes or shapes.

For slippage detection, we utilize data from the BioTac sensor. Fig. 3.6 show two sets of plots of sensors readings captured during grasping a wine glass, without and during slip respec-

Algorithm 2: Implementation of picking with slip detection

Procedure `SlippageDetection`**:**
  while *Hand has not been raised* **do**
    repeat
      Actuate J_1
      Move UR-10 up
    until *slip detected*
  end

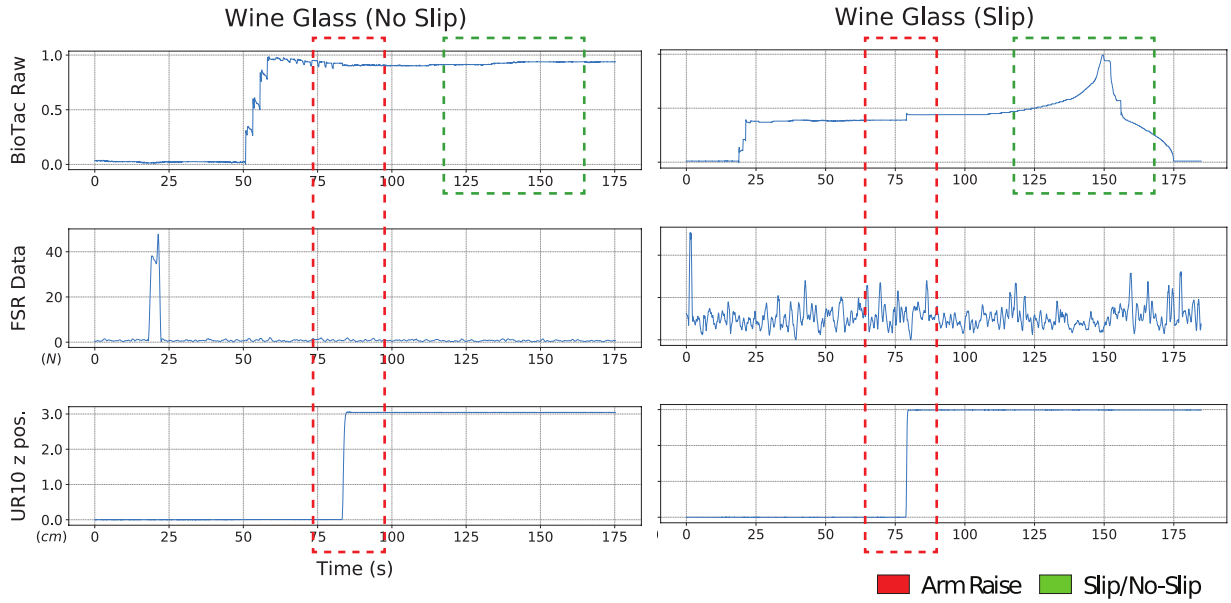


Figure 3.6: Plot of BioTac and FSR sensor data for without and with slip on the wine glass experiment.

tively. From top to bottom, the graphs represent the BioTac, FSR readings and the position of the UR-10 along the z-axis. For visual clarity, we plot data for only the first finger. The difference in readings during slip versus without slip is quite evident, with several micro-vibrations in the BioTac data while the object slowly slips off the hand. This is due to the frictional properties of the BioTac skin, as well as the weight of the object. These vibrations are absent when the object does not slip, and the readings maintain a mostly stable baseline. Our slip detection algorithm works by measuring and tracking the change in gradient of the sensor readings over a non-overlapping time-window of $\Delta t = 100ms$.

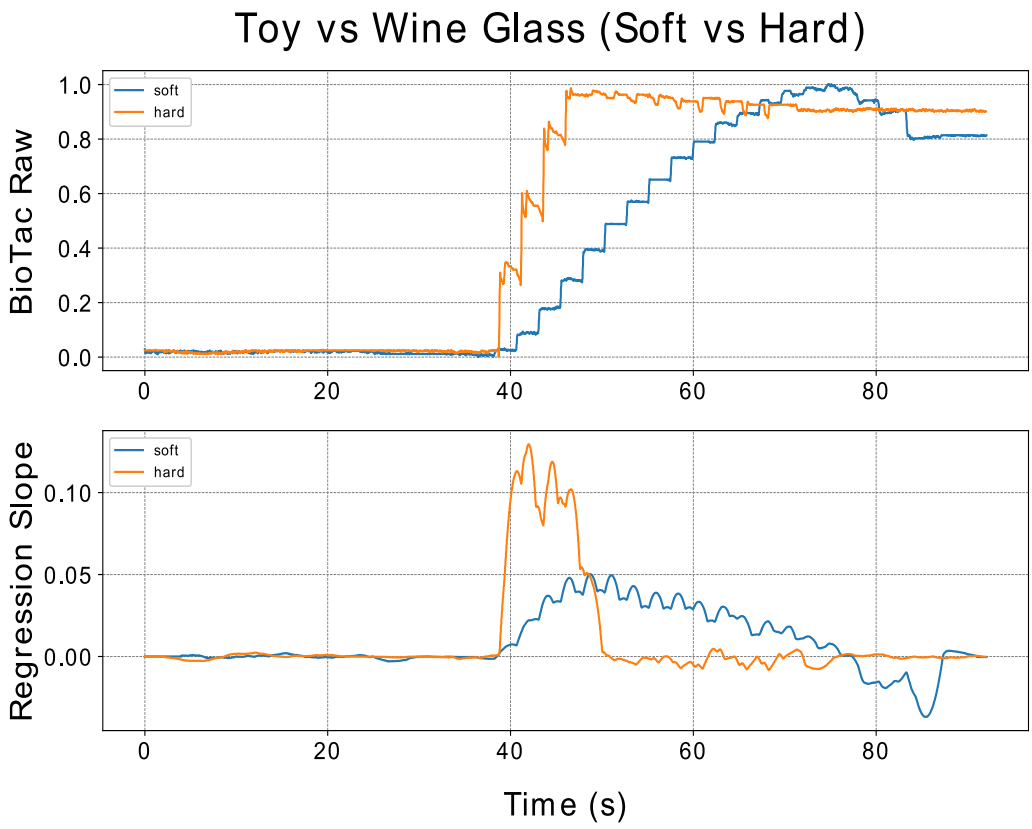


Figure 3.7: Variation of computed slope for soft and hard object. Notice the difference in the rate at which the slope changes.

We use linear regression [107] to obtain a the instantaneous slope over the time-window, and perform the comparison at consecutive intervals, as shown in Fig. 3.7. Consequently, by measuring the relative change in gradient, we are able to judge how fast the object is slipping, and provide larger or smaller control commands as necessary.

3.6 Experimental Results

3.6.1 Experiment Setup

The setup that we use to implement our pipeline includes multiple different robotic and sensing hardware, primarily the UR-10 manipulator and the Shadow Dexterous Hand, equipped with SynTouch BioTac tactile sensors [80]. Our approach utilizes a switching controller architec-

Table 3.1: Success rate over different object classes.

Objects	Success (%)
Bottle	85
Transparent Wine Glass	80
Tuna Can	70
Football	60
Softball	65
Jello Box	70
Electronics Box	85
Apple	75
Tomato	70
Soft Toy	75

ture, where we deploy different strategies for controlling the UR-10 arm and the different joints of the Shadow Hand [108] with feedback between controllers. The underlying control inputs come from various tactile sensors and the joint angles of the Shadow Hand.

We also introduce the `shadowlibs`[1] library, a software toolkit that contains several utility functions for controlling the Shadow Hand.

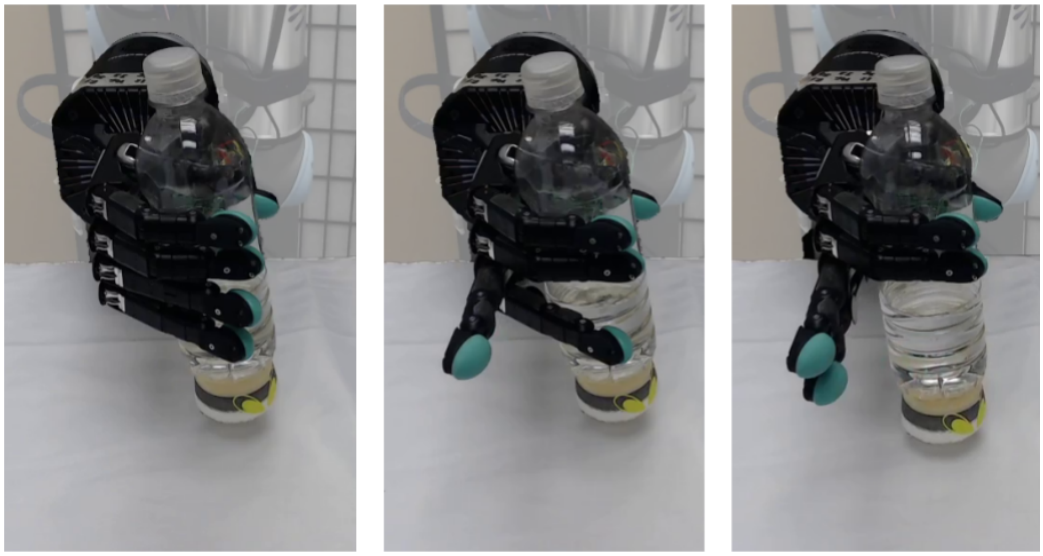
3.6.2 Results

We demonstrate our algorithm on a set of objects with varied shapes and sizes. The upper row of Fig. 3.8a shows the dataset used, and the lower row of Fig. 3.8a shows successful grasps of four classes of objects, namely spherical, non-rigid, cuboidal, and transparent cylindrical respectively. We are able to grasp a wine glass, a soft toy, a ball and a box without any human intervention and without prior knowledge of their shapes, sizes or weights. The criteria for successful grasp were based on the ability to not only grasp the object entirely, but also to lift it and hold it in suspension for 10 seconds.

Table 3.1 summarizes our results for various objects in our dataset.

[1] `kanishkaganguly/shadowlibs`

(a) Top: Dataset of objects used in experiments.
Bottom (L-R): Grasping of Softball, Soft Toy, Box and Wine Glass.



(b) Demonstration of the stable grasp as the number of fingers contacting the object are reduced.

Figure 3.8: Results from Experiments

Table 3.2: Comparison with other state-of-the-art approaches.

Paper	Method	Num. Objects	Success (%)
Li, *et al*. [109]	Simulation	3	53.3
Liu, *et al*. [110]	Simulation	50	66.0
Saxena, *et al*. [111]	Vision	9	87.8
Wu, *et al*. [112]	Reinf. Learning	10	98.0
Pablo, *et al*. [113]	GCN	51	76.6
Ours	Tactile based Control	10	73.5

3.7 Analysis

As can be seen from Fig. 3.7, there is a clear distinction between the BioTac's response to soft versus hard objects. The resulting slopes are also clearly distinguishable in their rate of change. Intuitively, softer objects slip slowly as compared to harder objects. This opens up interesting future avenues for material-adaptive grasping using our approach.

We also demonstrate in Fig. 3.8b, that our controller is able to adjust to changes to the number of fingers in contact with the object on-the-fly. In particular, the object remains in a stable grasp even after two fingers are removed from contact showing the adaptive and robust nature of our approach. Such an approach has built-in recovery from possible failure of finger joints.

Lastly, as can be seen from the results, our proposed method, while simple, is quite adept at zero-shot object grasping. Compared to other methods, as shown in Table 3.2, such as those that use vision or learning, we are able to achieve comparable accuracy. Since our approach only relies on tactile data, we can also robustly grasp transparent objects with relative ease.

3.8 Conclusions

In this work, we develop a simple closed-loop formulation to grasp and manipulate a zero-shot object (object without a prior on shape, size, material or weight) with only tactile feedback. Our approach is based on the concept that we need to compensate for object slip to grasp correctly. We present a novel tactile-only closed-loop feedback controller to compensate for object slip. We experimentally validate our approach in multiple real-world experiments with objects of varied shapes, sizes, textures and weights using a combination of the Shadow Dexterous Hand equipped with BioTac SP tactile sensors. Our approach achieves a success rate of 73.5%. As a parting thought, our approach can be augmented by a zero-shot segmentation method [114] to push the boundaries of learning new objects through interaction.

Chapter 4: GradTac: Spatio-Temporal Gradient Based Tactile Sensing

Tactile sensing for robotics is achieved through a variety of mechanisms, including magnetic, optical-tactile, and conductive fluid. Currently, the fluid-based sensors have struck the right balance of anthropomorphic sizes and shapes and accuracy of tactile response measurement. However, this design is plagued by a low Signal to Noise Ratio (SNR) due to the fluid based sensing mechanism "damping" the measurement values that are hard to model.

To this end, we present a spatio-temporal gradient representation on the data obtained from fluid-based tactile sensors, which is inspired from neuromorphic principles of event based sensing.

We present a novel algorithm (GradTac) that converts discrete data points from spatial tactile sensors into spatio-temporal surfaces and tracks tactile contours across these surfaces. Processing the tactile data using the proposed spatio-temporal domain is robust, makes it less susceptible to the inherent noise from the fluid based sensors, and allows accurate tracking of regions of touch as compared to using the raw data.

We successfully evaluate and demonstrate the efficacy of GradTac on many real-world experiments performed using the Shadow Dexterous Hand, equipped with the BioTac SP sensors. Specifically, we use it for tracking tactile input across the sensor's surface, measuring relative forces, detecting linear and rotational slip, and for edge tracking. We also release an accompany-

ing task-agnostic dataset for the BioTac SP, which we hope will provide a resource to compare and quantify various novel approaches, and motivate further research.

4.1 Introduction

Computational tactile sensing has myriad applications in robotics, especially in tasks related to grasping and manipulation. The robotics community has put a significant amount of effort into the design of hardware and algorithms to equip robots with tactile sensing capabilities that rival that of the human skin. Decades of research have led to the design of fluid based sensing mechanisms as the gold-standard for striking the balance between anthropomorphic shapes, sizes and responses. However, as computational algorithms have utilized such sensors widely, some largely unexplored issues still persist due to their non-linear behavior observed in both spatial and temporal responses due to external factors that are hard to model [115].

Primarily, these sensors have low Signal to Noise Ratios (SNR), owing to the use of a fluid-based transmission of forces from the skin to the sensing electronics which "damps" the values. Secondly, because of the non-uniform distribution of the sensing elements inside the mechanical construction, each sensing element has a different sensing range, and respective biases. These issues have prohibited the development of a standard representation of the data, and processing techniques have been designed engineered for a particular set of tasks rather than being general.

Many approaches have been proposed for interpreting the sensor data, with highly accurate computer models on one end [116,117], and a variety of signal processing techniques [118] on the raw data on the other. Both these approaches are computationally expensive and need extensive hand-crafted calibration procedures for them to be operational.

On the contrary, biological systems calibrate for these environmental factors on-the-fly by processing tactile information as spikes or events, which provides advantages for transmission and processing along with built-in robustness. This ideology inspired neuromorphic engineers to develop sensors and low-power hardware [119], that record and process events, as well as algorithms to compute events [120–122]. Recently event based hardware has become available for the research community. The best known among these is a vision sensor called DVS [119, 123], and another sensor is the event based audio cochlea [124]. Event-based processing has also been introduced to the olfactory domain [125] and for tactile data [126].

We propose a novel intermediate representation computed directly from the raw fluid-based tactile data such as that of the BioTac SP sensor. Instead of accurately simulating the deformations and forces on the sensor, as in [116, 117], we compute robust features from the spatio-temporal changes in the tactile data, which carry essential information about the sensor's deformation and forces at the location of touch. The approach is computationally inexpensive and sufficiently accurate to perform a series of tasks.

The main idea is to compute from a sequence of raw data, the significant changes in data values from individual sensors, which we call *Tactile Events*, and then compute the essential tactile features from these events via a spatial interpolation. Specifically, by temporally accumulating the tactile events we construct surface contours, that can be used as a generic representation for tracking touch across the BioTac SP skin. Our approach handles the challenges mentioned above, *i.e.*, it can account for noise and individual sensor biases.

4.1.1 Problem Formulation and Contribution

The question we tackle in this work can be summarised as *"What representation do we need to handle noisy data from a Fluid Based Tactile Sensor (FBTS)?"*. To answer this question, we draw inspiration from neuromorphic computing and propose a computational model for representing tactile data using spatio-temporal gradients. Our contributions are formally described next.

- We present an intuition for the relationship between the volumetric deformations of the skin and fluid on a fluid based tactile sensor and spatio-temporal gradients. We further discuss why our method can robustly compute the maximal region of deformation.
- We present a computational model to convert raw tactile signals from an FBTS into an interpolated spatio-temporal surface. This is then used to track regions of applied stimulus across the sensor's skin surface which corresponds to the regions of touch.
- We demonstrate the capabilities of our proposed approach on several real-world experiments, including detecting slippage during grasp, detecting relative direction of motion between fingers, and following planar shape contours.
- We release a novel dataset containing the various experiments we perform on the BioTac SP. It can be used to validate not only our method, but also for comparing other tracking algorithms for the BioTac SP. and help push the field forward.

4.1.2 Prior Work

Tactile sensors broadly fall into several broad categories, including but not limited to piezoresistive, piezoelectric, optical, capacitive, and elastoresistive. We further categorize them into two main classes, based on their sensing modality: optical-tactile (*i.e.*indirect) and direct. This categorization is based on whether the sensing element makes direct contact with the surface being touched. The main tasks performed with tactile data found in the literature include: 1) estimation of the contact location and the net force vector, 2) estimation of high-density deformations on the sensor surface, 3) slip detection and classification, and 4) tracking object edges. We next discuss state-of-the-art works on using the various classes of tactile sensors and solving tasks related to those mentioned above.

Studies that perform estimation directly on the sensor data include [127], who present an analytical method to estimate the 3D point of contact and net force acting on the BioTac sensor based on electrode values, where they assume that electrodes measure force in the direction their normals. [90] discuss several methods for force estimation from tactile data, including Locally Weighted Projection Regression and neural network based regression. They also present a signal processing technique for slip detection using the BioTac, comparing their results using an IMU. [128] introduce a method to infer forces from tactile data using a learning-based approach. They implement a 3D voxel grid to maintain spatial relations of the data, and use a convolutional neural network to map forces to tactile signals.

Recently, some studies modeled a mapping between sensor readings and the field of deformations on the whole sensor surface. [116] presented a finite element model for the 19 taxel BioTac sensor and demonstrated the most accurate simulations of the sensor thus far. They relate

forces applied to specific locations to the sensor's skin deformation. They learn using data they collected, the mapping between 3D contact locations and netforce vectors to the 19 taxel readings, and then by combining the FEM simulation and experimental data they extrapoloate a mapping between taxel sensor measurements and skin deformations and vice-versa. In [117] the authors extended this work using variational autoencoder networks to represent both FEM deformations and electrode signals as low-dimensional latent variables, and they performed cross-modal learning over these latent variables. This enhanced the accuracy of the mapping between taxel readings and skin deformations previously obtained. However, they also showed that for unseen indenter shapes these methods poorly generalise in predicting deformation magnitudes and distributions from electrode values.

[129] using a TacTip optical-tactile sensor ([130]) learn via a CNN to perform reliable edge detection, and then use that in a visual servoing control policy for tracking and moving across object contours. In related work by the authors ([131]), they present a Voronoi tesselation based processing pipeline to predict contact location, as well as shear direction and magnitude on the surface of the sensor. This method is novel in that it does not use any classification or regression techniques and is purely analytical in nature.

[132] use the NeuTouch, a novel event-based tactile sensor along with a Visual-Tactile Spiking Neural Network to perform object classification and rotational slip detection. They also perform ablation studies with an event-based visual camera, and compare their spiking neural networks to traditional network architectures like 3D convolutional networks, and Gated Recurrent Units.

We use the prior work described above as a source of motivation for our pipeline, and we attempt to use the validated experiments in them as a proof of concept of our approach. We

perform slip detection experiments as in [90], perform edge tracking using visual servoing as in [129] and compute forces from touch as described by [128].

4.1.3 Organization

The presented work is organized as follows: In Sec. 4.2, we present the motivation for using the BioTac SP sensor for tactile sensing, and how our method is a practical solution to the challenges posed by this particular type of sensor. We describe in detail why the spatio-temporal gradients are an intuitive way for computing features of deformation on the BioTac SP.

Sec. 4.3 discusses our high-level pipeline, and our experimental setup. We then go into detail regarding our algorithm to generate spatio-temporal gradients, *i.e*. events from raw tactile data, and then discuss how we generate contour surfaces from these events. Lastly, in this section we discuss how we use these spatio-temporal surfaces to track touch stimulus across the BioTac SP skin.

In Sec. 4.5 we demonstrate our pipeline on three distinctly different tasks, and discuss their results and outputs. We first show that our contour surfaces are able to accurately track tactile stimulus in motion across the surface of the BioTac SP skin. We then discuss results on experiments involving varying applied forces on the BioTac SP, where we show that our contour surfaces can distinguish between various levels of force. Lastly, we employ our algorithm on a more practical task of slippage detection during grasping, where we detect time of slippage, and distinguish between longitudinal and rotational slippage types.

4.2 Method

4.2.1 Motivation

We consider for our work the BioTac SP tactile sensor, which comes with a unique sensing mechanism as compared to other contemporary tactile sensors. Tactile sensing mechanisms, as they are available commercially today, lie on a spectrum ranging from *accurate sensing capabilities* on one side to *biomimetic form-factors* on the other. Most sensors on this spectrum make trade-offs on form factor to provide high accuracy. The BioTac SP is one particular sensor that strikes a right balance and is in the middle of the range, where we have a physical shape and sensing mechanism very close to the human finger tip, but this comes at the cost of accuracy and fidelity of sensing.

4.2.2 Challenges with Fluid-Conductive Sensors

Unlike optical-tactile, magnetic or capacitive tactile sensors, fluid based tactile sensors use a conductive fluid to transmit electrical impulses from spatially distributed excitation electrodes to a few sensing locations (taxels) distributed over a solid core. The values generated by the taxels are thus primarily dependent on the characteristics of the fluid, specifically its conductivity. The conductivity of a fluid, such as the electrolytic solution present in the BioTac SP sensor is non-linearly related to various external factors. These include, but are not limited to the temperature of the fluid, the humidity of the surroundings, the area and distance between the excitation and sensing electrodes, and the concentration of the conductive fluid. Each of these factors contribute non-linearly ([115]) to the noise of the individual taxels. Furthermore, the noise charac-

teristics of the sensor electronics are also non-linear, which further exacerbates the situation. We also need to consider sources of noise in the electronic implementation of each taxel's sensing mechanism, which include amplification and analog-to-digital conversion circuitry among others.

4.2.3 Modelling Fluid-Conductive Sensors

While it might be feasible to model each of the aforementioned sources of noise independently and in isolation, the combination and interactions between them when considered together in the system makes it an arduous task. There have been several attempts to develop a physical model of the BioTac sensor, the most recent of which is presented in the work by [117]. In this, the authors present a finite element model (FEM) of an ideal BioTac sensor, and provide an accurate simulation of the skin, the sensing core, and the internal fluid. While the FEM approach provides a physically accurate measurement of the deformation of the skin and fluid based on force stimuli, it does not account for the sources of noise described earlier. This is because the model of the sensor electronics is not considered along with the computational challenges of fluid modelling. Currently, to the best of our knowledge, there is no mathematical model between sensor readings and skin deformations, thereby inhibiting research in this area when utilizing raw sensor measurements.

4.2.4 Bio-inspired motivation for logarithmic change

In our work, we draw inspiration from nature regarding how changes over the skin surfaces may be related to location of touch and relative forces. To this end, we break away from the

core robotics ideology that one requires a complex and accurate mathematical model or a very high quality sensor to perform useful tasks. In particular, we are driven by nature's efficient and parsimonious implementations which perform amazingly well with minimal quality sensors and very simple computing.

To build such an efficient data representation for fluid based sensors, we turn to the Weber-Fechner Laws of psychophysics, which state that the perceived stimulus on any of the human senses is related via an exponential function to the actual stimulus. As a result, humans perceive stimuli such as touch, sound or light as the changes in the logarithm between existing values and new ones. It is thus not surprising that the manufacturers of the BioTac SP sensor, who designed it to be as anthropomorphic as possible, also recommend that the best way to process the data from such fluid-based sensors is to use relative changes instead of raw taxel values.

In practice, the two main challenges with the BioTac SP sensor are that a) the different taxels do not have same baseline value, and b) the taxel values exhibit a low signal to noise ratio. By computing only the taxel changes on a logarithmic scale, our values become independent of the baseline and are more robust to noise, thus tackling both aforementioned issues.

4.2.5 Computing events from raw data

One of the primary outputs of our pipeline is to generate "events" from raw tactile data. The concept of an event is inspired from the neuromorphic research community, which essentially is a data point in time that is "fired" only when there exists a change in the stimulus above a specified threshold.

We consider two consecutive packets of taxel data, at times t and $t + \delta$ respectively. Each

of these packets contains the raw taxel values $X^{t}_{k=1...24}$ and $X^{t+\delta}_{k=1...24}$. We then compute the logarithmic change between each of the $j \in 1, 24$ consecutive taxels, and fire an event when the logarithm of the value at a taxel increases or decreases by a threshold value τ. That is, when:

$$|\ln X_j^{t=i+\delta} - \ln X_j^{t=i}| > \tau \tag{4.1}$$

In other words, a positive event is said to be "fired" when

$$\ln X_j^{t=i+\delta} > e^{\tau} \ln X_j^{t=i} \tag{4.2}$$

and a negative event when

$$\ln X_j^{t=i} > e^{\tau} \ln X_j^{t=i+\delta} \tag{4.3}$$

This gives us intermediate taxel values between times t and $t+\delta$, and their respective timestamps for each taxel $j \in [1, 24]$.

We know from the design of the sensor, as well as ideal sensor simulations that the largest change in the values of the taxels correlates with the region of highest tactile stimulus. Also, this change is dependent on the forces already present on the region of touch, and reaches saturation and demonstrates hysteresis in the raw values. Our algorithm accounts for that by non-linearly interpolating the taxel data, on the log scale. The previously obtained taxel events thus give us a temporal gradient over the change in taxel values, caused by the deformation of the skin and fluid because of the applied force stimulus. Intuitively, these intermediate events between two discrete taxel data values signify change in localized volume of the skin and fluid over time due to the applied tactile stimulus.

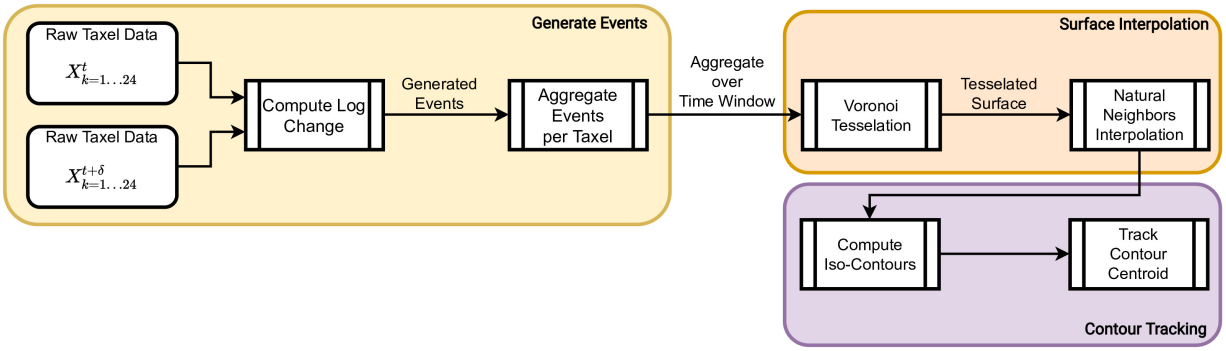


Figure 4.1: High-Level Organization of Our Pipeline

As part of our algorithm, we then process these spatially discrete events for each of the 24 taxel locations and convert them into a continuous, interpolated surface. We use a Voronoi tesselation of the discrete and irregular grid, and perform Natural Neighbors Interpolation to construct an event surface, that gives us an interpolated event value at each point. This surface indirectly depicts the deformation of the skin and fluid, due to applied stimulus. Since the deformation due to applied forces on the BioTac SP is greatest at the region of touch, we generate iso-contours of the event surface, and consider only the maximal valued contour as the region of touch.

4.3 Our Approach

4.3.1 Pipeline

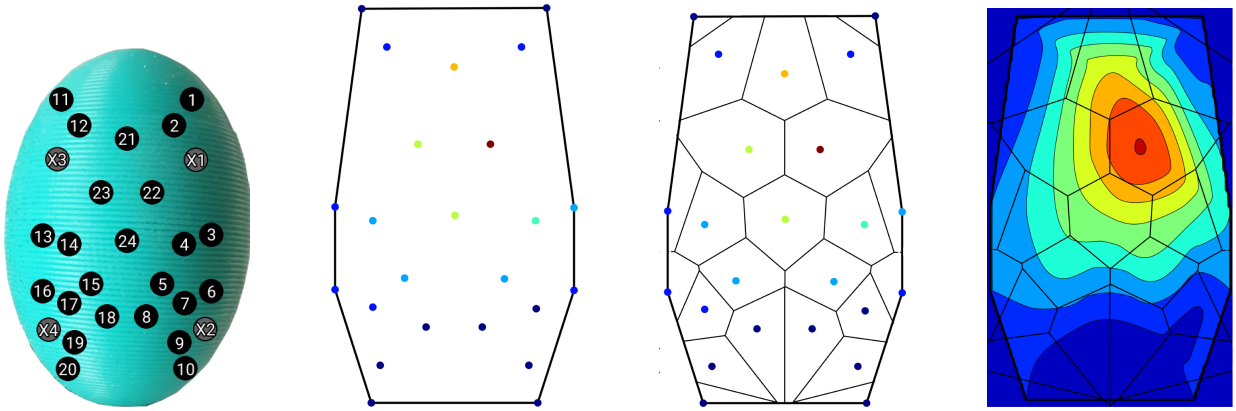
Fig. 4.1 shows an overview of the proposed framework, where we start with 24 points of raw tactile data from the BioTac SP sensors and generate a contact trajectory as output. The pipeline involves converting the raw data into events, aggregating said events by spatial clusters, performing Voronoi Tessellation on the aggregate events, and then using the interpolated values to generate a contour plot whose centroid is tracked over time. We elaborate each of the steps of our pipeline below.

4.3.2 Setup and Methodology

Our hardware setup consists of a UR-10 manipulator equipped with the Shadow Dexterous Hand, which has the BioTac SP sensors attached to each finger tip. The BioTac SP provides a ROS interface to obtain the raw data, at a rate of 100 Hz. This data consists of 24 electrode values which we term "taxels" (tactile element), as well as overall fluid pressure and temperature flux. For our pipeline, we use only the 24 taxel readings. These readings are the result of forces due to contact and the resultant compression of the skin and the enclosed fluid. The nature of our pipeline allows for processing readings from any other tactile sensor, as long as they are spatially distributed across some surface, and timestamps for each data packet are provided. We perform basic min-max normalization and Savitzky-Golay filtering before using the data. Our event-generation algorithm is influenced by principles of event-based sensors, which record logarithmic changes of signal on individual sensing elements, independently and asynchronously.

4.3.3 Generating Events from Raw Tactile Data

In Sec. 4.2.1, we established that our approach does not approximate the entire sensor's surface but only the regions with maximum tactile stimulus. The data from the BioTac SP sensor is obtained at a rate of 100 Hz, or one packet of data every 0.01 seconds. Our method computes the number of events at each taxel, where each event corresponds to the change of some threshold value τ. This essentially decides the granularity of change we are interested in measuring, and more events correspond to larger change, which is correlated to the amount of force that was applied to a particular region. For each event triggered, we also generate a corresponding timestamp between t and $t + \delta$. Taking inspiration from the Weber-Fechner laws of psychophysics



(A) (B) (C) (D)

Figure 4.2: Contour Generation Pipeline. (A) 24 taxel locations, projected onto 2D plane, (B) Initial event aggregates per taxel, (C) Voronoi tesselation of the grid, (D) Contours generated from the interpolated surface

mentioned earlier in Sec. 4.1, we trigger events based on the natural log of the threshold τ. This intuitively means that the frequency of events are higher initially at time t, and gradually taper off as we get closer to the value at time $t + \delta$.

Once the events have been computed for all the raw tactile data points, we aggregate them into temporal frames. The size of the temporal window used for aggregation is an important heuristic that can be fine tuned to favor robustness to noise or allow for a more sensitive tactile response.

4.3.4 Natural Neighbors Based Interpolation

The 24 taxels are located in some 3D space inside the BioTac SP, as per the sensor's design. We project these ellipsoidal locations onto a 2D surface, shown in Fig. 4.2(A), to get an irregular grid of locations on a plane. For each of these 24 2D points, we have the aggregate event counts, as shown in Fig. 4.2(B).

We proceed to perform Voronoi tessellation of this grid, based on the aggregate event values, shown in Fig. 4.2(C). Compared to other methods of interpolation, such as Inverse Distance

Weighting or Gaussian interpolation, Voronoi tessellation provides a more accurate representation of the underlying function we are trying to interpolate. Considering the unstructured nature of our data, i.e. an irregular grid of taxels, traditional methods of interpolation do not take into account the different areas of influence of each taxel when computing the interpolated function. Voronoi tesselation partitions the space proportional to the "strength" of each sample point, by "stealing" some area from the neighboring polygons any time a new point is interpolated [133].

This is mathematically represented by:

$$G(\mathbf{x}) = \sum_{i=1}^{n} w_i(\mathbf{x}) f(\mathbf{x}_i) \tag{4.4}$$

$$w_i(\mathbf{x}) = \frac{A(x_i)}{A(x)} \tag{4.5}$$

where $G(\mathbf{x})$ is the estimate computed at $\mathbf{x}$, and w_i are weights, and $f(\mathbf{x}_i)$ are the known data values at $\mathbf{x}_i$, which are obtained from the 24 event aggregate values. $A(x)$ is the volume of the new cell centered at x, and $A(x_i)$ is the volume of the intersection between the new cell centered in x and the old cell centered in x_i.

Owing to the irregular structure of the sensing locations (taxels) within the BioTac SP, we want to employ a method of interpolation that gives weight to each taxel location proportional to the applied stimulus. Intuitively, Voronoi tessellation partitions the space into irregularly shaped polygons that are proportional to (or representative of) the tactile stimulus exerted on each taxel location. This is better than say, nearest neighbors interpolation which interpolates force values uniformly around each taxel. Although similar to a weighted-average interpolation, Natural Neighbors interpolation weights values by their proportionate area instead of just the raw values at each taxel. This resultant interpolation is a more "truthful" representation of the underlying

surface function than other methods.

The results of the Voronoi tessellation are used to interpolate points on the 2D surface of the BioTac SP, resulting in a continuous surface (Fig. 4.2(D)) whose values correspond to the amount of force on each taxel, and consequently, the deformation of that region of the BioTac SP skin.

4.4 Dataset

There is a lack of standardized datasets in the tactile sensing community, especially when sensors like the BioTac SP are concerned. Most datasets available today are task-specific, or are from optical-tactile sensors. This makes quantitative comparisons difficult for novel algorithms being introduced to the field.

As part of our work, we are releasing an accompanying dataset on tactile motion on the BioTac SP sensor, which is independent of any particular task. The dataset samples include the following:

- Tactile responses from various indenter sizes, applied at different forces
- Motion across the sensor surface in various directional trajectories. We include a) top-to-bottom, b) bottom-to-top, c) left-to-right, d) right-to-left, e) diagonal top-to-bottom, f) diagonal bottom-to-top, g) clockwise and h) counter-clockwise data samples.
- Longitudinal slippage for various objects from a labelled list of objects, as well as the ground-truth timestamps for slip events.
- Rotational slippage for cylindrical object on a constant-speed turntable, as well as the

Figure 4.3: Hardware Setup: Shadow Hand mounted on UR-10 manipulator
ground-truth timestamps for slip events.

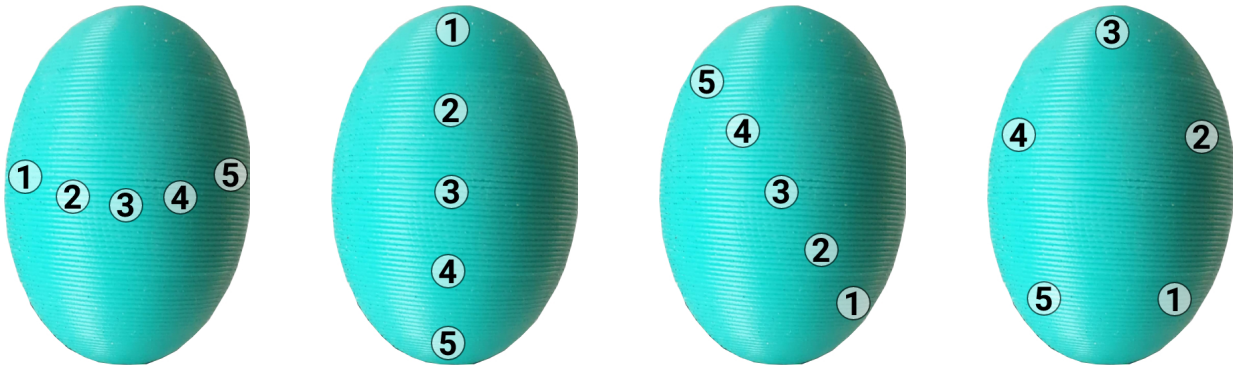
All our data is presented in both NumPy and CSV data formats, and includes all raw 24 taxel values as well as their timestamps. For ease of adoption and use, we eschew the use of ROS Bag format in this dataset, but it may be made available on request.

4.5 Experiments and Results

4.5.1 Experimental Setup

The hardware used to perform all experiments, shown in Fig. 4.3, consists of a UR-10 manipulator equipped with a Shadow Dexterous Hand, with one BioTac SP sensor attached to each of the five finger tips.

Alongside the 24 taxel values from each BioTac SP sensor, the setup also provides us with the 6 DoF pose of the arm and each finger, relative to a world coordinate system at the base of the manipulator. This information is used in the shape tracking experiments.



(A) (B) (C) (D)

Figure 4.4: Touch Tracking Ground Truth Marker Locations. (A) Markers for tracking horizontal trajectory, (B) Markers for tracking vertical trajectory, (C) Markers for tracking diagonal trajectory, and (D) Markers for tracking circular trajectory

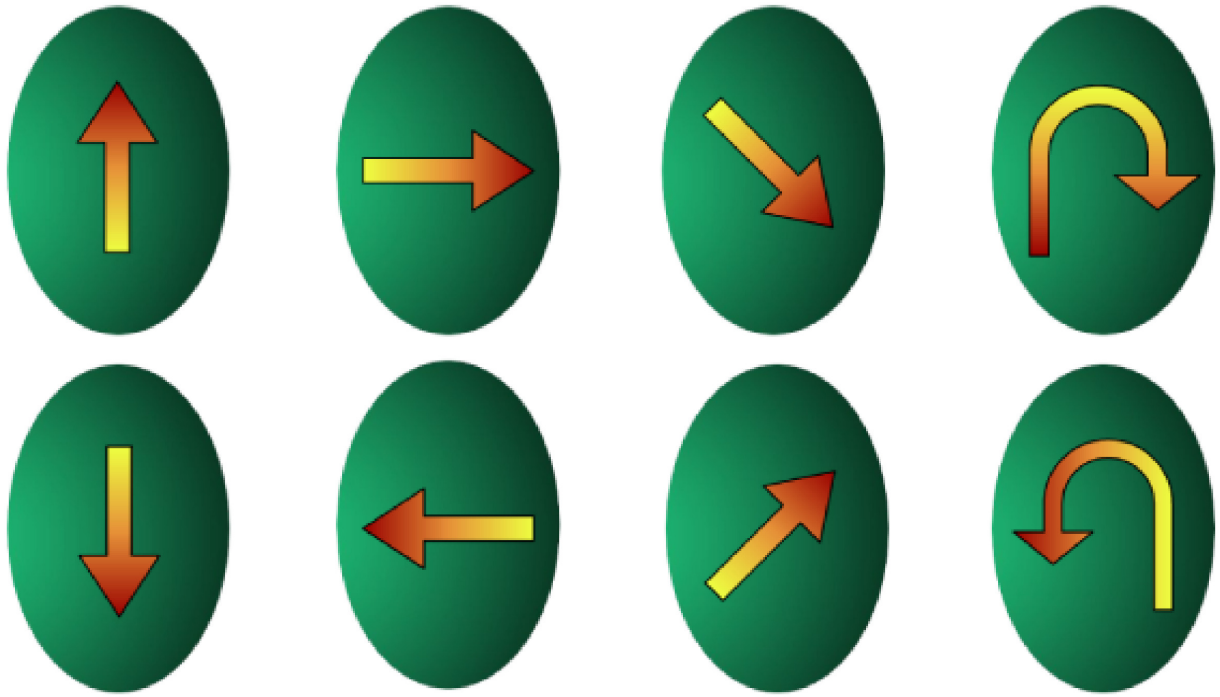
4.5.2 Tracking Location of Touch

We collected data from the BioTac SP at a rate of 100 Hz by making contact at different sensor locations. Three different probes with varying indenter diameters (1, 2 and 5 mm respectively) were used to gather this dataset. The taxel values are time-synchronised with an RGB camera feed which provides us with visual ground truth of contact location at every instance. This data was then used to generate events according to the method described in Sec. 4.3.3.

We evaluate our method of tracking contact by comparing it qualitatively with the ground truth trajectories of the probes obtained from the RGB images. We hand-label several marker locations (shown in Fig. 4.4 on the physical sensor and align them in image coordinate space to the 2D projected locations of the taxels. We used 8 different trajectories, as shown in Fig. 4.5.

We move the indenters on various trajectories along the surface of the BioTac SP, as shown in Fig. 4.5, from top to bottom, bottom to top (Fig. 4.5(A)), left to right, right to left ((Fig. 4.5(B)), diagonally top to bottom, diagonally bottom to top (Fig. 4.5(C)), circular clockwise, and circular counter-clockwise (Fig. 4.5(D)).

Fig. 4.6 shows the outputs from two sample trajectories – diagonal motion from bottom



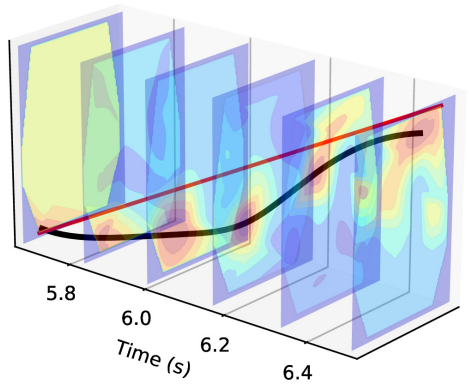
(A) (B) (C) (D)

Figure 4.5: Touch Tracking Trajectories. (A) Up and Down Trajectories, (B) Left and Right Trajectories, (C) Diagonal Trajectories, (D) Circular Trajectories

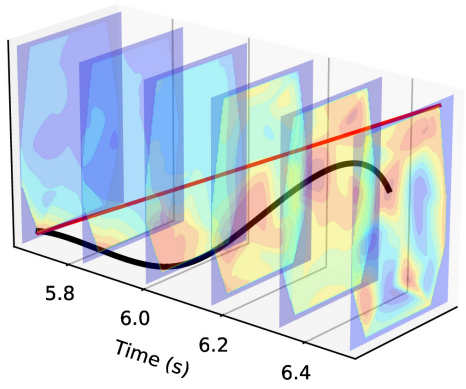
left to top right and counter-clockwise circular motion. Fig. 4.6(A) and Fig. 4.6(C) show the trajectories overlaid on the contour surfaces generated from the event aggregates stacked along the time axis. In both outputs, we can clearly see the event aggregates in red representing the current region of touch. Tracking these across time, we can generate a trajectory of touch across the skin surface.

For comparison, in Figs. 4.6(B) and 4.6(D) we compare the outputs obtained from the filtered, but otherwise unprocessed raw data from the BioTac SP. It is clear that the outputs from our approach, shown in in Fig. 4.6, produces smoother trajectories with reduced noise.

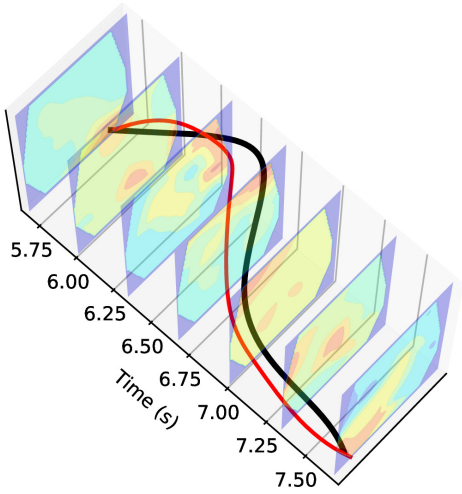
Fig. 4.7 quantifies the median error in touch tracking results for each of the waypoints, for each of the four classes of trajectories (horizontal, vertical, diagonal and circular). Our results are most accurate for the waypoints in the center of the BioTac SP as compared to those near the



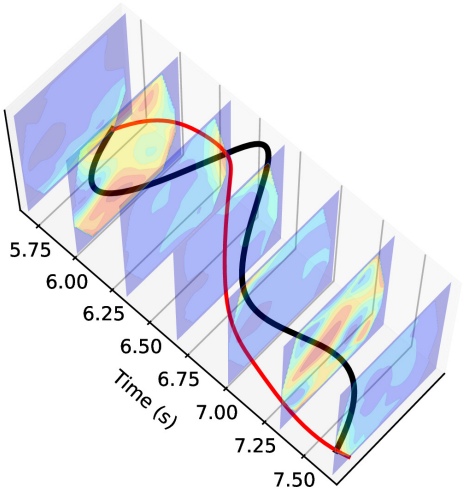
(A)



(B)



(C)



(D)

Figure 4.6: Touch Tracking Trajectory Plots. The red line denotes the ground truth trajectory. (A) Diagonal Trajectory using Events Data, (B) Diagonal Trajectory using Raw Data, (C) Circular Trajectory using Events Data, (D) Circular Trajectory using Raw Data

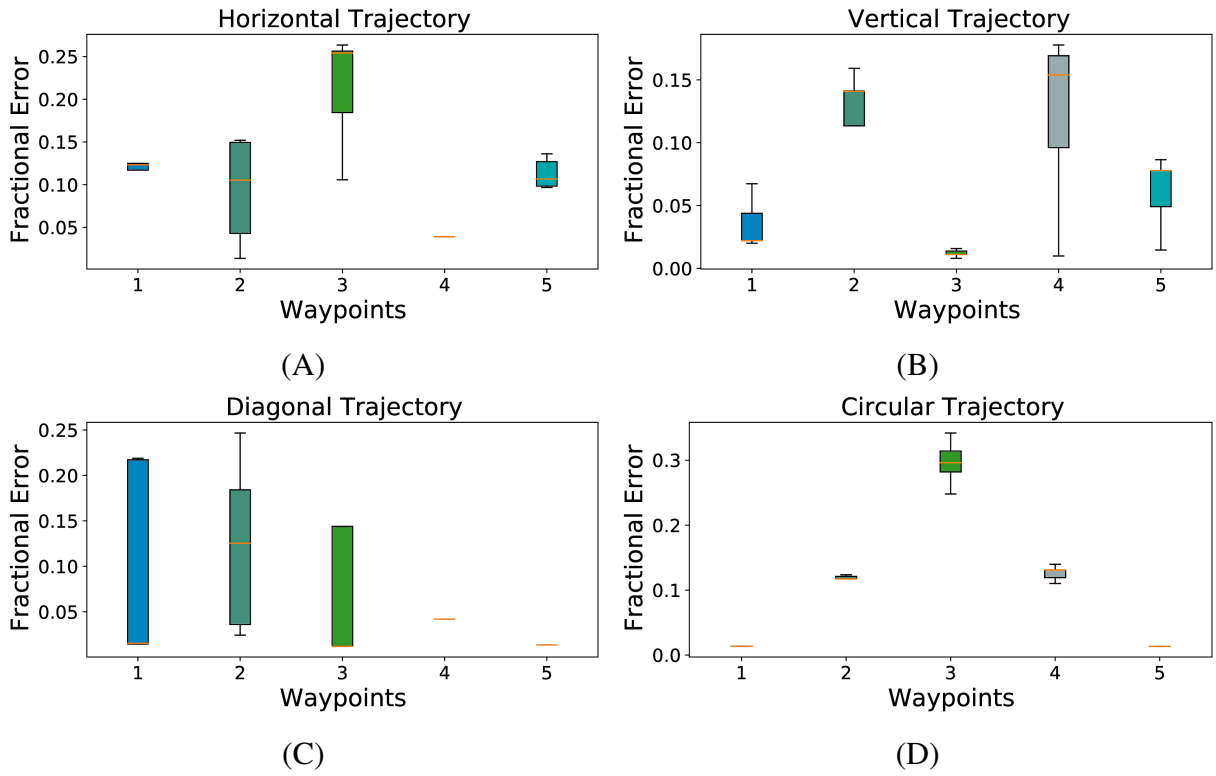


Figure 4.7: Touch Tracking Error Plots. The middle bar represents the median error, the width of each bar is the Interquartile Range, and the fence widths are $1.5 \times$ IQR. (A) Average Deviation for Vertical Trajectory, (B) Average Deviation for Horizontal Trajectory, (C) Average Deviation for Diagonal Trajectory, (D) Average Deviation for Circular Trajectory

		1	2	3	4	5
Circle	*Raw MLP*	0.76	0.47	0.78	0.45	0.75
	Raw Contours	0.07	**0.06**	0.35	0.15	0.31
	Event Contours	**0.04**	0.11	**0.29**	**0.12**	**0.02**
Diagonal	*Raw MLP*	0.86	0.30	**0.01**	0.28	0.75
	Raw Contours	**0.02**	0.24	0.17	0.05	0.02
	Event Contours	0.10	**0.11**	0.05	**0.05**	**0.01**
Horizontal	*Raw MLP*	0.47	0.33	**0.01**	0.30	0.45
	Raw Contours	**0.04**	0.10	0.32	0.40	0.50
	Event Contours	0.13	**0.09**	0.21	**0.04**	**0.01**
Vertical	*Raw MLP*	0.56	0.28	**0.02**	0.22	0.40
	Raw Contours	0.59	0.32	0.17	0.14	0.10
	Event Contours	**0.05**	**0.11**	**0.02**	**0.12**	**0.07**

Table 4.1: Mean errors in the ratio of computed contact location and BioTac SP width at each waypoint over different trajectories. The values in bold denote the best results (least error) for each trajectory.
edges due to the shape and fluid density of the underlying sensor.

Table 4.1 provides a comparison of the average pixel-wise error in tracking the known waypoints (Fig. 4.4), computed with three different methods: First, we take the 24 taxel values corresponding to the timestamp at which the indenter is on each of the known waypoints 1 through 5, and train a fully connected neural network for regression on predicted locations. The average pixel-wise errors are reported under the *Raw MLP* heading. Similarly, we obtain the average pixel-wise errors for each of the five waypoints, using the contours from raw data and from event data. These are reported under the *Raw Contours* and *Event Contours* headers respectively in Table 4.1.

4.5.3 Magnitude of Force

We applied varying forces on the BioTac SP skin using a 2mm indenter to demonstrate the ability of our event contours to measure the correlation between the magnitude of contact force and the area of the maximal contour. The ground truth forces were measured with the help of a calibrated and accurate force sensor.

In order to obtain the relationship between contour areas and applied force, we trained a fully connected neural network with 7 hidden layers, with layer widths of 8, 16, 32, 64, 32, 16 and 8 respectively using L2 loss. This network was used to compute a regression curve mapping the forces to the contour areas. We applied a logistic activation function, and used an inversely scaled learning rate. The network was trained for 5000 epochs, on 200 data points.

As a point of comparison, we applied two other regression methods, a stochastic gradient descent regression with the ElasticNet regularization and log loss, and another L2 regularized regression with Huber loss.

We used the mean absolute percentage error, defined as

$$\mathrm{MAP}(y, \hat{y}) = \mathbb{E}\frac{|y_i - \hat{y}|}{\max(\epsilon, |\mathrm{y_i}|)} \tag{4.6}$$

where $\mathbb{E}$ is the expectation operator.

Each of these methods were trained for 1000 epochs over 200 data points, but for brevity in Fig. 4.8 we only display 100 epochs. The figure also shows the results from the same regression techniques applied to the contours generated from raw data. It is evident from the plots, across all learning algorithms, that the event based data in comparison to the raw data, shows a better

validation loss curve during training, and has an overall lower loss score at testing.

Fig. 4.9 shows a visual representation of the contour regions correlated with the applied forces. As is qualitatively evident, higher forces correspond to larger regions of tactile stimulus, as shown by the highest contour regions in red.

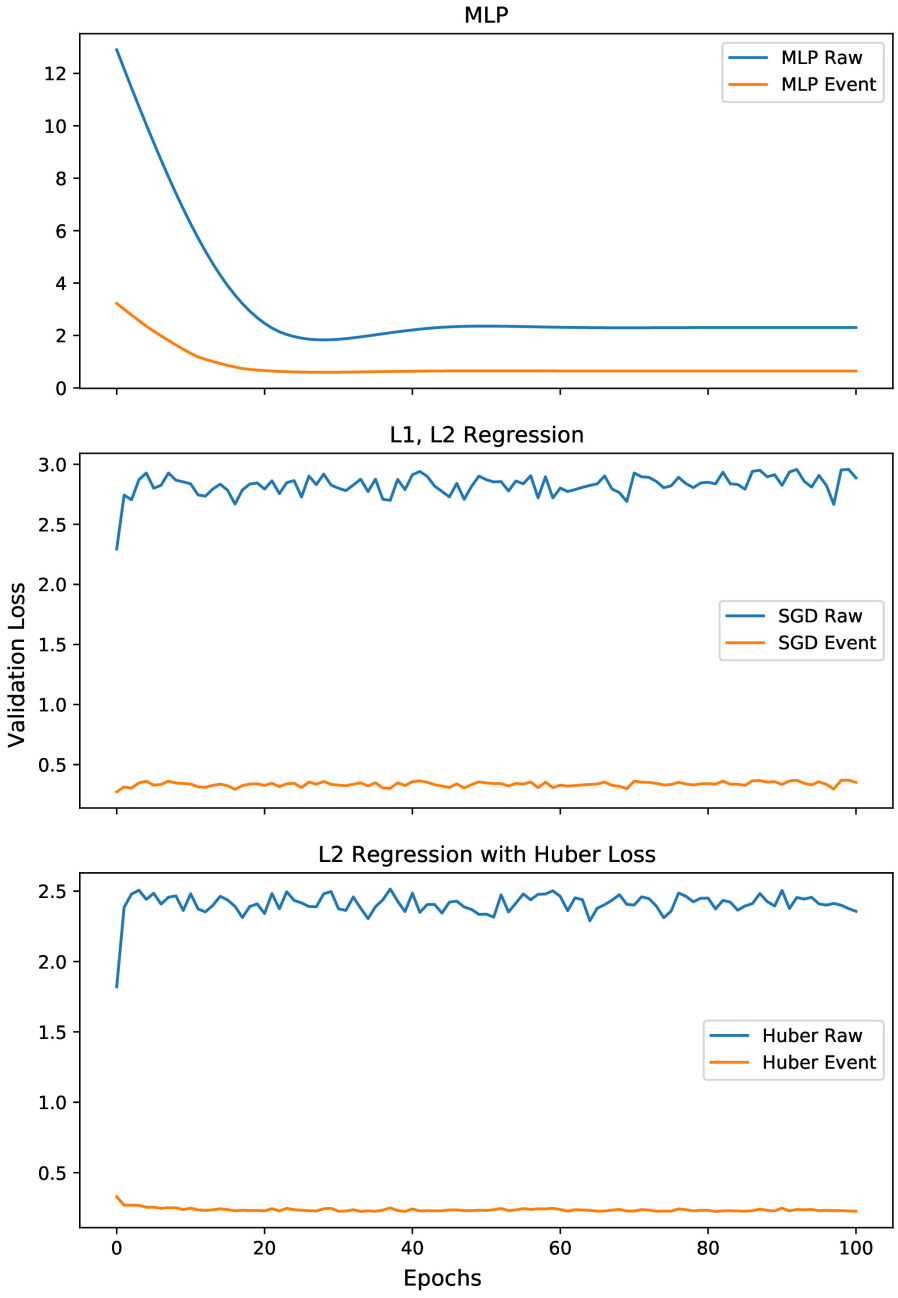
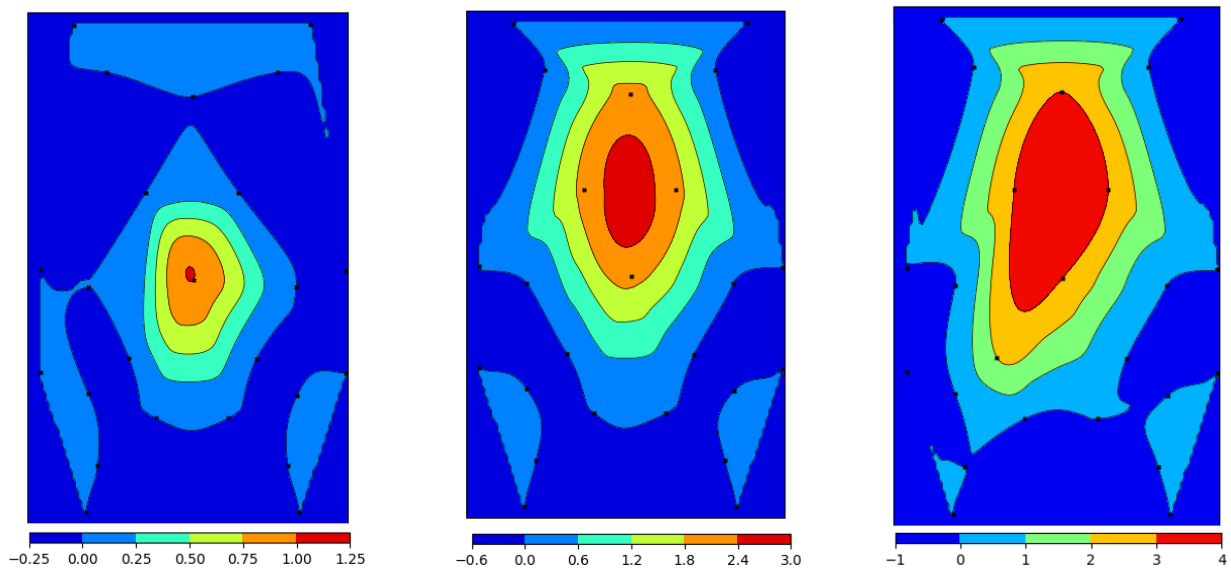


Figure 4.8: Comparing various force-area regression methods for raw vs. event data.



(A) (B) (C)

Figure 4.9: Contour region areas correlated with applied force. (A) 3N applied force, (B) 6N applied force, (C) 12N applied force

4.5.4 Slippage Detection and Classification

(A) (B) (C) (D)

Figure 4.10: Objects Used for Longitudinal and Rotational Slip Detection. (A) Box shape, (B) Spherical shape, (C) Cylinder shape, (D) Tumbler on constant-speed turntable

There are many different ways slippage detection has been achieved using the BioTac SP ([90], [97], [99], [134]), with most methods specifically designed for the task. Here we show that our generic method of spatio-temporal contours can also be used for slippage detection and classification, demonstrating that our approach is very adaptive.

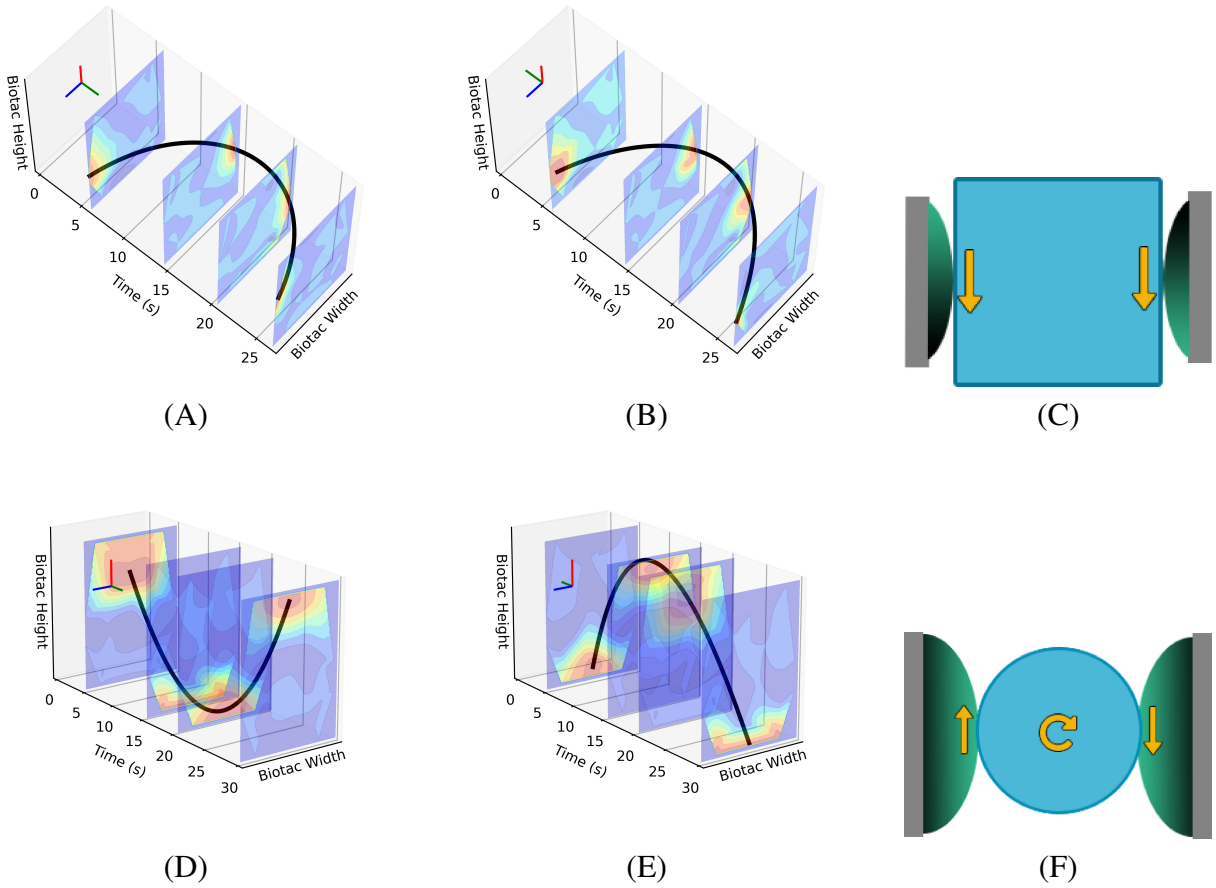


Figure 4.11: Examples of trajectories during longitudinal and rotational slippage. (A), (B) First Finger and Thumb trajectories for longitudinal slippage, (C) Directional Diagram for longitudinal slippage, (D), (E) First Finger and Thumb trajectories for rotational slippage, (F) Directional Diagram for rotational slippage

By tracking the contours spatio-temporally, we are able to detect both the time at which slippage occurs, as well as its directionality. In case of longitudinal slip, i.e. in which the object moves linearly between the fingers, we can measure the direction as "up" or "down". In case of rotational slip, i.e. in which the object rotates between the fingers, we can measure clockwise vs. counter-clockwise rotation.

We do this by comparing the event contours from fingers on opposing sides of the object, while the object is fully grasped by the Shadow Hand, as shown in Figs. 4.10. By tracking and comparing the trajectories generated by the contours on the first finger and the thumb, we can

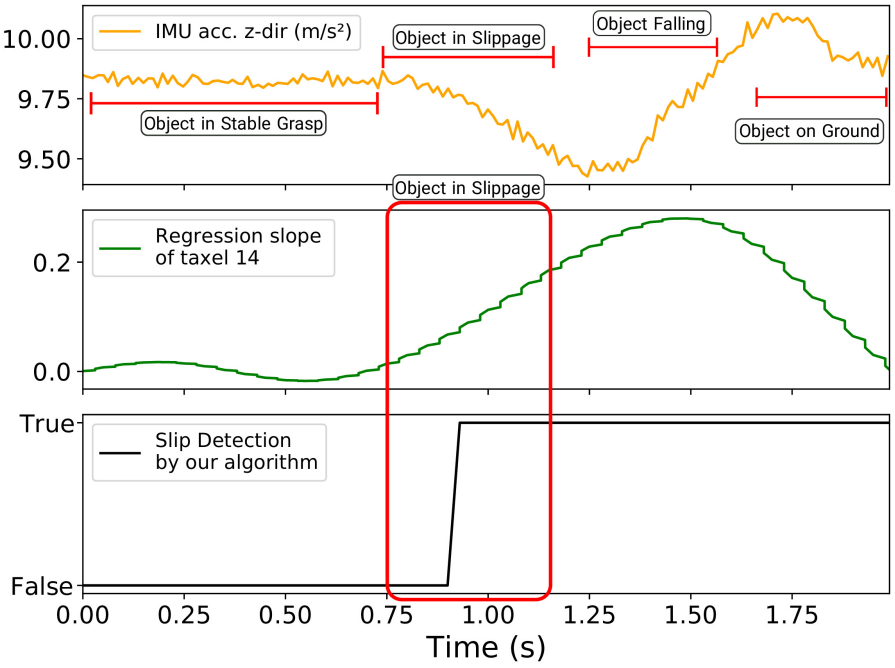


Figure 4.12: Slip Detection Comparison Plots. From top to bottom, we have a low-pass filtered acceleration on the z-axis, the regression slope on the raw data, and the binary slip detection results from event contours.

deduce both the time at which slippage occurs, as well as the direction. In case of longitudinal slippage, as in Fig. 4.11(C), based on the orientation of the BioTac SP sensors with respect to the object, both the contour trajectories have the same direction of motion. In case of rotational slippage, as in Fig. 4.11(F), because of opposing shear forces experienced on the thumb versus the first finger, the contour trajectories have opposing directions of motion.

For the longitudinal slippage scenario, the object is allowed to slide down and is then gradually pulled back up, while maintaining a stable grasp. This can be seen by the contours moving from left to right spatially across the sensor's surface, and then from right back to the left. As is evident from the trajectories, because of shear forces being in the same direction for both sensors, the direction of the respective trajectories are also the same. For the rotational slippage scenario, the object is affixed to a constant-speed turntable and allowed to rotate slowly between the opposing fingers. In this motion, due to the resultant opposing shear forces, the

contour trajectories for the index finger and the thumb have clearly opposite directions.

The event contour outputs of these experiments, longitudinal and rotational slippage, are shown in Figs. 4.11(A),(B) and 4.11(C),(D) respectively. In both cases the contours on the BioTac SP sensor are tracked over time, separately for the index finger and the thumb, which as per the diagrams (Figs. 4.11(A),(B) and 4.11(C),(D)) have different orientations.

We obtain ground truth for our experiments using an 6-axis IMU mounted on each object, and use time synchronized outputs from the IMU to compute the time of slip. We compare our event-based approach to a regression slope computed on the raw data, and the results of one such experiment is shown in Fig. 4.12.

4.5.5 Tracking Edges using Contact Location

As another implementation of our contour tracking pipeline, we demonstrate a simple controller that takes the contour location relative to the UR-10 manipulator, and outputs a motion vector for the finger to follow. The controller is based on a simple tactile servoing algorithm, where we try to maintain the location of the contour location in the center of the surface frame.

As the finger and the attached sensor move over the edge, only one portion of the BioTac SP is in contact with the edge surface. This can be detected and tracked by our controller, and since we start our controller execution with the sensor's center touching the edge, any deviations in the contours from this center is compensated by an opposing motion vector sent to the UR-10 manipulator as a control command.

We track the edges of various non-trivial patterns, namely circle, spiral, triangle, and zig-zag. We overlay the ground truth image of our shapes over the trajectory that is tracked from

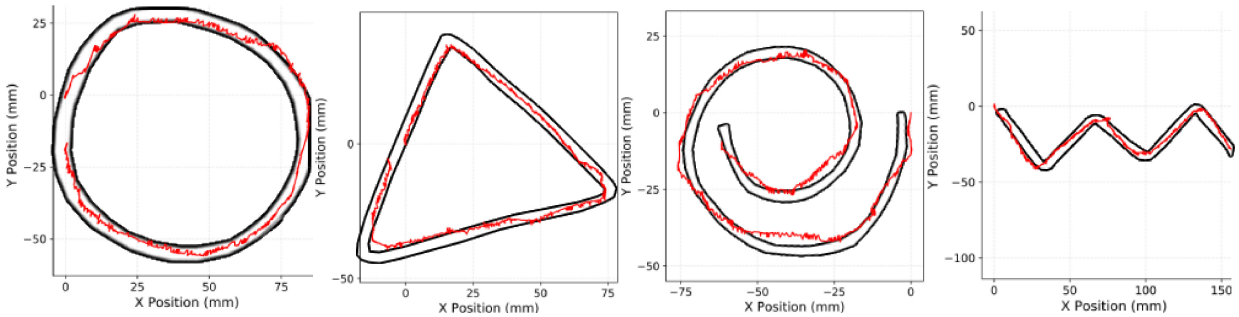


Figure 4.13: Edge Tracking Shapes, and Results. (A) Circular Edge, (B) Triangular Edge, (C) Spiral Edge, (D) Zig-Zag Edge

the robot's pose data for the finger. Barring minor alignment issues between the surface and the finger, and some sliding experienced during the execution, the controller is able to guide the finger across the edges with relative accuracy. The results, with the ground truth shapes, are shown in Figs. 4.13. In each of the plots, we have the trajectory of the BioTac SP in world coordinate space in red, and the black polygons denote the inner and outer diameters of the edges of the shapes we track, also in world coordinate space measured in millimeters. We perform pixel-wise trajectory alignment to align the sensor pose to the ground truth boundary.

4.6 Discussion

In conclusion, the work proposes a novel method to convert raw tactile data from the BioTac SP sensor into a spatio-temporal gradient (events) surface that closely tracks the regions of maximum tactile stimulus. Our algorithm approximates the region of touch on the skin of the BioTac SP sensor sufficiently accurate to perform various tactile feedback tasks. Specifically, we demonstrated the usefulness of the new representation experimentally, for the tasks of tracking tactile stimulus across the sensor, measuring relative force, slippage detection and classification of direction, and tracking edges on a plane. In comparison to other methods for data processing

of fluid-based tactile sensors, our method is real-time and requires minimal overhead in computation. Our approach provides a robust, analytical method for detecting and tracking location of tactile stimulus on the BioTac SP from just 24 data points, improves the signal-to-noise ratio of the raw data and is independent of the baseline taxel values. The benefits of this approach should be even more apparent if hardware based implementations of our algorithm is considered, due to the inherent nature of event-based processing transmitting only changes in tactile stimulus. Lastly, our approach is also independent of any particular sensor type, and we present an accompanying dataset of task-agnostic data samples gathered with the BioTac SP sensor. These include motion tracking over known trajectories and their time-synchronized RGB images, force sensor readings for varying forces applied to the surface using different indenter diameters, and slippage data for several objects and their accompanying ground truth timestamps.

Chapter 5: Conclusion

5.1 Summary

In this thesis, we present novel approaches to grasping and manipulation using tactile feedback from the BioTac SP sensor, on a Shadow Dexterous Hand end-effector setup. We reinforce the crucial role of tactile feedback in achieving successful grasping, and demonstrate how to process tactile information in various ways for control of a high degree-of-freedom end-effector.

We begin in Chapter 1, by presenting a pipeline for observing a human operating appliances, segmenting and detecting contact regions through point cloud processing, and converting them into abstract action parse trees for execution by a robotic agent. This motivates the necessity of contact and separation cues for replicating complex human actions in day-to-day tasks.

We then introduce the concept of *tactile flow*, based on neuroscientific studies where it was found that the parts of the human brain that reacts to visual motion (optical flow) also reacts to motion stimuli across the skin. We take this concept and apply it to the BioTac SP sensor, and perform two- and three-dimensional interpolation on the discrete data points to obtain continuous tactile flow surfaces. We show differences between flow fields under differing surface characteristics.

We next turn to a practical application of tactile flow in Chapter 3, by applying it to a variety of grasping tasks. Using tactile perception, we are able to detect loss of static friction between

the fingers and the object being grasped, and use that as a control signal for adjusting the forces applied. This pipeline allows us to detect and prevent slippage during grasp, which facilitates grasping of previously unseen objects of unknown mass and material types. We demonstrate challenging scenarios such as deformable soft-toys and transparent glasses, which are usually not feasible using visual perception alone.

In Chapter 4, we take a different approach to parsing tactile data from the BioTac SP sensor. Here, we take inspiration from neuromorphic computing techniques, and convert the raw tactile data into spatio-temporal gradients ("simulated events") which are subsequently used to generate an "event" surface. This tactile surface provides us a robust and noise-reduced mid-level feature that we use to track contact surfaces across the sensing skin of the BioTac SP. We demonstrate experimentally the ability to track direction of tactile motion, the correlation between event density and applied forces, and use tactile surfaces to track contoured surfaces.

In addition to these aforementioned contributions, we also introduce in Appendix B.2 the open-source `shadowlibs` library that was developed as a part of the research done in this thesis. It provides a easy to use software interface for the Shadow Dexterous Hand and the BioTac SP sensor through a C++ library, and includes utility functions for controlling the individual joints

of the Shadow Hand, as well as obtain readings from the corresponding sensors.

Chapter 5: Future Work

In future work, we use time-synchronized BioTac SP data from multiple fingers to obtain stable grasps under force closure, and use the perturbation in computed contact wrenches to perform various alignment tasks, such as stacking cups, peg-in-hole style tasks, and shape insertion tasks. Appendix 5 contains the ongoing work, under preparation for publication.

We are also looking into finite element models that closely resemble the internal sensing mechanism of the BioTac SP sensor, including the fluid dynamics and the electrical impulse transfer through the fluid used for sensing. This would better help in characterizing the noise model of the sensor, which thus far has been a non-feasible task. This would help us learn a model of the sensor, which could be used for reinforcement learning tasks for adaptive grasping and manipulation using tactile feedback. Detailed information about the BioTac SP sensor is provided in Appendix A.3.4, and contains initial results from finite element models generated.

Appendix A: StackTac: Object Alignment using Visuo-Tactile Control

A.1 Introduction

Practical applications of dexterous manipulation involve interactions with a variety of objects, and most of these interactions can be classified as some variation on "alignment" tasks. For instance, using a tool such as a screwdriver requires *aligning* the screwdriver with the screw, drilling a hole into a wooden block requires *aligning* the drill with the wall, hammering a nail requires *aligning* the hammer with the nail, and so on and so forth.

For most objects, it is also theorized that their primary affordance lies along their longitudinal axis. This is intuitively true since man-made objects are designed with grasping and manipulation capabilities in mind, which requires them to be shaped with specific geometry in mind. Taking the aforementioned examples of tools, we can see that each of the tools are used along the direction of their longest axis.

The challenge of using robotic end-effectors to align objects lies in having continuous sensory feedback regarding the position of both items being aligned. Contemporary approaches on solving this problem have relied on visual or force sensor inputs for feedback during the alignment process. However, these approaches are generally plagued by occlusion once the parts being aligned are no longer in the line of sight of the vision sensor, or are susceptible to loss of pose state when alignment errors occur.

We propose a novel visuo-tactile feedback control system for the Shadow Dexterous Hand equipped with the BioTac SP sensor that combines the large-scale sensing capabilities of a RGB-D camera with the fine-grained tactile sensing capabilities of the BioTac SP to perform a variety of alignment tasks, including peg-in-hole style operations and stacking cups of different dimensions.

A.2 Prior Work

There have been various approaches to solving the alignment problem, under various scenarios and constraints.

From a neuroscientific standpoint, the authors in [135] discuss the role of the primary and somatosensory cortices *M1* and *S1* respectively, and how they are sensitive to grasping. For unconstrained grasping scenarios, they find that both are responsible for modulating digit forces and positions, and are individually responsible for storing and retrieving learned digit forces and positions. They also reinforce the combination of visual feedback and tactile perception being important to grasping by humans.

An example of an earlier work is in [136], where the authors study the relationship between actions and sensing for orientation tasks. They perform a series of push-align actions and distance measurements to find a set of steps to align objects. As part of this work, they also model shape uncertainty, for convex polygonal shapes.

In [137], the authors present a novel approach for grasping non-planar surfaces using deformable grippers through computation of the entire 6-DoF friction wrench, which greatly improves contact forces as compared to a 3-DoF approximation. They also perform FEM simulations of their models and evaluate their approach against these representative simulations on a

vertical-lifting task.

In [138], the authors tackle a simulated cup-stacking problem in which they present a reactive planning pipeline, where a human may interfere with the ongoing high-level task and the robot is able to react to these changes accordingly under resource constraints. They present a way to convert high-level finite tasks into an abstraction, that can then be solved using game strategies.

In [139], the authors use a deep reinforcement learning approach using a simulated UR-5 robot combined with camera views as input, to perform a peg-in-hole alignment task. They consider a force/torque sensor mounted at the wrist as the sensing modality, and the visual input is used to generate the error metric for calculating reward at each action. They demonstrate results using only tactile feedback, using tactile and visual feedback in separate stages, and by combining them as part of a neural network input. They show that combining visual and tactile perception, and learning jointly demonstrably improves the time taken for alignment completion.

Much of the theoretical motivation presented here has been adapted from *The Fundamentals of Grasping* [140], and *Contact Modeling for Grasping* [141].

A.3 Method

Our alignment pipeline comprises two stages, namely 1. gross alignment multi-view reconstruction and segmentation 2. fine alignment using differential tactile measurements

A.3.1 Visual Alignment

We begin with an Asus XTion RGB-D sensor, that is calibrated to the robot's frame of reference. We use the Open3D processing library to perform all the operations on our visual

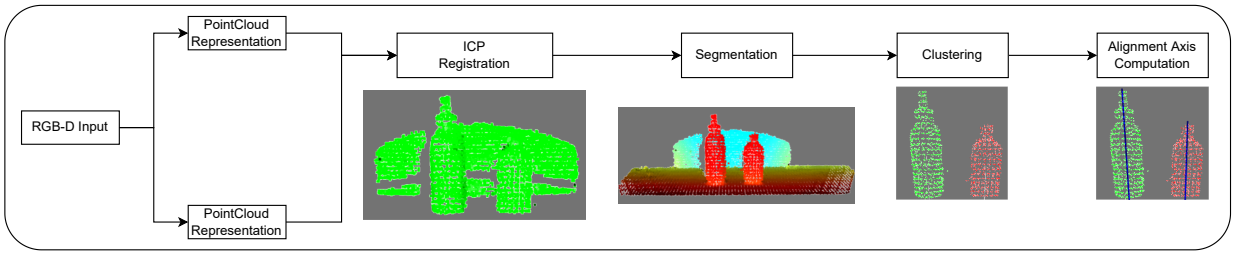


Figure A.1: Visual Alignment Pipeline

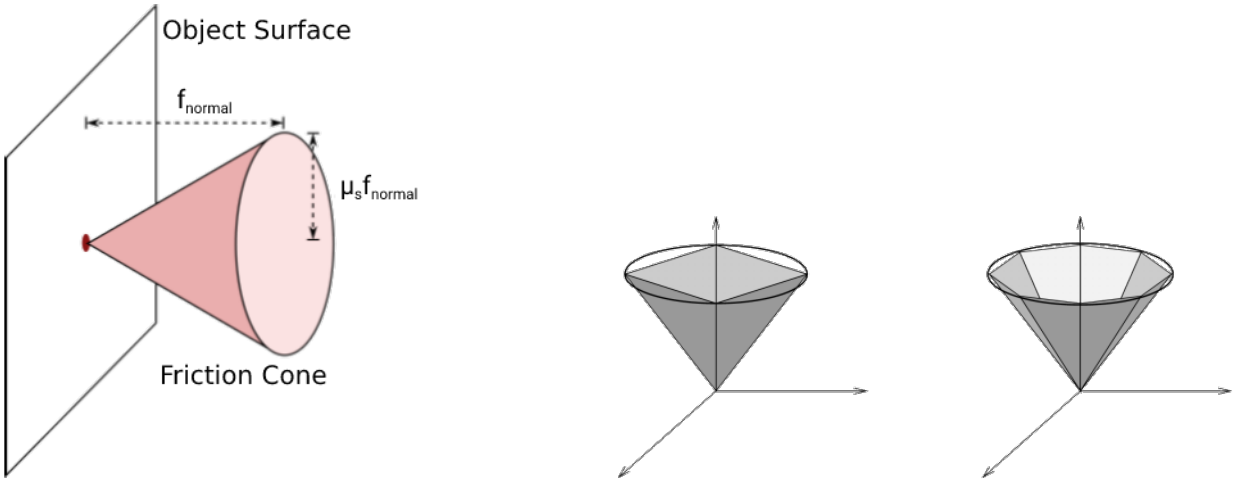
input. This camera is used to capture multiple frames of RGB and registered depth frames, which are then converted to a point cloud format. These point clouds are downsampled to facilitate faster computation, and a point-to-plane Iterative Closest Point algorithm (Eqn. A.1) is applied to each cloud in succession.

$$E = \sum_{i} \|\text{ppd}(p_i, q_i, n_i)\|_2 \rightarrow 0 \tag{A.1}$$

Here, p_i and q_i are the i^{th} points of clouds P and Q respectively, and n_i is the normal to the point q_i. *ppd* is the point-to-plane distance that is being optimized between the corresponding points, and we find the transform T that minimizes the point-to-plane distance. This results in the points of cloud Q being transformed to the coordinate system of the previous cloud P, which combines multiple views of the scene into a reconstructed cloud.

Once we obtain this reconstructed scene point cloud, we segment the table using standard robust algorithms, *i.e.*using least squares fitting under RANSAC. The obtained table points are then removed from the reconstructed cloud. This ideally leaves us with only the points comprising the objects on the table, which are the objects to be aligned.

The next step in the pipeline involves using the *Density-based spatial clustering of applications with noise* (DBSCAN) algorithm, which clusters the points according to local, contiguous density, parameterized by the minimum number of points considered to be a valid cluster, and a



(a) Friction Cone Representation

(b) 4 and 6 Facet Cone Approximations

Figure A.2: Friction Cone Models of Contact

distance metric for neighborhood points. This step gives us n distinct clusters for each object in the scene.

Lastly, we take the points in each clustered object and find the dominant axis vectors along all three dimension. We employ *Principal Component Analysis* (PCA) on the 3D coordinates of each point to find the axis vectors in each dimension. The eigenvector corresponding to the largest eigenvalue from the PCA result is used to compute the major axis of the segmented object, and we use the robot manipulator to then align the corresponding axes.

A.3.2 Tactile Alignment

A.3.2.1 Contact Modelling

Let us introduce the concept of the Coulomb Friction Model, which deals with friction forces between finger tip and object at the region of contact. Here, $f_t, f_n \in \mathbb{R}$ refer to the magnitudes of tangential and normal forces respectively. Coulomb's Law states that slippage

occurs when

$$\begin{aligned} &|f_t| > \mu f_n, \\ &\text{constrained by} \quad |f_t| \leq \mu f_n, f_n > 0 \end{aligned} \tag{A.2}$$

where μ is the coefficient of static friction.

The friction cone FC, illustrated in Subfig. A.2a, is defined by

$$\text{FC} = \left\{ f \in \mathbb{R}^3 : \sqrt{f_1^2 + f_2^2} \leq \mu f_3, f_3 \geq 0 \right\} \tag{A.3}$$

While Eqn. A.3 only quantifies a single-point contact, it is not a realistic model. Instead, we prefer to model a soft-finger contact which allows frictional forces, as well as torques about the normal. This is defined as

$$\text{FC} = \left\{ f \in \mathbb{R}^4 : \sqrt{f_1^2 + f_2^2} \leq \mu f_3, f_3 \geq 0, |f_4| \leq \gamma f_3 \right\} \tag{A.4}$$

where γ is the torsional friction coefficient.

For ease of computation, this friction cone is generally approximated by n number of planar facets, and Eqns. A.3, A.4 can be represented by a positive linear combination of these facet vectors. This is illustrated in Subfig. A.2b.

Each contact point i between object and end-effector applies a wrench on the object. The wrench is defined as a vector that describes the forces $f \in R^3$ and torques $\tau \in R^3$ applied to object, and is given with respect to a fixed frame on the body.

The applied wrench, from Eqn. A.4 can be represented by a 6-facet approximation of the

cone, and given by the following matrix:

$$w = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} f_{c_i}, \quad f_{c_i} \in \mathrm{FC_{c_i}} \tag{A.5}$$

where FC_{c_i} is the friction cone defined by Eqn. A.4.

With the aforementioned definitions of *wrench* and *friction cone*, we can define a *grasp* as the set of all possible wrenches that can be exerted by the contact points between end-effector and object. Any force f_i applied at a point of contact i can be mapped to the corresponding wrench on the object as $G_i f_i$, where G_i is the wrench basis matrix, providing a transformation from contact frame of reference to global object frame of reference. This now allows us to compute the summation of all the k forces applied on the objects as one net wrench w as

$$w_{\text{net}} = \sum_{i=1}^{k} G_i f_i = G \begin{bmatrix} f_1 \\ \vdots \\ f_k \end{bmatrix}, \quad G = [G_1 \dots G_k] \tag{A.6}$$

The combined matrix G is the *Grasp Map*, and along with the Friction Cone, can be used to describe any grasp.

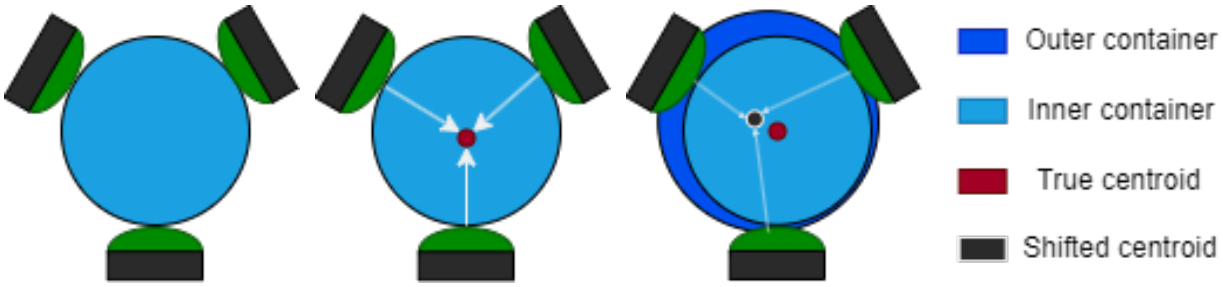
A.3.2.2 Grasp Closure

The generated grasps can be broadly categorized into two classes, namely *Form Closure* and *Force Closure*, and defines conditions under which grasps can be maintained under external disturbances or loads. Form closure requires the object being grasped to be kinematically constrained, i.e. there is no room for relative motion between contact points and object. Force closure relies on forces (typically static friction) between the contact points and the object to resist external wrenches applied. Our system specifically relies on the force closure properties of grasp stability, where for any external wrench w_{ext} applied to the object, there exist contact forces $f_i \in \text{FC}$ such that

$$Gf_i = -w_{\text{ext}} \tag{A.7}$$

A.3.2.3 Wrench Space Perturbation

From Theorem 5.2.3 in [140], for a 3D object with a 6D wrench space, the minimum number of contact points needed for force closure with friction is three. Since we know the pose of each of the BioTac SP sensors in contact with the object, we are able to estimate normal forces and consequently, the wrench space. We assume force closure with three finger contacts, and any perturbation caused due to collision/contact between object being inserted and the cavity being inserted into is transmitted as changes in the readings from the BioTac SP. These differential changes in force measurements can be converted to a vector in the direction opposing the point of collision.



(a) Initial grasp configuration (b) Differential force centroid (c) Shifted centroid due to misalignment

Figure A.3: Tactile Alignment Pipeline

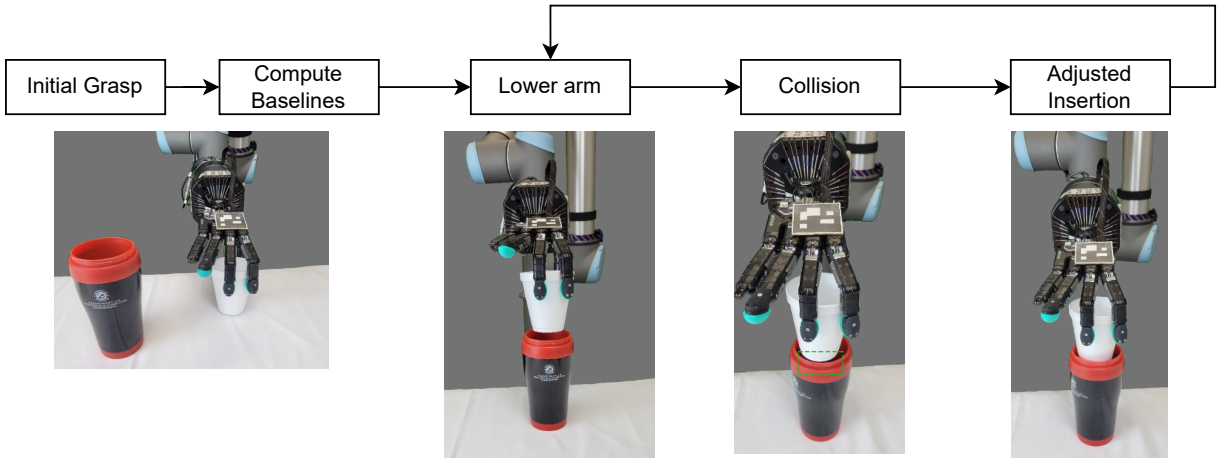


Figure A.4: Tactile Alignment Pipeline

A.3.3 Pipeline

Once the two objects are aligned approximately in space, using the visual alignment pipeline mentioned above in Subsec. A.3.1, we begin the process of refining the alignment using tactile feedback. Our controller assumes three points of contact as necessary and sufficient for a stable grasp, and that these contact points are distributed along the perimeter of the object being aligned. Fig. A.3a shows a simplified diagram of the three BioTac SP sensors arranged around a circular object.

At this stage, we can define the two objects as "outer" and "inner" respectively where the inner object is being aligned with the outer object. For stacking or insertion tasks, it is expected

that the outer object will be necessarily larger than the inner object. The first step requires a baseline calibration of the forces experienced by the tactile sensors without any external contact. Fig. A.3b shows our arrangement. Here, we can consider the normal force vectors from the point of contact to be extending through the object and intersecting at some point "within" the bounds of the object. This condition is necessary for a stable grasp, since if this "balanced centroid" lies outside the bounds, then we have a net torque experienced by the object due to the forces exerted by the end-effectors.

Our pipeline, illustrated in Fig. A.4, after initial grasp is made, then proceeds to lower the arm along a Cartesian trajectory in the $-Z$ (downward) axis. While this motion plan is executed independently of the Hand, the controller receives input based on sensor readings from the BioTac SP. If the visual alignment is not entirely accurate, then at some point in the insertion process, there is a collision experienced between the object being inserted and the boundary walls of the "outer" object. This causes a change in the baseline readings of the three BioTac SP sensors, whose change is proportional to the relative poses of each BioTac SP. By computing the differential change in baselines for each sensor, we are able to generate a normalized vector whose direction is opposing the estimated location of collision. Once this vector is computed, it is scaled appropriately and passed to the UR-10 manipulator as a control target.

As the arm moves the object away from the point of collision, the readings return to baseline and the downward motion can continue as before, until collision is experienced again.

A.3.4 Ongoing Work and Conclusions

The aforementioned work is a proof of concept, and is a work in progress to be sent for review at an upcoming conference. We demonstrate simple alignment tasks, using both the vision and tactile pipelines and show that we are able to detect disturbances under force closure and apply motion compensation to remove these disturbances.

We employ a vision pipeline for gross alignment of the inner object with the cavity of the outer object, using 3D reconstruction and object segmentation to extract the objects to be aligned and find their major alignment axes.

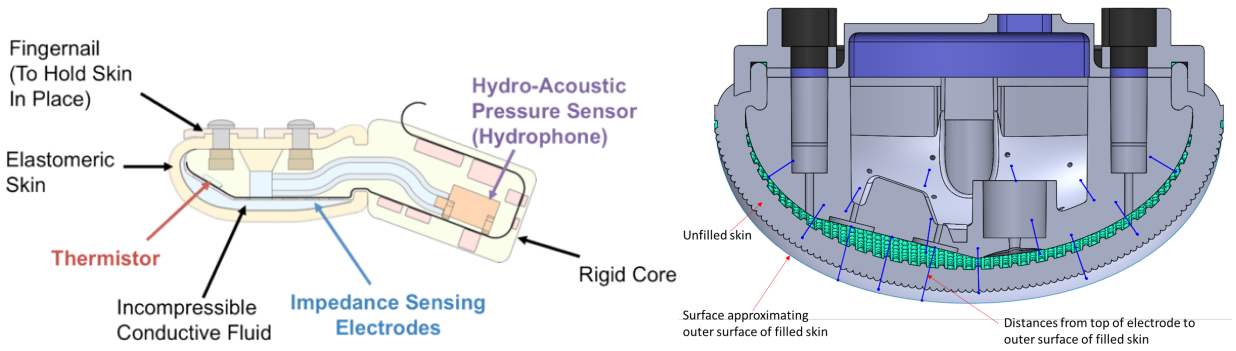
Once grossly aligned, we switch to a tactile feedback control policy which computes a planar (X-Y) vector for the UR-10 manipulator, based on BioTac SP values from all three fingers. The differential measurements allow the system to counteract the effect of collision of the inner object with the outer object in case of a misalignment. The Z-axis motion of the manipulator follows a constant downward velocity.

As part of the final approach, we intend to perform detailed experiments on different sizes of objects, and take into consideration shape alignment as well.

Appendix B: The BioTac SP sensor

The BioTac SP sensor is a fluid-based tactile sensor that conforms closely to the form factor and sensing mechanisms of the human skin. It is unique among its contemporary tactile sensors in that it uses a "direct" sensing modality, i.e. requires contact with the sensing element, as compared to other options such as optical-tactile sensors.

B.1 Internal Structure



(a) Internal structure of the BioTac SP (b) Simulated structure of BioTac SP

Figure B.1: Internal structures of the BioTac SP

The physical construction of the BioTac SP is important to understand its sensing modality. Figs. B.1a,B.1b show a cross-section of the internal structure of the sensor. It consists of a sensing core surrounded by a conductive fluid that is kept confined by a pliable elastomeric "skin" that is attached around the edges to the frame. The sensing core comprises two types of electrodes,

the sensing and the excitation electrodes. These 24 sensing electrodes are distributed across the surface of the core, while the 4 excitation electrodes are spatially separated.

There is also a separate pressure sensor that is used to measure the overall fluid pressure, as well as a thermistor that measures the ambient temperature of the liquid.

B.2 Sensing Mechanism

The four excitation electrodes send out pulses at specified time intervals, which are sensed by the 24 sensing electrodes through a multiplexer circuit. The volume of the column of fluid above each sensing electrode, which is a direct consequence of the deformation of the skin enclosing the fluid, determines the resultant voltage sensed at each electrode location. This is done through a voltage divider circuit mechanism.

Thus, as the skin deforms through contact with external bodies, it results in variations in the sensed voltages, which is then converted from analog to digital signals and processed by the system to which it is connected.

Appendix C: The `shadowlibs` library

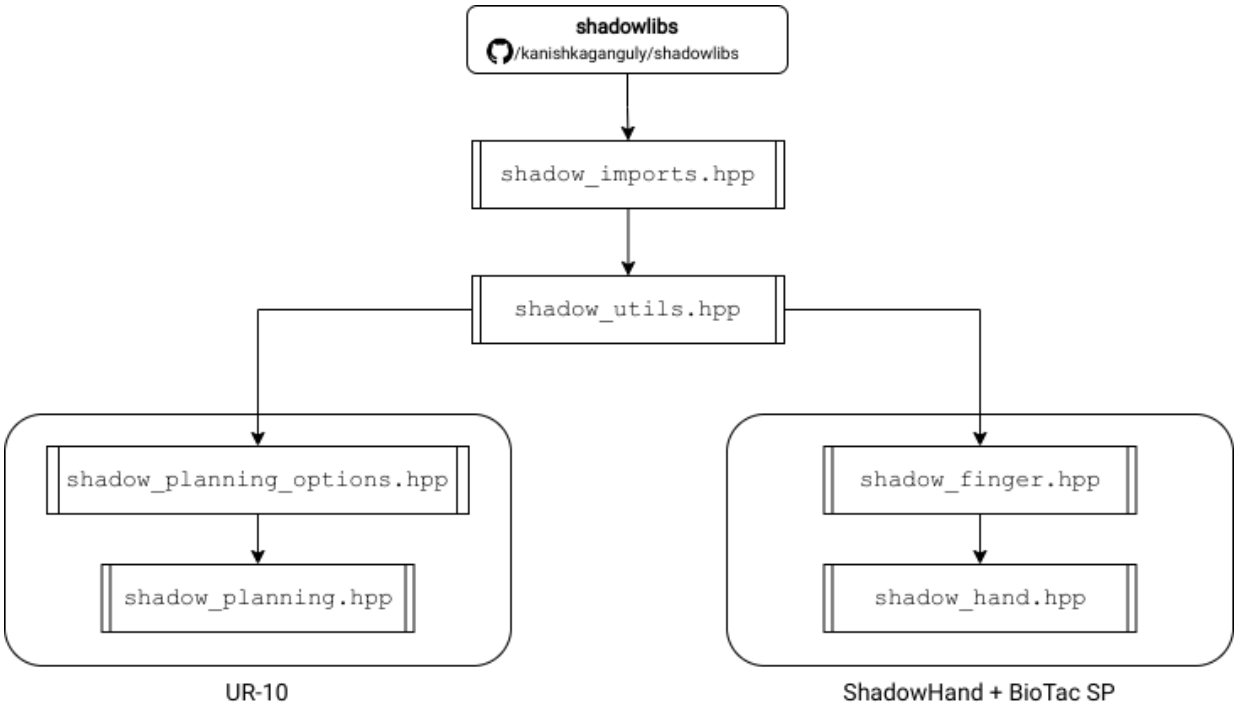


Figure C.1: High-level organization of the library

As part of this disseration, we have developed a utility library for controlling the Shadow Dexterous Hand equipped with the BioTac SP sensors on each finger, which is mounted on the UR-10 manipulator. The primary means of controlling the hardware is through a heavily customized ROS infrastructure, which is running in a Docker container system.

C.1 System Architecture

The overall system is broken down into three main parts, namely **(a)** ROS Host PC, **(b)** Controller PC, **(c)** ODroid XU-4 .

The **ROS Host PC** acts as the main point of entry for the entire system, and is responsible for launching all auxiliary services. The main controller for the UR-10 is launched on this system, as well as remotely starting the external realtime controller for the ShadowHand on the **Controller PC**. The **Controller PC** is a separate Intel NUC system connected to the same local area network, that runs a modified operating system with a realtime kernel. This is done to improve responsiveness and performance when controlling the ShadowHand. Lastly, we also have an **ODroid XU-4** single board computer that is solely responsible for running a minimal server used to receive, parse, and transmit external force sensor data from an Arduino microcontroller to a client on the **ROS Host PC**, which then broadcasts that information over the ROS network for consumption.

C.2 Utilities

We provide a set of simple yet functional utilities as part of `shadowlibs`, used to simplify planning functionality. These functions include various conversion functions for different angular definitions, adding and manipulating objects in the planning scene, obtaining transformations between known coordinate frames, and performing inverse kinematic calculations.

C.3 Manipulator

The UR-10 is controlled through the *MoveIt!* planning framework, using a user-selectable planner algorithm. The choice of planning algorithms for a 7 DoF manipulator are all probabilistic, and the default option is the *BiRRT*. We introduce an object-oriented class for managing planner options, such as the algorithm used, the acceleration and velocity scaling factors, and planner tolerances. This planner options object can be reused for multiple planning phases. We also provide a high-level wrapper over verbose *MoveIt!* functionality for planning tasks on the manipulator. This class accepts a set of planning options, and generates several types of plans as needed including planning to pose or joint targets, as well as planning to pre-defined "named" targets.

C.4 End-Effector

The control system for the ShadowHand is a bit more complicated, despite having an included *MoveIt!* interface. The primary reason is the kinematic structure of the Hand naturally separates it into five separate kinematic chains, one for each finger starting at the wrist joint up to the respective finger tip. This makes planning and execution for the Hand an involved affair. Also, for most of the tasks related to dexterous grasping, it becomes imperative to not have explicit planning routines but requires more "manual" control over joint angles.

To facilitate these requirements, the `shadowlibs` library provides an object-oriented approach to controlling the ShadowHand where each finger can be constructed as a generic `Finger` instance. A combination of all these fingers can then be used to instantiate a `Hand` class. This

interface is more intuitive and conforms to a more natural abstraction for the ShadowHand.

We override the *MoveIt!* library functionality, and disable the onboard trajectory controller to directly apply control commands to the individual joints and this is multithreaded, allowing for simultaneous motion commands for all five fingers. We also integrate each finger's respective BioTac SP sensor as a class object, thus allowing easy implementations of tactile-feedback into custom control schemes.

Bibliography

[1] Ronald Poppe. A survey on vision-based human action recognition. *Image and vision computing*, 28(6):976–990, 2010.

[2] Daniel Weinland, Remi Ronfard, and Edmond Boyer. A survey of vision-based methods for action representation, segmentation and recognition. *Computer vision and image understanding*, 115(2):224–241, 2011.

[3] Ueli Rutishauser, Dirk Walther, Christof Koch, and Pietro Perona. Is bottom-up attention useful for object recognition? In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages Ii–ii. Ieee, 2004.

[4] Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. Multiple object recognition with visual attention. *arXiv preprint arXiv:1412.7755*, 2014.

[5] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 3551–3558. Ieee, 2013.

[6] Katerina Pastra and Yiannis Aloimonos. The minimalist grammar of action. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, Jan 2012.

[7] Abhijit Ogale, Alap Karapurkar, Gutemberg Guerra Filho, and Yiannis Aloimonos. View invariant identification of pose sequences for action recognition. 01 2004.

[8] Yezhou Yang, Anupam Guha, Cornelia Fermüller, Yiannis Aloimonos, and A. V. Williams. A cognitive system for understanding human manipulation actions. 2014.

[9] Yezhou Yang, Cornelia Fermüller, and Yiannis Aloimonos. Detection of manipulation action consequences (mac). In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2563–2570, 2013.

[10] Yi Li, Cornelia Fermuller, Yiannis Aloimonos, and Hui Ji. Learning shift-invariant sparse representation of actions. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2630–2637, 2010.

[11] Ching Teo, Yezhou Yang, Hal Daume, Cornelia Fermüller, and Yiannis Aloimonos. Towards a watson that sees: Language-guided action recognition for robots. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 374–381, 05 2012.

[12] Eadom Dessalene, Chinmaya Devaraj, Michael Maynord, Cornelia Fermuller, and Yiannis Aloimonos. Forecasting action through contact representations from first person video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021.

[13] Yezhou Yang, Yiannis Aloimonos, Cornelia Fermüller, and Eren Erdal Aksoy. Learning the semantics of manipulation action. *CoRR*, abs/1512.01525, 2015.

[14] Somak Aditya, Yezhou Yang, Chitta Baral, Yiannis Aloimonos, and Cornelia Fermüller. Image understanding using vision and reasoning through scene description graph. *Computer Vision and Image Understanding*, 173:33–45, 2018.

[15] Yezhou Yang, Yi Li, Cornelia Fermuller, and Yiannis Aloimonos. Robot learning manipulation action plans by "watching" unconstrained videos from the world wide web. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1), Mar. 2015.

[16] Somak Aditya, Yezhou Yang, Chitta Baral, Cornelia Fermüller, and Yiannis Aloimonos. Visual commonsense for scene understanding using perception, semantic parsing and reasoning. In *AAAI Spring Symposia*, 2015.

[17] Anupam Guha, Yezhou Yang, Cornelia Fermu¨ller, and Yiannis Aloimonos. Minimalist plans for interpreting manipulation actions. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5908–5914, 2013.

[18] Konstantinos Zampogiannis, Yezhou Yang, Cornelia Fermüller, and Yiannis Aloimonos. Object-wise spatial relations and manipulation semantics. *IEEE Intl. Conference on Robotics and Automation*, Mar. 2015.

[19] Yezhou Yang, Cornelia Fermüller, Yi Li, and Yiannis Aloimonos. Grasp type revisited: A modern perspective on a classical feature for vision. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 400–408, 2015.

[20] Dimitrios Lymberopoulos, Abhijit S. Ogale, Andreas Savvides, and Yiannis Aloimonos. A sensory grammar for inferring behaviors in sensor networks. In *Proceedings of the 5th International Conference on Information Processing in Sensor Networks*, IPSN '06, page 251–259, New York, NY, USA, 2006. Association for Computing Machinery.

[21] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014.

[22] Norbert Krüger, Christopher Geib, Justus Piater, Ronald Petrick, Mark Steedman, Florentin Wörgötter, Aleš Ude, Tamim Asfour, Dirk Kraft, Damir Omrčen, et al. Object–action complexes: Grounded abstractions of sensory–motor processes. *Robotics and Autonomous Systems*, 59(10):740–757, 2011.

[23] Karinne Ramirez Amaro, Michael Beetz, and Gordon Cheng. Understanding human activities from observation via semantic reasoning for humanoid robots. In *IROS Workshop on AI and Robotics*, 2014.

[24] Douglas Summers-Stay, Ching L Teo, Yezhou Yang, Cornelia Fermüller, and Yiannis Aloimonos. Using a minimal action grammar for activity understanding in the real world. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 4104–4111. Ieee, 2012.

[25] Yezhou Yang, Anupam Guha, C Fermüller, and Yiannis Aloimonos. A cognitive system for understanding human manipulation actions. *Advances in Cognitive Sysytems*, 3:67–86, 2014.

[26] Yezhou Yang, Yi Li, Cornelia Fermüller, and Yiannis Aloimonos. Robot learning manipulation action plans by" watching" unconstrained videos from the world wide web. In *Aaai*, pages 3686–3693, 2015.

[27] Eren Erdal Aksoy, Alexey Abramov, Johannes Dörr, Kejun Ning, Babette Dellen, and Florentin Wörgötter. Learning the semantics of object–action relations by observation. *The International Journal of Robotics Research*, 30(10):1229–1249, 2011.

[28] Konstantinos Zampogiannis, Yezhou Yang, Cornelia Fermüller, and Yiannis Aloimonos. Learning the spatial semantics of manipulation actions through preposition grounding. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 1389–1396. Ieee, 2015.

[29] Zike Yan and Xuezhi Xiang. Scene flow estimation: A survey. *arXiv preprint arXiv:1612.02590*, 2016.

[30] Evan Herbst, Xiaofeng Ren, and Dieter Fox. Rgb-d flow: Dense 3-d motion estimation using color and depth. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 2276–2282. Ieee, 2013.

[31] Julian Quiroga, Thomas Brox, Frédéric Devernay, and James Crowley. Dense semi-rigid scene flow estimation from rgbd images. In *European Conference on Computer Vision*, pages 567–582. Springer, 2014.

[32] Mariano Jaimez, Mohamed Souiai, Jörg Stückler, Javier Gonzalez-Jimenez, and Daniel Cremers. Motion cooperation: Smooth piece-wise rigid scene flow from rgb-d images. In *3D Vision (3DV), 2015 International Conference on*, pages 64–72. Ieee, 2015.

[33] Mariano Jaimez, Mohamed Souiai, Javier Gonzalez-Jimenez, and Daniel Cremers. A primal-dual framework for real-time dense rgb-d scene flow. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 98–104. Ieee, 2015.

[34] Mariano Jaimez, Christian Kerl, Javier Gonzalez-Jimenez, and Daniel Cremers. Fast odometry and scene flow from rgb-d cameras based on geometric clustering. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 3992–3999. Ieee, 2017.

[35] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4040–4048, 2016.

[36] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, volume 4, page 30, 2007.

[37] Gary KL Tam, Zhi-Quan Cheng, Yu-Kun Lai, Frank C Langbein, Yonghuai Liu, David Marshall, Ralph R Martin, Xian-Fang Sun, and Paul L Rosin. Registration of 3d point clouds and meshes: a survey from rigid to nonrigid. *IEEE transactions on visualization and computer graphics*, 19(7):1199–1217, 2013.

[38] Brian Amberg, Sami Romdhani, and Thomas Vetter. Optimal step nonrigid icp algorithms for surface registration. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. Ieee, 2007.

[39] Richard A Newcombe, Dieter Fox, and Steven M Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 343–352, 2015.

[40] Matthias Innmann, Michael Zollhöfer, Matthias Nießner, Christian Theobalt, and Marc Stamminger. VolumeDeform: Real-time Volumetric Non-rigid Reconstruction. October 2016.

[41] Miroslava Slavcheva, Maximilian Baust, Daniel Cremers, and Slobodan Ilic. Killingfusion: Non-rigid 3d reconstruction without correspondences. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 3, page 7, 2017.

[42] Yezhou Yang, Cornelia Fermuller, Yi Li, and Yiannis Aloimonos. Grasp type revisited: A modern perspective on a classical feature for vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 400–408, 2015.

[43] Qingshuang Chen, He Li, Rana Abu-Zhaya, Amanda Seidl, Fengqing Zhu, and Edward J Delp. Touch event recognition for human interaction. *Electronic Imaging*, 2016(11):1–6, 2016.

[44] Jingyu Yan and Marc Pollefeys. A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate. In *European conference on computer vision*, pages 94–106. Springer, 2006.

[45] Roberto Tron and René Vidal. A benchmark for the comparison of 3-d motion segmentation algorithms. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. Ieee, 2007.

[46] Joao Costeira and Takeo Kanade. A multi-body factorization method for motion analysis. In *Computer Vision, 1995. Proceedings., Fifth International Conference on*, pages 1071–1076. Ieee, 1995.

[47] Ken-ichi Kanatani. Motion segmentation by subspace separation and model selection. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2, pages 586–591. Ieee, 2001.

[48] Shankar Rao, Roberto Tron, Rene Vidal, and Yi Ma. Motion segmentation in the presence of outlying, incomplete, or corrupted trajectories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(10):1832–1845, 2010.

[49] René Vidal and Richard Hartley. Motion segmentation with missing data using power-factorization and gpca. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages Ii–ii. Ieee, 2004.

[50] Anton Mitrokhin, Zhiyuan Hua, Cornelia Fermüller, and Yiannis Aloimonos. Learning visual motion segmentation using event surfaces. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14402–14411, 2020.

[51] Anton Mitrokhin, Chengxi Ye, Cornelia Fermüller, Yiannis Aloimonos, and Tobi Delbruck. Ev-imo: Motion segmentation dataset and learning pipeline for event cameras. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6105–6112, 2019.

[52] Anton Mitrokhin, Cornelia Fermüller, Chethan Parameshwara, and Yiannis Aloimonos. Event-based moving object detection and tracking. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9, 2018.

[53] Chengxi Ye, Anton Mitrokhin, Cornelia Fermüller, James A. Yorke, and Yiannis Aloimonos. Unsupervised learning of dense optical flow, depth and egomotion with event-based sensors. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5831–5838, 2020.

[54] Francisco Barranco, Cornelia Fermüller, and Yiannis Aloimonos. Contour motion estimation for asynchronous event-driven cameras. *Proceedings of the IEEE*, 102(10):1537–1556, 2014.

[55] Dov Katz, Moslem Kazemi, J Andrew Bagnell, and Anthony Stentz. Interactive segmentation, tracking, and kinematic modeling of unknown 3d articulated objects. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 5003–5010. Ieee, 2013.

[56] Evan Herbst, Xiaofeng Ren, and Dieter Fox. Object segmentation from motion with dense feature matching. In *ICRA Workshop on Semantic Perception, Mapping and Exploration*, volume 2, 2012.

[57] Martin Rünz and Lourdes Agapito. Co-fusion: Real-time segmentation, tracking and fusion of multiple objects. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4471–4478, May 2017.

[58] Thomas Whelan, Stefan Leutenegger, R Salas-Moreno, Ben Glocker, and Andrew Davison. Elasticfusion: Dense slam without a pose graph. Robotics: Science and Systems, 2015.

[59] Peter Ochs, Jitendra Malik, and Thomas Brox. Segmentation of moving objects by long term video analysis. *IEEE transactions on pattern analysis and machine intelligence*, 36(6):1187–1200, 2014.

[60] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor Fusion IV: Control Paradigms and Data Structures*, volume 1611, pages 586–607. International Society for Optics and Photonics, 1992.

[61] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, pages 145–152. Ieee, 2001.

[62] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136. Ieee, 2011.

[63] Kok-Lim Low. Linear least-squares optimization for point-to-plane icp surface registration. *Chapel Hill, University of North Carolina*, 4, 2004.

[64] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Cvpr*, 2017.

[65] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. Ieee, 2005.

[66] Michael J Jones and James M Rehg. Statistical color models with application to skin detection. *International Journal of Computer Vision*, 46(1):81–96, 2002.

[67] Vladimir Vezhnevets, Vassili Sazonov, and Alla Andreeva. A survey on pixel-based skin color detection techniques. In *Proc. Graphicon*, volume 3, pages 85–92. Moscow, Russia, 2003.

[68] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[69] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on pattern analysis and machine intelligence*, 13(4):376–380, 1991.

[70] Dave Hershberger, David Gossow, and Josh Faust. rviz. `https://github.com/ros-visualization/rviz`, 2012.

[71] Dave T. Coleman. "moveit!" simple grasps. https://github.com/davetcoleman/moveit_simple_grasps, 2016.

[72] S. Chitta, I. Sucan, and S. Cousins. Moveit! [ros topics]. *IEEE Robotics Automation Magazine*, 19(1):18–19, March 2012.

[73] Steven M. Lavalle. Rapidly-exploring random trees: A new tool for path planning. Technical report, Iowa State University, 1998.

[74] Stefan Schaal. Dynamic movement primitives - a framework for motor control in humans and humanoid robotics, 2002.

[75] Antonio Bicchi, Enzo P. Scilingo, Emiliano Ricciardi, and Pietro Pietrini. Tactile flow explains haptic counterparts of common visual illusions. *Brain Research Bulletin*, 75(6):737–741, 2008. Special Issue: Robotics and Neuroscience.

[76] A. M. De Nunzio, S. Dosen, S. Lemling, M. Markovic, M. A. Schweisfurth, N. Ge, B. Graimann, D. Falla, and D. Farina. Tactile feedback is an effective instrument for the training of grasping with a prosthesis at low- and medium-force levels. *Exp Brain Res*, 235(8):2547–2559, 08 2017.

[77] Antonio Bicchi, Davide Dente, and Enzo Pasquale Scilingo. Haptic illusions induced by tactile flow. 2003.

[78] Laurence R. Harris, Kenzo Sakurai, and William H. A. Beaudot. Tactile flow overrides other cues to self motion. *Scientific Reports*, 7(1), 2017.

[79] Emiliano Ricciardi, Nicola Vanello, Davide Dente, Nicola Sgambelluri, Enzo Scilingo, L Sani, Vincenzo Positano, Mario Guazzelli, James V. Haxby, Luigi Landini, and Antonio Bicchi. Perception of visual and tactile flow activates common cortical areas in the human brain. *Proceedings of the EuroHaptics 2004*, 01 2004.

[80] https://www.syntouchinc.com/en/sensor-technology/.

[81] Moscatelli A. Bianchi, M. *Human and Robot Hands*, chapter 7.3. Springer, 2016.

[82] C. King, M. O. Culjat, M. L. Franco, C. E. Lewis, E. P. Dutson, W. S. Grundfest, and J. W. Bisley. Tactile feedback induces reduced grasping force in robot-assisted surgery. *IEEE Transactions on Haptics*, 2(2):103–110, April 2009.

[83] Alberto Garcia-Garcia, Brayan Stiven Zapata-Impata, Sergio Orts-Escolano, Pablo Gil, and Jose Garcia-Rodriguez. Tactilegcn: A graph convolutional network for predicting grasp stability with tactile sensors. *arXiv preprint arXiv:1901.06181*, 2019.

[84] Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.

[85] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Scandinavian conference on Image analysis*, pages 363–370. Springer, 2003.

[86] Tully Foote. tf: The transform library. In *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*, Open-Source Software workshop, pages 1–6, April 2013.

[87] Ivan Camponogara and Robert Volcic. Grasping movements toward seen and handheld objects. *Scientific Reports*, 9(1):3665, Mar 2019.

[88] Rossella Breveglieri, Annalisa Bosco, Claudio Galletti, Lauretta Passarelli, and Patrizia Fattori. Neural activity in the medial parietal area v6a while grasping with or without visual feedback. *Scientific Reports*, 6(1):28893, Jul 2016.

[89] Rachel K. Clifton, Darwin W. Muir, Daniel H. Ashmead, and Marsha G. Clarkson. Is visually guided reaching in early infancy a myth? *Child Development*, 64(4):1099–1110, 1993.

[90] Zhe Su, Karol Hausman, Yevgen Chebotar, Artem Molchanov, Gerald E Loeb, Gaurav S Sukhatme, and Stefan Schaal. Force estimation and slip detection/classification for grip control using a biomimetic tactile sensor. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 297–303. Ieee, 2015.

[91] Yevgen Chebotar, Karol Hausman, Zhe Su, Artem Molchanov, Oliver Kroemer, Gaurav Sukhatme, and Stefan Schaal. Bigs: Biotac grasp stability dataset. 05 2016.

[92] Brayan S. Zapata-Impata, Pablo Gil, and Fernando Torres Medina. Non-matrix tactile sensors: How can be exploited their local connectivity for predicting grasp stability? *CoRR*, abs/1809.05551, 2018.

[93] Nicholas Pestell, Luke Cramphorn, Fotios Papadopoulos, and Nathan F Lepora. A sense of touch for the shadow modular grasper. *IEEE Robotics and Automation Letters*, 4(2):2220–2226, 2019.

[94] B. Belousov, A. Sadybakasov, B. Wibranek, F. Veiga, O. Tessmann, and J. Peters. Building a library of tactile skills based on fingervision. In *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, pages 717–722, 2019.

[95] Akihiko Yamaguchi. Fingervision. `http://akihikoy.net/p/fv.html`, 2017.

[96] Mike Lambeta, Po-Wei Chou, Stephen Tian, Brian Yang, Benjamin Maloon, Victoria Rose Most, Dave Stroud, Raymond Santos, Ahmad Byagowi, Gregg Kammerer, et al. Digit: A novel design for a low-cost compact high-resolution tactile sensor with application to in-hand manipulation. *IEEE Robotics and Automation Letters*, 5(3):3838–3845, 2020.

[97] Filipe Veiga, Benoni Edin, and Jan Peters. Grip stabilization through independent finger tactile feedback control. *Sensors (Basel, Switzerland)*, 20(6):1748, Mar 2020.

[98] Yuzuko C Nakamura, Daniel M Troniak, Alberto Rodriguez, Matthew T Mason, and Nancy S Pollard. The complexities of grasping in the wild. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pages 233–240. Ieee, 2017.

[99] Roberto Calandra, Andrew Owens, Dinesh Jayaraman, Justin Lin, Wenzhen Yuan, Jitendra Malik, Edward H Adelson, and Sergey Levine. More than a feeling: Learning to grasp and regrasp using vision and touch. *IEEE Robotics and Automation Letters*, 3(4):3300–3307, 2018.

[100] Ankur Handa, Karl Van Wyk, Wei Yang, Jacky Liang, Yu-Wei Chao, Qian Wan, Stan Birchfield, Nathan Ratliff, and Dieter Fox. Dexpilot: Vision-based teleoperation of dexterous robotic hand-arm system. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9164–9170. Ieee, 2020. Teleoperation.

[101] OpenAI, Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Józefowicz, Bob McGrew, Jakub W. Pachocki, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning dexterous in-hand manipulation. *CoRR*, abs/1808.00177, 2018.

[102] OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving rubik's cube with a robot hand. *CoRR*, abs/1910.07113, 2019.

[103] Sergey Levine, Peter Pastor, Alex Krizhevsky, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection, 2016.

[104] Henry Zhu, Abhishek Gupta, Aravind Rajeswaran, Sergey Levine, and Vikash Kumar. Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3651–3657. Ieee, 2019.

[105] Ilija Radosavovic, Xiaolong Wang, Lerrel Pinto, and Jitendra Malik. State-only imitation learning for dexterous manipulation. *arXiv preprint arXiv:2004.04650*, 2020.

[106] Yueyue Liu, Zhijun Li, Huaping Liu, and Zhen Kan. Skill transfer learning for autonomous robots and human-robot cooperation: A survey. *Robotics and Autonomous Systems*, page 103515, 2020.

[107] Jeffrey M. Stanton. Galton, pearson, and the peas: A brief history of linear regression for statistics instructors. *Journal of Statistics Education*, 9(3):null, 2001.

[108] Paul Tuffield and Hugo Elias. The shadow robot mimics human actions. *Industrial Robot: An International Journal*, 2003.

[109] Miao Li, Kaiyu Hang, Danica Kragic, and Aude Billard. Dexterous grasping under shape uncertainty. *Robotics and Autonomous Systems*, 75:352–364, 2016.

[110] Min Liu, Zherong Pan, Kai Xu, Kanishka Ganguly, and Dinesh Manocha. Deep differentiable grasp planner for high-dof grippers, 2020.

[111] Ashutosh Saxena, Lawson Wong, Morgan Quigley, and Andrew Y. Ng. A vision-based system for grasping novel objects in cluttered environments. In Makoto Kaneko and Yoshihiko Nakamura, editors, *Robotics Research*, pages 337–348, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[112] Bohan Wu, Iretiayo Akinola, Jacob Varley, and Peter Allen. Mat: Multi-fingered adaptive tactile grasping via deep reinforcement learning, 2019.

[113] Pablo Gil Zapata-Impata and Fernando Torres. Tactile-driven grasp stability and slip prediction. *Robotics*, 2019.

[114] Chahat Deep Singh, Nitin J. Sanket, Chethan M. Parameshwara, Cornelia Fermuller, and Yiannis Aloimonos. NudgeSeg: Zero-shot object segmentation physical interaction. *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[115] Nicholas Wettels, Veronica J Santos, Roland S Johansson, and Gerald E Loeb. Biomimetic tactile sensor array. *Advanced Robotics*, 22(8):829–849, 2008.

[116] Yashraj S Narang, Balakumar Sundaralingam, Karl Van Wyk, Arsalan Mousavian, and Dieter Fox. Interpreting and predicting tactile signals for the syntouch biotac. *arXiv preprint arXiv:2101.05452*, 2021.

[117] Yashraj Narang, Balakumar Sundaralingam, Miles Macklin, Arsalan Mousavian, and Dieter Fox. Sim-to-real for robotic tactile sensing via physics-based simulation and learned latent projections. *arXiv preprint arXiv:2103.16747*, 2021.

[118] Nicholas Wettels and Gerald E. Loeb. Haptic feature extraction from a biomimetic tactile sensor: Force, contact location and curvature. *2011 IEEE International Conference on Robotics and Biomimetics*, pages 2471–2478, 2011.

[119] Christian Brandli, Raphael Berner, Minhao Yang, Shih-Chii Liu, and Tobi Delbruck. A 240$\times$ 180 130 db 3 μs latency global shutter spatiotemporal vision sensor. *IEEE Journal of Solid-State Circuits*, 49(10):2333–2341, 2014.

[120] Anton Mitrokhin, Cornelia Fermüller, Chethan Parameshwara, and Yiannis Aloimonos. Event-based moving object detection and tracking. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9. Ieee, 2018.

[121] Guillermo Gallego, Tobi Delbrück, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew J Davison, Jörg Conradt, Kostas Daniilidis, et al. Event-based vision: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(1):154–180, 2020.

[122] Nitin J Sanket, Chethan M Parameshwara, Chahat Deep Singh, Ashwin V Kuruttukulam, Cornelia Fermüller, Davide Scaramuzza, and Yiannis Aloimonos. Evdodgenet: Deep dynamic obstacle dodging with event cameras. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10651–10657. Ieee, 2020.

[123] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. A 128×128 120db 15μs latency asynchronous temporal contrast vision sensor. *IEEE journal of solid-state circuits*, 43(2):566–576, 2008.

[124] Minhao Yang, Chen-Han Chien, Tobi Delbruck, and Shih-Chii Liu. A 0.5v 55μw 64×2-channel binaural silicon cochlea for event-driven stereo-audio sensing. *IEEE Journal of Solid-State Circuits*, 51(11):2554–2569, 2016.

[125] Ya-Qi Jing, Qing-Hao Meng, Pei-Feng Qi, Ming Zeng, and Ying-Jie Liu. Signal processing inspired from the olfactory bulb for electronic noses. *Measurement Science and Technology*, 28(1):015105, 2016.

[126] Ella Janotte, Michele Mastella, E Chicca, and Chiara Bartolozzi. Touch in robots: A neuromorphic approach. *ERCIM News*, (125):34–51, 2021.

[127] Chia-hsien Lin, Jeremy Fishel, and Gerald E Loeb. Estimating Point of Contact , Force and Torque in a Biomimetic Tactile Sensor with Deformable Skin. page 6, 2013.

[128] Balakumar Sundaralingam, Alexander Lambert, Ankur Handa, Byron Boots, Tucker Hermans, Stan Birchfield, Nathan Ratliff, and Dieter Fox. Robust learning of tactile force estimation through robot interaction. 2019.

[129] Nathan F. Lepora, Alex Church, Conrad De Kerckhove, Raia Hadsell, and John Lloyd. From pixels to percepts: Highly robust edge perception and contour following using deep learning and an optical biomimetic tactile sensor. *IEEE Robotics and Automation Letters*, 4(2):2101–2107, April 2019.

[130] Benjamin Ward-Cherrier, Nicholas Pestell, Luke Cramphorn, Benjamin Winstone, Maria Elena Giannaccini, Jonathan Rossiter, and Nathan F. Lepora. The tactip family: Soft optical tactile sensors with 3d-printed biomimetic morphologies. *Soft Robotics*, 5(2):216–227, 2018. Pmid: 29297773.

[131] Luke Cramphorn, John Lloyd, and Nathan F. Lepora. Voronoi features for tactile sensing: Direct inference of pressure, shear, and contact locations. *CoRR*, abs/1805.07868, 2018.

[132] Tasbolat Taunyazoz, Weicong Sng, Hian Hian See, Brian Lim, Jethro Kuan, Abdul Fatir Ansari, Benjamin Tee, and Harold Soh. Event-driven visual-tactile sensing and learning for robots. In *Proceedings of Robotics: Science and Systems*, July 2020.

[133] G.W. Lucas. A fast and accurate algorithm for natural neighbor interpolation. 2021. `https://gwlucastrig.github.io/TinfourDocs/NaturalNeighborTinfourAlgorithm/index.html`.

[134] Fariborz Baghaei Naeini, Aamna M AlAli, Raghad Al-Husari, Amin Rigi, Mohammad K Al-Sharman, Dimitrios Makris, and Yahya Zweiri. A novel dynamic-vision-based approach for tactile sensing applications. *IEEE Transactions on Instrumentation and Measurement*, 69(5):1881–1893, 2019.

[135] P. J. Parikh, J. M. Fine, and M. Santello. Dexterous Object Manipulation Requires Context-Dependent Sensorimotor Cortical Interactions in Humans. *Cerebral Cortex*, 30(5):3087–3101, December 2019.

[136] Akella S. and Mason M. Parts orienting by push-aligning. *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, 1995.

[137] Xu Jingyi, Aykut Tamay, Ma Daolin, and Steinbach Eckehard. 6DLS: Modeling Nonplanar Frictional Surface Contacts for Grasping Using 6-D Limit Surfaces. *IEEE Transactions on Robotics*, 37(6):2099–2116, 12 2021.

[138] He K., Lahijanian Morteza, Kavraki L.E., and Vardi M. Y. Reactive synthesis for finite tasks under resource constraints. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 9 2017.

[139] Wang Y., Zhao L., Zhang Q., Zhou R., Wu L., Ma J., Zhang B., and Zhang Y. Alignment Method of Combined Perception for Peg-in-Hole Assembly with Deep Reinforcement Learning. *Journal of Sensors*, 2021:1–12, 9 2021.

[140] Bohg J., Pavone M., and Sadigh D. Fundamentals of Grasping.

[141] Hao Su. Deep learning for semantics, geometry, and physics in robotics. `https://robotmlcourse.github.io/SP20/lectures/lec8_grasping.pdf`, Spring 2020.