Preprint

Recognizing Actions during Tactile Manipulations through Force Sensing

Guru Subramani[1] Daniel Rakita[2], Hongyi Wang[2], Jordan Black[1], Michael Zinn[1] and Michael Gleicher[2]

***Abstract*— In this paper we provide a method for identifying and temporally localizing tactile force actions from measured force signals. Our key idea is to use the *continuous wavelet transform* (CWT) with the Complex Morlet wavelet to transform force signals into feature vectors amenable to machine learning algorithms. Our method uses these feature vectors to train a classifier that recognizes different actions. We demonstrate our approach in a system that records human activities with an instrumented set of tongs. Our system successfully identifies a wide range of actions based on a small set of labeled examples.**

I. Introduction

Recognizing the type and timing of tactile force actions is useful in a range of robotics applications. For example, recognizing contact, friction and slip is valuable in manipulation and automated testing. Recognizing force actions in human activities can be useful in robot programming by demonstration or task analysis. Identifying higher-level force actions exerted on objects such as gripping, snapping or releasing is particularly useful because it allows reasoning about motions at a semantic level of abstraction. Performing such action recognition from force measurements is attractive because forces are often necessary to differentiate between actions in circumstances where other sources of information are insufficient.

Recognition of force actions is commonly done in an *ad hoc* manner. Specific recognition algorithms can be devised for specific actions, however, such solutions do not generalize. Therefore, considerable effort may be required as the set of different actions to be recognized grows and more complex actions are considered. Our goal is to provide a force action recognition method that easily generalizes and scales to many diverse action types.

In this paper, we introduce an approach for performing recognition of tactile actions based on measurement of forces. Our method is based on machine learning and can be trained to classify a wide range of action types, providing not only the type of the recognized action, but also its timing. The key idea is to transform the measured force signals into *feature vectors* by using the Continuous Wavelet Transform (CWT) [1]. The CWT provides a concise description of the temporal neighborhood for every sample in a signal. Given a set of training motions recorded as force signals, along with action labels for each time step, our approach constructs a classifier that can label each time step of new motions.

We demonstrate and evaluate our approach in a system that records human tactile actions using an instrumented set of tongs. Considering only the force measurements from the tongs, our system is able to correctly label tactile actions such as inserting a cartridge, having an object slip from a grasp, and twisting a marker cap. It is sufficiently sensitive to distinguish grasping a soft object from a hard one. The system requires no per-action programming: it can be trained on new actions by simply providing a few labeled examples of the action.

The contributions of this paper include:

- We provide a technique using the Continuous Wavelet Transform with the Complex Morlet wavelet that encodes force measurements as features amenable to a variety of machine learning methods.
- We show the utility of this CWT technique in an approach that applies supervised machine learning to action recognition.
- We show that tactile actions can be recognized from force measurements alone.

II. Background

While others have developed action recognition systems using computer vision approaches [2] [3], we focus on action recognition from forces. Many task specific solutions to the problem of recognizing tactile actions from forces have been considered in the literature. Examples include contact-state detection for the peg in the hole problem [4], slip detection [5], [6] and snap detection [7]. However, each of these solutions involves specialized algorithms and therefore are difficult to generalize. In contrast, our approach can be trained to identify many different kinds of actions.

The tactile action recognition problem demands a representation of the force signals that provides both a compact expression of vibration and transient phenomena but also the ability to localize temporal events. The CWT meets these demands. It has many applications in image and signal processing such as clustering [8], detecting seismic events [9] and analyzing EEG signals [10]. We apply the CWT technique to tactile action recognition.

An alternative is to completely forgo useful signal characterization and use the raw signal directly in the recognition process. Such approaches, for example Convolution Neural

[1] Department of Mechanical Engineering, University of Wisconsin - Madison. [gsubramani|jeblack2|mzinn]@wisc.edu

[2] Department of Computer Sciences, University of Wisconsin - Madison. [rakita|hongyiwang|gleicher]@cs.wisc.edu

Networks(C-NN) [11], typically require many training examples, and are therefore inappropriate for our applications.

III. Key Idea

The central problem in tactile action recognition is to represent the measured force signal in a manner that is amenable to classification techniques. Our key idea is to use the *continuous wavelet transform* (CWT) to transform a scalar valued force signal into a vector-valued one where each time step is represented by a *feature vector*. Vector valued input signals are similarly transformed into concatenated wider feature vectors.

The CWT, with appropriate choice of wavelet (we use the Complex Morlet Wavelet), effectively summarizes the vibration and transient signal phenomena around each time step in a concise descriptor of the time step. Unlike other time-frequency characterization methods like the short time fourier transform(STFT), the CWT avoids windowing effects. We sample the wavelet at k scales, so that each time step of a scalar force signal is represented by a descriptor of length k. A vector valued force signal with vector length m has each time step represented by a feature vector of length $m*k$. We perform scaling and transformations to the CWT so the feature representation is suitable for machine learning algorithms.

After wavelet transformation, we treat each time step independently. In a training signal of length N, each time step has an associated action label. Hence we build a set of N pairs of $k*m$ length descriptor vectors and labels for training. Subsequent test signal labeling is done by applying the trained classifier to each time step of the test signal independently.

IV. Methods

The purpose of this work is to detect the type and timing of force actions. We use the CWT of the measured force signal using the complex morlet wavelet, as a set of features for representing force actions. The complex morlet wavelet extracts vibrational content from a signal. The intuition behind choosing the complex morlet wavelet is that many physical interactions between contacting objects are characterized by the frequency content of the signal. For example, contact with a hard object would result in a rapidly changing signal containing significant high frequency content while interactions with softer and/or heavier objects might result in greater low frequency content. Further, most physical interactions can be characterized by a linear system. Since higher order linear systems vibrate, and tend to amplify certain frequencies but not others, like the resonant frequency in second order systems, it can be seen that most interactions involve vibrational effects.

A. Continuous Wavelet Transform with the Complex Morlet Wavelet

The continuous wavelet transform of a signal can be understood as a transform of a signal which compares the shifted and scaled form of a prototype kernel function sometimes

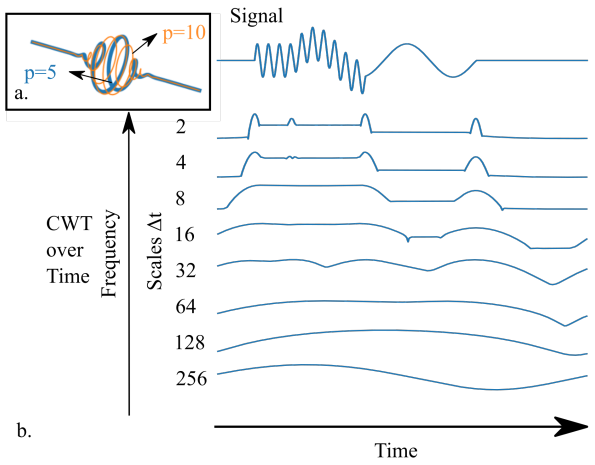

Fig. 1. **a. Complex Morlet Wavelet - Complex sinusoid windowed by a Gaussian at different p values. b. Signals represent the magnitude of the CWT coefficients over time on the test signal.**

referred to as the analyzing wavelet Ψ. This comparison is achieved by performing a convolution of the measured time signal with different scaled versions of the analyzing wavelet. We can define the kernel wavelet at different scaled versions of the analyzing wavelet as:

$$\Psi_{(a)} = \Psi\left(\frac{t}{a}\right) \tag{1}$$

where a is a scaling factor. Hence the CWT of a square integrable signal $f(t)$ using an analyzing wavelet Ψ is defined as:

$$X_{(a,b)} = \frac{1}{\sqrt{a}}\int_{-\infty}^{+\infty}\Psi^*\left(\frac{t-b}{a}\right)f(t)\,dt : a \neq 0 \tag{2}$$

where b is time shift parameter. Equation 2 is a convolution of scaled and translated version of the analyzing wavelet. We must still define the analyzing wavelet function. We use the Complex Morlet Wavelet [12] given by the following equation:

$$\Psi t = \pi^{-1/4} e^{ipt} e^{\frac{-t^2}{2}} \tag{3}$$

where the parameter p indicates the trade off between frequency and time resolution. On closer inspection we see that the Complex Morlet wavelet is a complex sine wave windowed with a Gaussian kernel centered at $t = 0$. Fig. 1a. shows the Morlet wavelet at different values of p.

B. Computing the CWT

We perform the CWT of a discrete signal f of length N in the Fourier space using the discrete Fourier transform. From the convolution theorem the CWT is given by:

$$X_{a,b} = \sum_{k=1}^{N-1} \bar{f}_k \bar{\Psi}^*(a\omega_k)\, e^{i\omega_k b\Delta t} \tag{4}$$

where $\bar{f}$ is the discrete Fourier transform given by:

$$\bar{f}_k = \frac{1}{N}\sum_{n=0}^{N-1} f_n e^{\frac{-2\pi i k n}{N}} \tag{5}$$

where the angular frequency ω is defined as:

$$\omega_k = \begin{cases} \frac{2\pi k}{N\Delta t} & : k \leq \frac{N}{2} \\ \frac{-2\pi k}{N\Delta t} & : k > \frac{N}{2} \end{cases} \quad (6)$$

Finally the CWT provides a coefficient for every time shift b and frequency(scale) a of the wavelet function. If the analyzing wavelet is a complex valued function then the CWT of that wavelet will also be complex valued. We can determine a power and phase spectrum from these coefficients given by:

$$M_{a,b} = \left\| X_{(a,b)} \right\| \quad (7)$$

$$\phi_{a,b} = arg\left(X_{(a,b)}\right) \quad (8)$$

Fig. 1b. provides the magnitudes of the CWT with the Complex Morlet wavelet on a test signal composed of two different frequencies.

C. Using the Complex Morlet Wavelet coefficients as Features for Machine Learning

The use of the Complex Morlet Wavelet (as the analyzing wavelet function shown in equation 3) require the selection of parameter, p, which controls the trade-off between frequency resolution and time resolution. For our analysis we chose a value of $p = 5$ balancing the need to (approximately) meet the admissibility criteria [13] of the wavelet while maintaining sufficient temporal resolution. To create a usable representation of the coefficients, we perform additional transformations:

- Use a logarithmic scale distribution: A linear scale distribution would create too many features and is highly redundant.
- Decompose the CWT complex coefficients into real-valued magnitude and phase components: The magnitude provides the intensity of force, while phase provides a measure of duration.
- Logarithmically scale the magnitudes: All the features have similar scales.
- Discard low frequency scales: Very low frequency scales do not characterize local behavior.
- Stack the CWT features from multiple channels in a signal: Force signals are usually vector quantities and contain multiple channels.

The CWT is a highly redundant transform if one takes all possible scales into account. The scale distribution is given by:

$$a_j = a_0 2^{j\Delta j}; where j = 0, 1, ...J \quad (9)$$

$$J = \Delta j^{-1} log_2\left(N\frac{\Delta t}{a_0}\right) \quad (10)$$

For our experiments we chose $\Delta j = 1$ and $a_0 = 2$. We found that this value of j was sufficient for sampling scales, albeit smaller values will give richer CWT coefficients. The DFT algorithm we use introduces windowing artifacts at the beginning and the end of the signal. It is useful to pad the signal at each end. The amount of padding depends on the width of the lowest frequency scale used.

Splitting the complex coefficients into magnitudes and phases allows resolving between actions of the same magnitude but with different phase. Finally standardizing the magnitude by computing their natural logarithm is useful as signal amplitudes can vary by orders of magnitude. In particular, high frequency signals generally have much lower amplitudes when compared to low frequency ones but are critical in identifying actions such as contacts with objects.

In cases where we have more than one channel in a signal, we stack up the coefficients to create a combined representation. If we have q channels in a signal and use k scales then the features X_b representing time instance b of the signal would be:

$$X_b = \begin{bmatrix} M_{b1}^T & M_{b2}^T & ... & M_{bq}^T & \phi_{b1}^T & \phi_{b2}^T & ... & \phi_{bq}^T \end{bmatrix}^T \quad (11)$$

Once we compute this, each sample b of the signal has a *feature vector* X_b. For a signal of length N there will be N such feature vectors which can be expressed as examples in an unlabeled data set. Note that these examples have order as they correspond to samples in the signal. If we were to provide labels to each of these examples then we could treat these as members of a labeled data set. Since the examples are associated with a sample time, the labels for these examples are also associated with that same sample time. Hence we can define a new time series $L(t)$ which evaluates to the label names corresponding to the signal at time t.

D. A Sample Scenario for Supervised Learning

Consider a sample data set consisting of k signals containing multiple channels with different portions of their signals labeled. We can consider a training set T_r with $k-1$ signals and test set T_s with the last signal k. The supervised learning procedure would be as follows:

1) For each signal in T_r, compute the CWT features to create their associated *examples*. An *example* is a pair, consisting of a feature vector and an associated label name, corresponding to a specific sample time.
2) Use these *examples* to train a model.
3) For signal T_s compute the CWT feature vectors at each sample time.
4) Predict labels at each sample instance of the test signal T_s to get a label time series L.
5) Filter the time series L.

E. Choice of Supervised Learning Algorithm

Since we have abstracted the signal representation into a set of features at every time instance, it is possible to use almost any conventional classifier for the supervised learning task. In practice we see that the Neural Network gives us the best execution time and relatively good accuracy when compared to other methods we tested including, Naive Bayes, Decision Trees, SVMs and K-Nearest Neighbor. Note that feature representation affords us the flexibility to

choose many algorithms. For the experimental evaluations performed here, we used a Neural Network from Scikit-learn [14] with default options which contains one hidden layer with 100 neurons.

F. Filtering Predicted Labels

The raw results from classification when arranged in time contain chattering. Since the predicted label names are in the form of a time series, we can filter them. Some filtering methods are:

- Implement a probability thresholding filter which removes labels classified with a probability less than a particular threshold and sets them to unclassified.
- Define a condition that, if an action is predicted to occur for less time than a factor of the minimum duration of that particular action in the signals used for training, it is labeled as unclassified. We refer to this as the Drop Heuristic.
- Assign discrete numbers to each type of label and filter these using a median filter.

G. Why not other Wavelets?

Using the Morlet wavelet allows encoding both phase and magnitude information. If we were to use the real Morlet wavelet we would not be able to distinguish between phase and magnitude information. This is a particularly bad choice since this wavelet would phase in and phase out, changing the magnitude of the wavelet in a signal, even for one whose magnitude is constant. Alternatively, if we were to use the Haar wavelet, we would not be able to extract phase information.

To demonstrate this we compute the CWT features for a trapezoidal test signal using:

- Complex Morlet Wavelet with magnitude and phase
- Complex Morlet Wavelet with only magnitude
- Derivative of Gaussian Wavelet (DOG)
- Haar Wavelet [13]

Fig. 2. shows this comparison. We see that the Complex Morlet Wavelet with both magnitude and phase, is the only wavelet that can distinguish between increasing and decreasing portions of the test signal.

V. Experimental Setup for Evaluating Methodology

A. Data Collection System

We developed an object manipulation device to record user performed tasks. While for this work, we do not use the kinematic (motions) information, the device is fully capable of recording its spatial location and the grasp configuration through an HTC Vive VR controller and an encoder. Force interactions are recorded using Optoforce force sensors. Data collection and synchronization was performed using ROS. The body was designed and 3D printed in-house. This is shown in Fig. 3. We record 3 axis force values from each sensor. We convert these 6 signals into 4 signals by computing the normal and tangential components of the native force signals. This transforms the signals into a form that is approximately invariant to forces in the plane tangential to the force sensors.

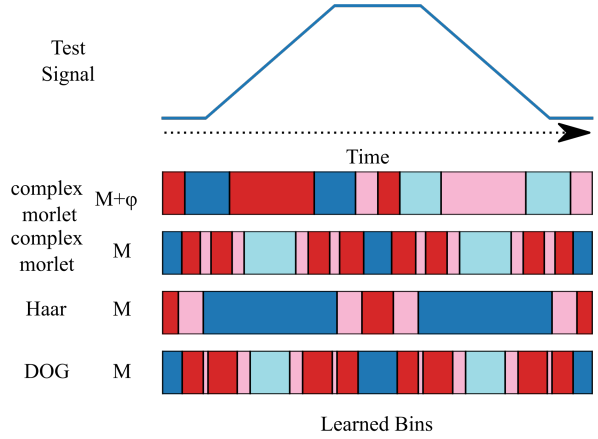

Fig. 2. **Clustering CWT coefficients using 4 clusters with the k-means algorithm over the Trapezoidal test signal. Row 1 shows the complex morlet wavelet capable of distinguishing between rising and falling edges while other wavelets cannot. Colors correspond to different clusters. M - Magnitude, φ - Phase**

B. Experimental Evaluation Methodology

A user performs specific tasks which are recorded by the instrumented device described in the previous section. For clarity, we will define three words and their relationships:

- Force Action: Something that can be represented by a single action which describes what is happening, like a verb.
- Demonstration: A single recording of a set of signals while performing a series of force actions.
- Task: A specific choreography of a demonstration.

A task is the overall objective of the demonstration. For example, a task could be the process of picking up a bottle and placing it in a different location. A demonstration is an instance of a task. Each task can be broken up into different force actions like grab, still or release. We can think of force actions as primitives to create tasks.

The data of one demonstration is essentially a time series of multiple signals and each time instance is labelled with a force action label. In our experiments, a human labels sections of the time series. Each time step is labeled with a force action in each demonstration. We define a set of force action labels for each task. In each experiment two

Fig. 3. **Instrumented tongs used to manipulate objects.**

coders (in consensus) define which locations in the force signal correspond to force actions but independently code the demonstrations allotted to them. Each task is recorded with a frame by frame, time synchronized video recordings of the task. The human coders code the task demonstration by looking at the force signal, using the video recording as an aid.

C. Sample Demonstration

Fig. 4 shows the force signals for a sample demonstration. The choreography of the force actions that occur during a demonstration of the task is also shown. In this experiment a lego block is picked up from a table and fastened on to another lego, then it is removed and placed back on the table. From a set of demonstrations and human coded labels we learn a model which is capable of classifying each point in time with a label. During transitions between force actions, the classified labels chatter between two different label names. We mitigate this by applying the filtering techniques described in IV-F. We used a median filter of window size 10, a probability threshold of 80% and a drop factor of 1/8.

D. Experimental Evaluation Metrics

In evaluating the model we made use of the following metrics:

1) Qualitative Metric TIMELINES: In each timeline plot pair, the upper plot displays the human coded labels while the lower plot displays the predicted label. If the timelines in the pairs (ie. upper and lower plots) match then the trained classifier has predicted every label correctly. A qualitative measure of the models performance can be obtained by examining the alignment of the upper (human coded labels) and lower (predicted labels) timelines. To generate the timelines

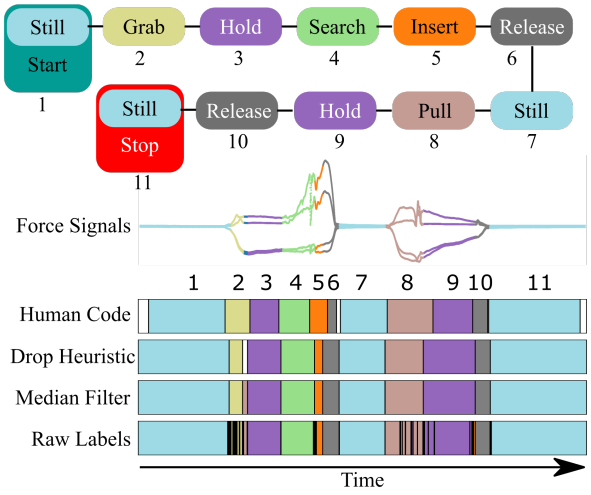

Fig. 4. **Top - Semantic timeline of Lego assembly and disassembly task with the corresponding labels numbered according to their occurrence in the human coded timeline. Bottom - Human Code corresponds to the labels given by a human coder, Drop Heuristic corresponds to applying the Drop Heuristic and the Median Filter. Median filter corresponds to only applying the median filter to the classified labels. The white spaces correspond to unlabeled samples.**

we train and test on demonstrations in a leave one out type cross validation setup. In other words, for a particular task, we test on one demonstration and train the model using the others. In this fashion, we test on all demonstrations and plot the results in the form of timelines of predicted labels. The colors correspond to the individual action labels. The white color corresponds to unlabeled examples. If they occur in the human coded time line then these time samples are dropped during training. If the algorithm has insufficient confidence about the label then it labels them white.

2) Quantitative Metric: We compute a confusion matrix which provides information on which label is misclassified with what other label. To build this matrix, we use the same cross validation technique described above unless stated otherwise.

Since these are force signals and the location of a force action is not well defined it is difficult to come up with a fair quantitative metric to describe the algorithms accuracy. At the interfaces between two different force actions, it is very difficult for the human coder to consistently code the force action. Since these are signals, the adjacent force actions bleed into each other. Some force actions like still, dominate others. Hence, mere classification accuracy is not a good way of reporting the algorithm's behavior.

VI. Results

In this section we validate our feature representation method by performing two sets of experiments:

- Detecting force actions in different demonstrations of the same task. Training set and test set include demonstrations of the same type of task.
- Detecting force actions in demonstrations of a different task. Training set includes demonstrations from one type of task. Test set includes demonstrations from a different task.

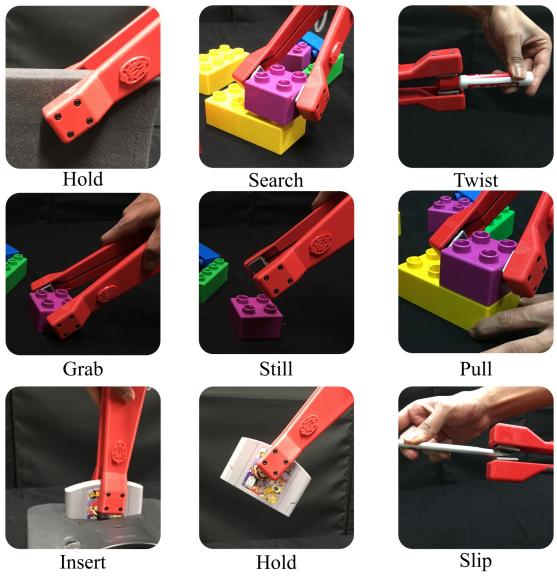

Fig. 5. **Force actions and their corresponding object manipulations**

A. *Detecting force actions in similar tasks*

We ran multiple experiments to see if we could identify the same force action in different demonstrations of the same task. Fig. 5 shows a glossary of actions with their label names. We chose a few scenarios which were found in robotics literature, namely:

- Peg in the Hole problem
- Slip and Friction detection
- Distinguishing between hard and soft objects

1) Peg in the Hole Problem - Inserting a Nintendo 64® Cartridge: In this task we pick up a Nintendo 64® cartridge and insert it into the console. The cartridge clicks into place during insertion. Fig. 6 shows the choreography of the task along with the force signals and associated force action labels. Fig. 6 displays an experimental evaluation of the demonstrations. We see that the pull action is misclassified frequently. On close inspection we see that this is not surprising as it might be due to coding errors. The human coder has difficulty coding the pull and grab action consistently between demonstrations. Although the coder can observe the force signals along with the video recording of the task, the video aid is not that helpful as the cartridge is enclosed in the game console and occluded during insertion. We can also see that the pull action is confused with the grab action. While the human coder does not consider a grab before a pull action, the algorithm occasionally finds a grab action. Intuitively, the cartridge must be grabbed before it can be pulled out of the console.

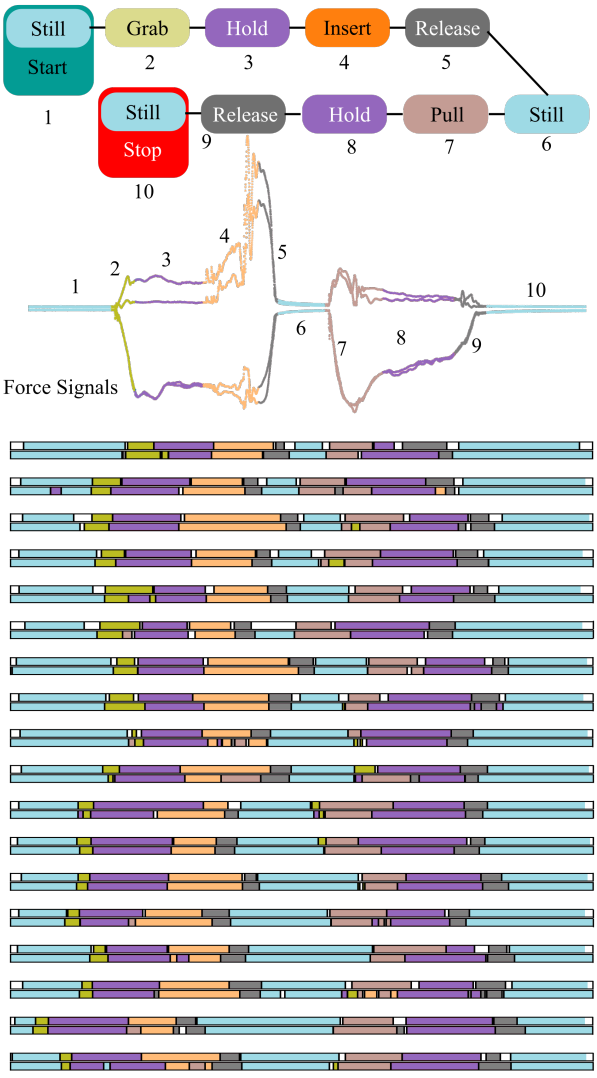

Fig. 6. **Summary of the Nintendo 64® trials. Top - Timeline of force actions in numerical order as they occur. Bottom - Summary of all the demonstrations' predicted labels and human coded labels for each demonstration of the Nintendo 64 task. In each pair, the upper color coded timeline bar corresponds to the human coded labels. The lower bar corresponds to the predicted labels. Notice that the predicted force actions align with the human coded force actions. TIMELINES MUST BE VIEWED AS PAIRS AND IN COLOR, see Sec. V-D**

2) Peg in the Hole Problem - LEGO® Block Assembly: We evaluated the algorithm by examining LEGO® block assembly. This task is characteristic of tasks found in various assembly tasks that involve both interference and transition fits between parts. A defining characteristic of LEGO® block assembly is the need to fumble for the proper mating configuration between parts, referred to here as the *search* force action. From Fig. 7 we see that the algorithm consistently identifies the search actions and distinguishes this from the hold and insert actions.

3) Slip and Friction Detection - Detecting Slip During Grasp: In this scenario, we attempt to recognize different force actions that occur during slip. We simulate this by grasping a pen between the force sensors and pulling until the pen is released from its grasp. In this case, we chose an peculiar oblong object (i.e. a writing stylus) to allow

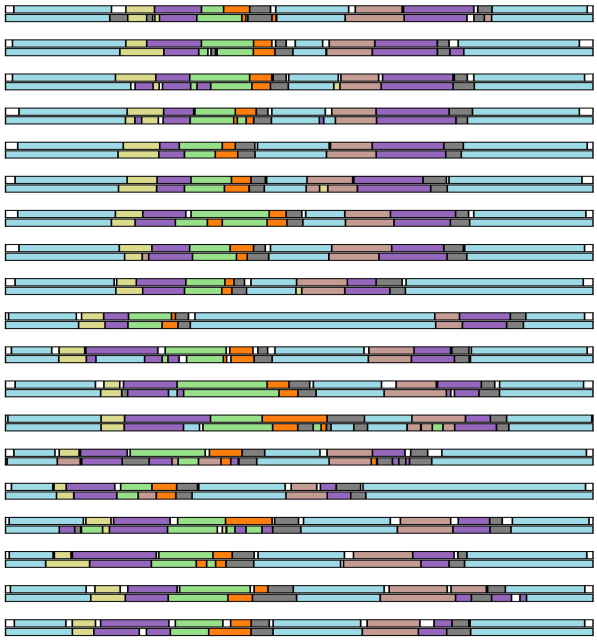

Fig. 7. **Summary of the LEGO® trials. Please refer to Fig. 4 for the label names. In each pair of timelines, the upper timeline corresponds to the human coded labels and the lower timeline corresponds to the predicted labels. Notice that in almost every trial the order and locations of the force actions are predicted correctly. TIMELINES MUST BE VIEWED AS PAIRS AND IN COLOR, see Sec. V-D**

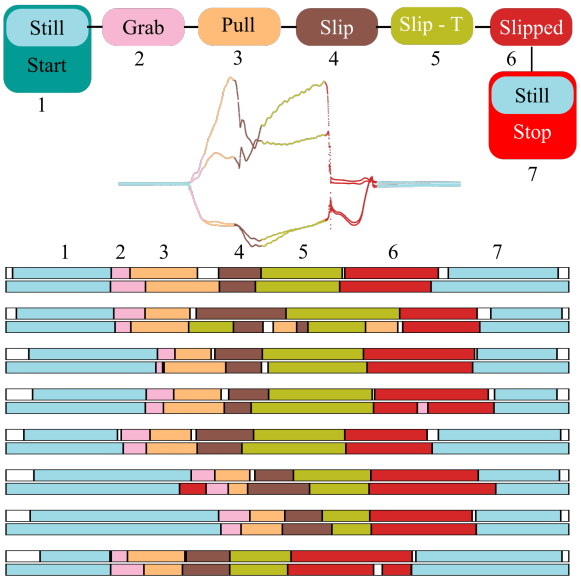

Fig. 8. **Slip detection task with stylus. The predicted labels clearly align with the human coded labels. The predicted labels have very close arrangements when compared to the human coded labels. TIMELINES MUST BE VIEWED AS PAIRS AND IN COLOR, see Sec. V-D**

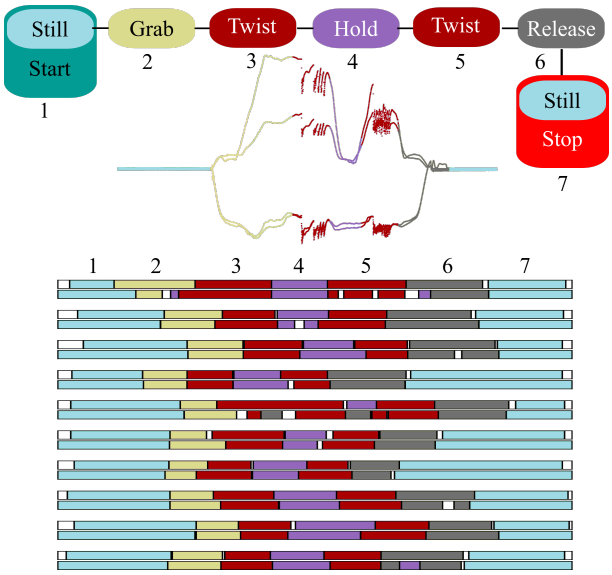

Fig. 9. **Friction detection task with the dry erase Expo® marker. The predicted labels clearly align with the human coded labels. The predicted labels have very close arrangements when compared to the human coded labels. TIMELINES MUST BE VIEWED AS PAIRS AND IN COLOR, see Sec. V-D**

examination of the algorithm's performance at various stages of slip. The variable diameter of the stylus causes changes in the friction force. The labels describing the slipping included (1) *slip*, indicating onset of motion between the stylus and the tongs, (2) *slip transition* indicating motion between the tongs and the stylus. The end of the slip action was labeled as *slipped* indicating the completion of the slip action, identified during the experiment through the clap of the instrumented tongs. Fig. 8 shows the choreography of the action. We see good time alignment and order of actions for most demonstrations.

4) Slip and Friction Detection - Detecting Friction Force: To evaluate the algorithm's ability to detect friction actions, which contain significant high frequency vibrations, we used our instrumented tongs to apply torque to a dry erase marker cap. The cap has a significant amount of static friction which results in sticky, discontinuous motion when twisted. Fig. 9 shows the action choreography and the results. The technique does well but has difficulty during the hold action. This may be due in part to the difficulty that the human coders have in understanding and identifying the force actions. For example, holding the marker cap after the first *twist* action can be overlooked while coding.

5) Distinguishing between Hard and Soft objects: In this experiment we evaluate the algorithm's ability to detect objects of different hardness. A foam block and a Nintendo 64® cartridge were used as soft and hard objects respectively. We trained and tested using concatenated datasets of the foam block and the Nintendo 64® cartridge. The foam block task was simple pickup and drop of the foam block. Fig. 10 shows the result in a confusion matrix.

B. Detecting similar force actions in different tasks

In the previous section, we show examples of our recognition algorithm where the training and test sets use demonstrations from the same task. The purpose of this section is to demonstrate how well our algorithm can recognize learned force actions in a new task. In this experiment the training set contains demonstrations from the LEGO® assembly task. The test set contains examples from the Nintendo 64®

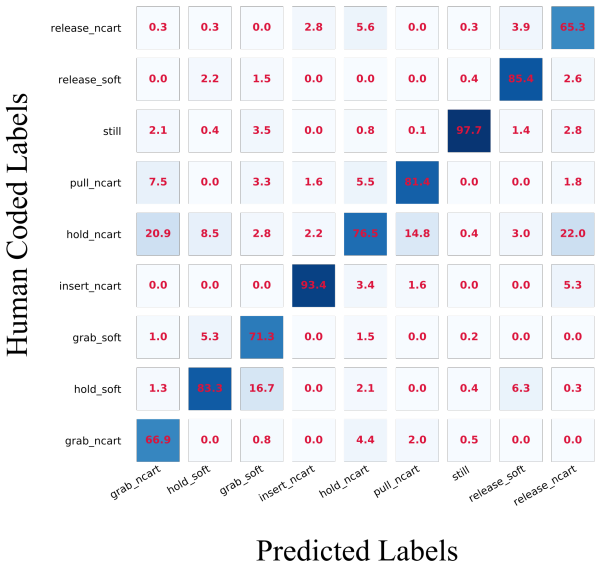

Fig. 10. **Confusion Matrix for training set containing both hard and soft object manipulations. Larger numbers along the diagonal correspond to better results. The soft object was the foam block (labelled with suffix _soft) while the hard object was the Nintendo 64 Cartridge (labelled with suffix _ncart). The algorithm can clearly distinguish between actions performed on the hard object compared to actions performed on the soft object.**

Fig. 11. **Confusion Matrix: Trained on the Lego task and tested on the Nintendo 64 task. There are no search actions in the human coded labels. Since the Lego task contains search actions, the classifier finds them close to the insert actions. While inserting a cartridge the user feels for the opening which corresponds to the search action.**

cartridge insertion task. To evaluate our results we make use of a confusion matrix shown in Fig. 11 described in V-D.

Note that the human coder does not code a *search* label for the Nintendo 64® task. We see that the algorithm misclassifies the *insert* label with the *search* label but classifies all other labels correctly. The *search* label corresponds to the fumbling of the LEGO® block. The cartridge was not always inserted smoothly into the Nintendo 64® and sometimes the user tried to feel for the opening. The human coder coded all these interactions along with the insertion of the cartridge with an *insert* label. It is reasonable for the algorithm to misclassify some of the *insert* labels as *search* labels. For the other labels, the algorithm appears to do well. Overall, the algorithm finds correspondences between similar force actions even when the training set contains no demonstrations from the tested task.

VII. Conclusions and Discussion

We proposed a feature representation technique which characterizes the local behavior of a force signal with a set of features at each time sample. We use the CWT with the Complex Morlet analyzing wavelet as a way to transform a time domain signal into feature vectors describing each sample instant.

We show that this feature representation provides a good way to describe a time domain force signal through a series of experiments and example supervised learning tasks. We intentionally provided little post-processing to the estimated test signals to study the limitations and advantages of using this representation.

At boundaries between different force actions we see chattering of the predicted labeled actions (before any post-processing). Such chattering indicates imprecision in localizing the beginning and end of actions as they often do not have hard boundaries. In some scenarios where correct ordering and approximate timing are sufficient it may be acceptable to use filtering techniques during post-processing.

The post-processing methods used in this paper require further improvement and merely act as a starting point for filtering techniques. These methods are insufficient to correct classification errors like labeling a *hold* action for a very brief period of time between two *still* actions. Hence the CWT method requires good post-processing techniques to correct such problems.

As a supervised learning approach, our method is limited by and may not generalize beyond the training data. For example, in this paper we show how our algorithm can detect slip between the stylus and tongs. However, it may not perform well on a different object using the training examples from the stylus experiments as they do not contain examples of different objects. Another major draw back with this method is that it is offline. Many robotics applications require realtime recognition.

Possible future work would be to apply probabilistic models which could use labels occurring before or after as priors to correct predicted labels. It would be advantageous to develop methods to make use of the CWT based models for realtime or approximately realtime recognition.

VIII. Acknowledgment

Support for this research was provided in part by NSF award 1208632 and the University of Wisconsin-Madison Office of the Vice Chancellor for Research and Graduate Education with funding from the Wisconsin Alumni Research Foundation.

References

[1] S. Mallat, *A wavelet tour of signal processing*. Academic press, 1999.
[2] R. Poppe, "A survey on vision-based human action recognition," *Image and vision computing*, vol. 28, no. 6, pp. 976–990, 2010.
[3] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 1, pp. 221–231, 2013.
[4] I. F. Jasim and P. W. Plapper, "Contact-state monitoring of force-guided robotic assembly tasks using expectation maximization-based Gaussian mixtures models," *International Journal of Advanced Manufacturing Technology*, vol. 73, no. 5-8, pp. 623–633, 2014.
[5] C. Melchiorri, "Slip detection and control using tactile and force sensors," *IEEE/ASME Transactions on Mechatronics*, vol. 5, no. 3, pp. 235–243, 2000.
[6] Z. Su, G. E. Loeb, G. S. Sukhatme, and S. Schaal, "Force Estimation and Slip Detection for Grip Control using a Biomimetic Tactile Sensor."
[7] J. Rojas, N. Yamanobe, E. Yoshida, and K. Nagata, "Towards snap sensing," no. September 2013, 2017.
[8] S. C. Gholamhosein Sheikholeslami and A. Zhang, "Wavecluster: A multi-Resolution Clustering Approach for Very Large Spatial Databases," *International Journal on Very Large Data Bases (VLDB98)*, vol. 8, no. 3-4, pp. 428–429, 1998.
[9] R. Zhang, "Spectral decomposition of seismic data with CWPT," *The Leading Edge*, vol. 27, no. 3, p. 326, 2008.
[10] S. J. Schiff, A. Aldroubi, M. Unser, and S. Sato, "Fast wavelet transformation of EEG," *Electroencephalography and Clinical Neurophysiology*, vol. 91, no. 6, pp. 442–455, 1994.
[11] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. April 2016, pp. 255–258, 1995.
[12] G. Kaiser, *A friendly guide to wavelets*. Springer Science & Business Media, 2010.

[13] C. Torrence and G. P. Compo, "A Practical Guide to Wavelet Analysis," *Bulletin of the American Meteorological Society*, vol. 79, no. 1, pp. 61–78, 1998.

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