Semantic Gaze Labeling for Human-Robot Shared Manipulation

Reuben M. Aronson
Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania
rmaronson@cmu.edu

Henny Admoni
Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania
henny@cmu.edu

ABSTRACT

Human-robot collaboration systems benefit from recognizing people's intentions. This capability is especially useful for collaborative manipulation applications, in which users operate robot arms to manipulate objects. For collaborative manipulation, systems can determine users' intentions by tracking eye gaze and identifying gaze fixations on particular objects in the scene (i.e., *semantic gaze labeling*). Translating 2D fixation locations (from eye trackers) into 3D fixation locations (in the real world) is a technical challenge. One approach is to assign each fixation to the object closest to it. However, calibration drift, head motion, and the extra dimension required for real-world interactions make this position matching approach inaccurate. In this work, we introduce velocity features that compare the relative motion between subsequent gaze fixations and a finite set of known points and assign fixation position to one of those known points. We validate our approach on synthetic data to demonstrate that classifying using velocity features is more robust than a position matching approach. In addition, we show that a classifier using velocity features improves semantic labeling on a real-world dataset of human-robot assistive manipulation interactions.

CCS CONCEPTS

• **Human-centered computing** → *User models*; *Interaction devices*; • **Computer systems organization** → *Robotic components*;

KEYWORDS

eye tracking, intention recognition, semantic gaze labeling, human-robot interaction, assistive robotics

ACM Reference Format:
Reuben M. Aronson and Henny Admoni. 2019. Semantic Gaze Labeling for Human-Robot Shared Manipulation. In *2019 Symposium on Eye Tracking Research and Applications (ETRA '19), June 25–28, 2019, Denver , CO, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3314111.3319840

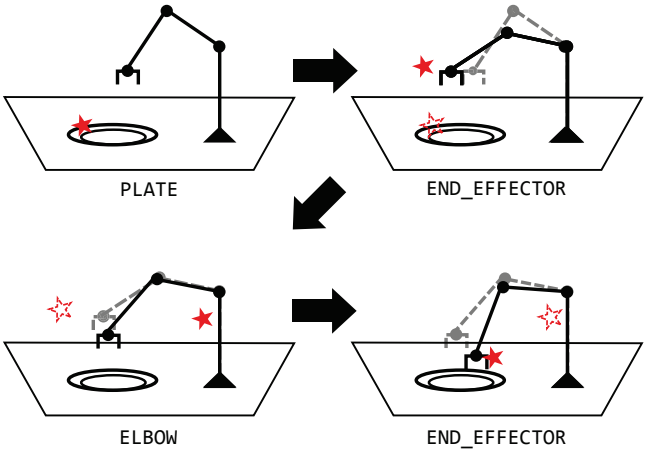


Figure 1: Schematic representing the semantic gaze labeling problem. Given a gaze point of regard and the location of several (possibly moving) objects in the scene, determine which object the participant is looking at. This problem is made more difficult by the errors induced by the motion of scene objects and 3D gaze calibration.

1 INTRODUCTION

Human-robot collaboration systems benefit from recognizing people's intentions, which can often be assessed through eye gaze. A particularly promising application for human-robot collaboration comes from the realm of assistive robotic manipulation. In this domain, a user controls a robot arm through an operating interface to perform a task. This scenario applies to various tasks, from factory operations to space robotics to disaster recovery, but among the most exciting is when using assistive devices. Robot arms that mount onto wheelchairs are already commercially available and in use, but these devices tend to be difficult to use especially for complex tasks [Herlant et al. 2016]. To improve these systems, roboticists build systems to understand the user's intentions and add automation to the robot to help them accomplish their task [Aronson et al. 2018; Orlov et al. 2018]. And since eye gaze is such a useful signal for understanding user intent, exploring its utility in this application is particularly valuable.

In this paper we focus specifically on the problem of *semantic gaze labeling*: given a scene and a gaze position within the scene, what object is the user looking at? In this formulation, we assume that the user is looking at one of a finite set of (possibly moving) objects or locations, here termed *keypoints*. This problem is particularly resonant within the manipulation task domain, as (1) there are relatively few task-relevant objects in the scene, and their positions may already be known due to the system requirements for the rest

ETRA '19, June 25–28, 2019, Denver , CO, USA

ACM ISBN 978-1-4503-6709-7/19/06…$15.00
https://doi.org/10.1145/3314111.3319840

of the system and (2) people's gaze is particularly task-relevant during manipulation [Brouwer and Knill 2007; Hayhoe et al. 2003; Johansson et al. 2001; Land and Hayhoe 2001], especially robotic manipulation [Aronson et al. 2018]. Semantic gaze labeling is one useful preprocessing step for understanding what people are intending to do; it yields information such as look-ahead fixations indicating planning [Aronson et al. 2018; Mennie et al. 2007] and can provide a signal for noting unexpected or problematic events that occur [Aronson and Admoni 2018].

However, solving this semantic gaze labeling problem within real-world domains such as robotic manipulation can present significant challenges. The interaction occurs in a complex, 3D environment, which presents a much wider error space. While significant work has been done to investigate this problem in a real-world setting [Atienza and Zelinsky 2005; Pfeiffer 2012; Pfeiffer and Renner 2014], introducing a third dimension inherently raises the threshold for accuracy. When people are allowed to move their heads freely, including the effect of head position relative to scene position can also induce inaccuracy. Though highly calibrated systems can compensate for these effects, it is nevertheless valuable to explore algorithmic solutions to ease the accuracy burden of gaze collection systems.

In this work, we present features derived from the relative motion between subsequent fixations, termed *velocity features*. These features reduce the impact of slow-changing offsets (e.g. calibration drift or tracker motion), since they consider only relative motion. We evaluate the usefulness of these velocity features on both synthetic data and real data. On synthetic data, which is generated by randomly generating keypoint and fixation patterns and manually adding increasing offset magnitudes, using velocity features increases classification accuracy when offset size exceeds a significant fraction of the average inter-keypoint distance. We also evaluate these features on data collected from a collaborative manipulation task using a robot. On this task, using the velocity features improves overall semantic classification accuracy from 65.9% to 75.8%, with most improvement coming from data with higher offsets.

This paper begins by summarizing related work in Sec. 2 and defining the semantic gaze labeling problem in Sec. 3. Then, it describes how to calculate and use velocity features in Sec. 4. We demonstrate the usefulness of these features in both synthetic data (Sec. 5) and on a real dataset collected from interactions with a robot (Sec. 6). Finally, we discuss limitations of this approach and future work.

2 RELATED WORK

Using eye gaze to understand people's mental states has taken many forms depending on the task. For understanding behavior such as reading [Just and Carpenter 1980] or driving [Braunagel et al. 2015], bottom-up analysis based on raw gaze features has proven useful. However, for particular tasks, labeling which of a small set of objects someone is looking at can provide valuable analysis. One particular task that this is relevant for is object manipulation: the relationship between people's intentions and their gaze in this domain is well categorized, with look-ahead fixations indicating planned object placement [Hayhoe and Ballard 2005; Hayhoe et al. 2003; Land et al. 1999; Land and Hayhoe 2001] in tasks such as block movement [Johansson et al. 2001] or food preparation [Land and Hayhoe 2001]. This phenomenon extends to manipulation using a robot rather than by hand: people illustrate similar patterns of planning glances but also occasionally fixate on the manipulator arm [Aronson et al. 2018]. Explicitly identifying the objects being examined is useful for such applications as manipulation planning and assistance [Li et al. 2017] or failure recovery [Aronson and Admoni 2018].

Several systems have been proposed for solving the semantic labeling problem [Hagihara et al. 2018; Li et al. 2017; Paletta et al. 2013a; Singh et al. 2018]. One recent representative example is EyeSee3D [Pfeiffer and Renner 2014] (and its follow-up EyeSee3D 2.0 [Pfeiffer et al. 2016]). In this approach, head pose is recovered from fiducial tags placed in the workspace, and this pose information is used to perform ray tracing to intersect the line of sight with given object meshes. Compared to manual labeling, this system performed with 64% accuracy, indicating the difficulty of this problem. Other approaches use alternate methods for head pose recovery or environment modeling, but are fundamentally similar in their approach to the semantic gaze labeling problem.

More sophisticated approaches for labeling have also been explored. Pfeiffer [2012] discusses expanding a single gaze ray to an *attention volume*, a Gaussian centered around the point of regard. Mantiuk et al. [2013] discuss a strategy for incorporating both keypoint positions and velocities at each timestep into the labeling process. Vidal et al. [2013] uses correlations between screen-based targets moving in different patterns and eye motion to identify which of the targets is inducing smooth pursuit behavior. Bernhard et al. [2014] presents a probabilistic approach to semantic gaze labeling, in which a *gaze-to-object mapping* (GTOM), or a probability distribution over a finite set of keypoints, is generated by combining different position or velocity features with Bayesian fusion; this work also presents several possibilities for generating probabilities from local position or velocity values. Our contribution supplements these works by presenting bulk features that work on the time scale of fixations rather than single timesteps.

3 SEMANTIC GAZE LABELING

There are a variety of ways to use eye gaze to recover different parts of someone's mental state, and each task domain can have different approaches. For tasks such as manipulation, in general there are only a relatively small number of objects that are relevant, and these objects can be enumerated in advance. Understanding when people look at which objects in a scene can be a revealing signal for what they are intending to do (via look-ahead fixations [Mennie et al. 2007]) or if an error has occurred [Aronson and Admoni 2018].

In order to use gaze in this way, we must be able to robustly identify a gaze or fixation location with a particular object. We can define this *semantic gaze labeling* problem as follows: Within an egocentric video, let the eye gaze position over time τ be denoted $(x_\tau, y_\tau) = \bar{x}_\tau$. Choose a finite number n of *keypoints* $(k^i_{x,\tau}, k^i_{y,\tau}) = k^i_\tau$ for $i = 1 \cdots n$ that track the locations of specific objects in the scene. (These object positions can be determined from projecting the 3D scene positions into the camera position, through video object tracking, or any other method). The semantic gaze labeling problem consists of assigning to each time a label ℓ_τ, an index $1 \cdots n$

into the keypoints representing the object being fixated at that time. To reduce the data complexity required, we can preprocess the gaze signal by dividing the gaze into individual fixations with mean locations f_t, where we assume that each fixation shares a label, and discarding the saccadic transitions between fixations. (A single fixation index t spans a range of times $[\tau_{t,0}, \tau_{t,1}]$.) Then, the labeling problem becomes assigning a label ℓ_t to each fixation location f_t.

4 APPROACH

The most straightforward approach for semantic labeling is to assign to each fixation the closest object in the scene. In this paper, we call this *position matching*. If the point of regard is captured accurately, and the relevant scene object positions are known precisely, the semantic labeler can simply label each fixation with the closest object (using some appropriate metric, e.g. cosine distance between the rays from the head position). This or similar approaches have been used successfully for highly accurate data [Hagihara et al. 2018; Li et al. 2017; Paletta et al. 2013a; Pfeiffer and Renner 2014; Pfeiffer et al. 2016; Singh et al. 2018].

However, this approach performs poorly in the presence of errors. Inaccuracy during initial calibration can cause static errors in the scene, where the true gaze location is somewhat off from a computed point of regard. Moreover, and particularly when using mobile eye trackers, gradual motion of the sensor components can cause the calibration to drift over time. This problem is exacerbated when reconstructing full 3D scenes, as the increased number of parameters and sensors needed for unrestricted gaze direction reconstruction all increase the sensitivity of the calibration to inaccuracy or drift.

To mitigate this problem, we borrow techniques from signal processing. First, we note that these calibration drifts are usually slow with respect to actual eye motion. Eye tracker calibration error is constant; eye tracker slip is often slow; and object motion is generally slow relative to eye motion. Therefore, we can expect that relative to the time scale of individual gaze fixations, the calibration error will stay generally constant. Furthermore, we assume that we know the locations of all of the relevant objects in the scene (the *keypoints*). Therefore, to the standard *position features* that encode the distance from the gaze fixation to each keypoint, we add in *velocity features*. These features compare the distance traveled from the previous fixation to the current one with the changes between the previous keypoint and each of the keypoints at the current time.

In particular, when labeling a fixation f_t, the velocity feature consists of comparing the change required to transform the previous fixation location f_{t-1} to f_t with the n transformations that would move the previous keypoint label to each of the current keypoints at those appropriate times. That is, if the fixation at $t-1$ had been previously labeled i, compare this fixation velocity with the velocity between k^i_{t-1} and k^j_t for all j. Since the static error term is added at both the previous and the current time step, computing the velocity features by taking their difference cancels much of it out[1]. Therefore, we can expect adding in these features to improve the classification accuracy with larger error. The position and velocity features are represented schematically in Fig. 2.

One approach to this problem would perform the entire calculation in 2D. That is, we could compare Euclidean distances between pixel locations representing keypoints and gaze targets. However, for better accuracy in the 3D environment, we choose instead to consider rays starting at a single point (represented by the position of the egocentric camera used for eye tracking) and projected into the environment. These rays can be identified with single pixel locations on the egocentric video camera frame; however, their vector representation must be preserved when calculating differences and differences-of-differences, as is necessary for these feature representations. Sec. 4.1 describes the appropriate calculations for determining these features in ray space, and Fig. 2c shows a representation of these projection rays.

In addition, velocity features are not sufficient on their own. Note that velocity features are reliant on the *previous* label being correct: otherwise, the features are computed relative to the wrong point and the data is useless. Moreover, the classification must be seeded with a correct initial label, for which it cannot use the velocity features. Therefore, a complete algorithm combines both position and velocity features to perform accurate semantic labeling.

4.1 Feature definitions

Mathematically, we can compute the features as follows. Represent each fixation average point $f_1 \cdots f_T$ as a unit vector indicating the direction of gaze, and compute the average direction for each keypoint i at each fixation time t as

$$k^i_t = \operatorname*{mean}_{\tau=\tau_{0,t}\cdots\tau_{1,t}} k^i_\tau,$$

the average of the keypoint position over the duration of the fixation. Both position and velocity feature can be computed from these components.

4.1.1 Position features. For each fixation t, compute the position features p_i as the cosine distance between the fixation location and the position feature. That is,

$$p^i_t = 1 - f_t \cdot k^i_t$$

is a normalized representation of the inner product between the unit vectors representing the average fixation and keypoint rays.

4.1.2 Velocity features. To compute velocity features, we must first compute the change over time of the fixation and each keypoint. We can represent these changes as *quaternions*, standard four-parameter representations of 3D rotations that are relatively easy to compute and compare.

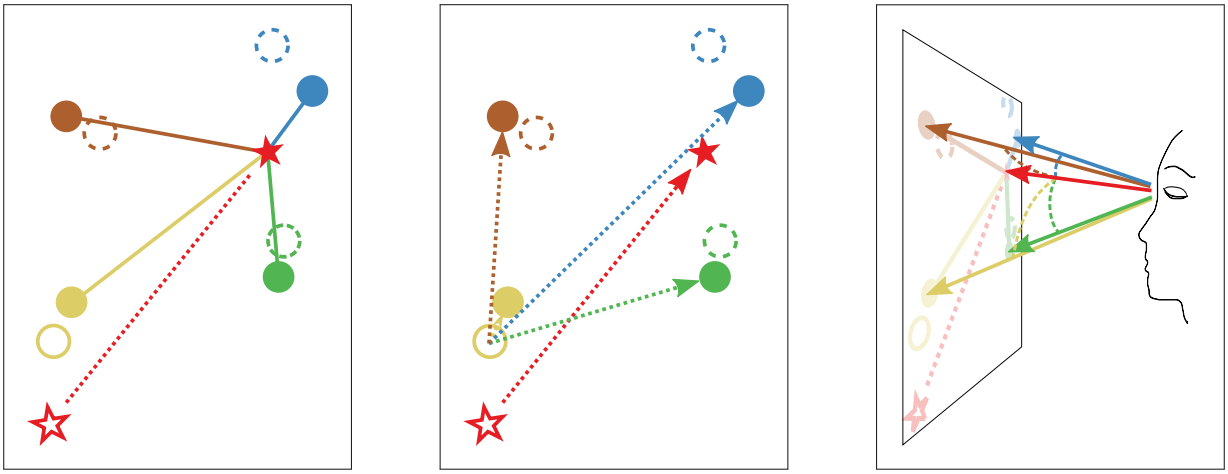
The fixation change df_t is calculated by computing the rotation axis and angle, then combining them to form a quaternion. First, the rotation *axis* $\hat{x}$ is computed as

$$\hat{x} = f_{t-1} \times f_t,$$

the cross product of the previous and the current gaze ray directions. The angle θ between them is then computed from their dot product,

$$\theta = \arccos(f_{t-1} \cdot f_t).$$

[1] If the static term were in fact constant and the filtering were performed in 2D, all of the error would be removed. However, the additional warping brought in by using angular features (see below) means there will necessarily be some residual error.



(a) Position features.

(b) Velocity features.

(c) Three-dimensional representation of position features.

Figure 2: A schematic representation of the calculated features. Colored circles represent keypoints. Filled circles represent keypoint positions at the current time. Outlined circles represent keypoint positions at the previous time, with the solid outlined circle representing the keypoint that was assigned as label; the other previous keypoints are dashed. The filled red star represents the average fixation location at the current time, and the outlined star the previous fixation location. Figure 2a represents the position features at the current time; the closest keypoint to the fixation is the blue one, but the distance is similar to the green distance due to a constant offset. Figure 2b represents the velocity features; the relative motion that the fixation would have taken between the previous time and the current time is represented by a dashed arrow for each keypoint and the observed relative motion by the dashed red arrow. The high similarity between the blue arrow and the red arrow leads to a small velocity feature for the blue keypoint independent of the constant offset. Figure 2c emphasizes that while we use simpler 2D representations for discussions, the actual feature computation is performed in 3D. Keypoints represent rays in three dimensions originating from the participant's head, as depicted by colored vectors in the image. Position features are computed from a vector metric related to the rotations between the fixation ray and each keypoint ray, as shown by the dashed lines. Velocity features (not shown) are computed by comparing the rotations between pairs of vectors with each other.

Finally, the entire quaternion difference df_t is computed from the axis-angle quaternion form

$$df_t = \cos(\theta/2) + \sin(\theta/2)\left(x_1\hat{i} + x_2\hat{j} + x_3\hat{k}\right),$$

where we use the coordinate decomposition of the rotation axis $\hat{x} = (x_1, x_2, x_3)$ as the coefficients of the quaternion unit vectors $\hat{i}$, $\hat{j}$, and $\hat{k}$. We compute the n keypoint distances similarly, where dk_t^{ij} is computed from k_{t-1}^i and k_t^j using the same procedure.

To put everything together, assume that f_{t-1} has already been labeled as keypoint i. Then the velocity features are

$$v_t^{ij} = 1 - (df_t \cdot dk_t^{ij}),$$

which are again normalized representations of the inner product between the fixation and keypoint changes over time.

4.2 Classification

To demonstrate the utility of these velocity features, we use a simple one-parameter classifier, with a weighting parameter γ to adjudicate between position and velocity features. Then, the semantic label ℓ_t for a fixation f_t is assigned as the label that yields the minimum total feature value:

$$\ell_t = \underset{j=1\cdots n}{\arg\min}\left[(1-\gamma)p_t^j + \gamma v_t^{\ell_{t-1}j}\right].$$

Here, the velocity features used are only the n features derived from the previous label, as indicated by the ℓ_{t-1} in the label. Note that all features are inherently bounded to the range $[0, 2]$, where 0 indicates complete agreement and 2 is complete disagreement; therefore, no significant weighting adjustments are required. While more sophisticated classifiers using this features can be developed, demonstrating the power of even this simpler classifier shows the usefulness of the velocity features in counteracting constant offsets.

We compare all data using three different classifiers:

Position classifier. This benchmarking classifier uses only the position features above (i.e., $\gamma = 0$). It represents the standard approach with no velocity features.

True position/velocity classifier. To demonstrate the utility of the velocity features while removing the error stackup problem caused by relying on the previous classification, we used a classifier where the velocity features were calculated from the *true* previous label rather than the assigned label. Therefore, each time step was classified in isolation. By observation, we set $\gamma = 0.8$.

Sequential position/velocity classifier. This classifier represents how these labels would be used on a real dataset where correct labels are not available. It is identical to the *true* position/velocity classifier, except that velocity features are calculated from the *assigned* label. Again, $\gamma = 0.8$.

5 SYNTHETIC DATA VALIDATION

We first validated this approach on synthetic gaze fixation data. In particular, we demonstrate that adding fixation-level velocity features indeed improves classification accuracy for a signal of this type before we add in the complication of real, noisy gaze data. Therefore, we are not especially concerned that the gaze dynamics (scanpath or fixation/saccade duration and dynamics) perfectly model human gaze; rather, we simply show that in a similar situation, the effect of constant error is mitigated. Using synthetic data additionally allows us to manually control the static error in the detection accuracy while maintaining ground-truth labels. Therefore, we can demonstrate the usefulness of velocity features in eliminating static error under controlled conditions.

5.1 Synthetic data generation

To build this data, we randomly generated keypoint and fixation data that would roughly correspond to object-focused gaze behavior. First, four keypoints were placed on an image-sized canvas. These points were initialized uniformly randomly throughout the image frame and at each (30 Hz) time step moved in a Gaussian random walk throughout the frame. A simulated gaze signal was generated by concatenating fixations of randomly specified lengths, with each fixation assigned to a uniformly randomly chosen keypoint. The fixation center was determined by averaging its corresponding keypoint positions over the duration of the fixation. To roughly simulate humanlike behavior, fixation durations were drawn from a normal distribution with mean 530 ms and standard deviation 100 ms and clipped to a minimum duration of 100 ms; these parameters match published fixation means and histograms for manipulation tasks[Hayhoe et al. 2003; Land et al. 1999]. Fixations were separated by saccade-style spacing with durations drawn from an exponential distribution with mean 100 ms. Two hundred such simulated behavior sequences were generated, each with a duration of 33.3 seconds (1000 samples).

5.2 Evaluation

We applied the three classification algorithms (as described in Sec.4.2) to this generated data. To the true fixation locations, we added a pixel offset of constant magnitude to all the data and randomly varied the offset direction per sequence. A plot of the classification accuracy versus induced offset magnitude for each algorithm appears in Fig. 3. All classifiers perform nearly perfectly when no error is present, as the fixation is much closer to its target keypoint than to any of the alternative keypoints. Once the applied offset nears the average inter-keypoint distance, though, the applied offset can cause confusion for the position features, so the position classifier drops in performance. However, the position/velocity classifiers maintain their performance for higher offset values, indicating that the velocity features successfully mitigate the effect of constant errors. Therefore, this technique is promising to pursue on a real dataset.

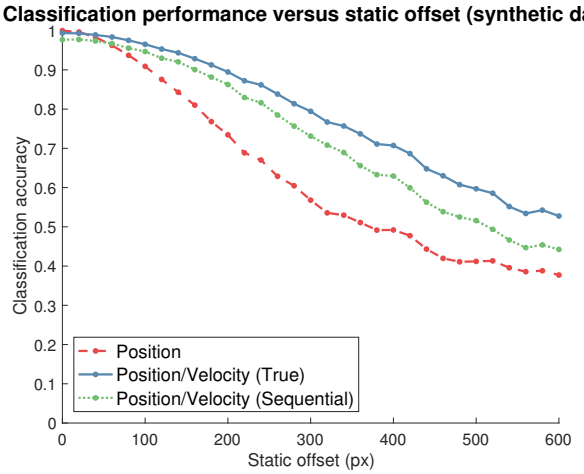


Figure 3: Plot of classification accuracy versus offset magnitude for classifiers using position features only, position plus true velocity features, and position plus estimated velocity features (see Sec. 4.2). Adding the velocity features makes the classification significantly more robust to constant offsets.

6 REAL DATA APPLICATION

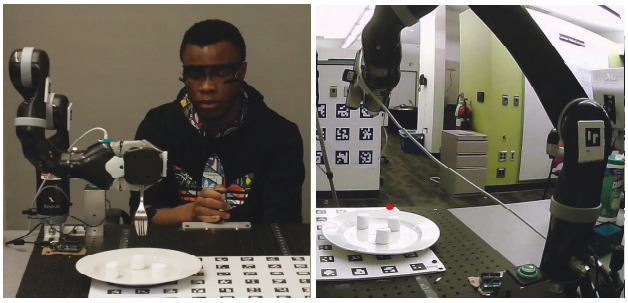
Our eventual goal is to employ this semantic gaze labeling system in real-world human-robot collaborations. We aim to infer people's intentions using their gaze fixation locations, so that a robot can autonomously take assistive action toward their intended goal. To determine the utility of our semantic gaze labeling approach in this scenario, we evaluated it on a data corpus from real-world human-robot collaborative manipulation task.

6.1 Data collection

This corpus of eye-tracking data was derived from a human participants study in which 24 people were asked to teleoperate a robot while wearing a Pupil Labs Pupil [Pupil Labs, Inc. 2017] mobile eye tracker. Each participant operated a robot manipulator arm using a joystick to pick up one of three marshmallows on a plate, in a simulation of a feeding activity (see Fig. 4). While the entire dataset contained trails in several assistive modes, for this evaluation only the data during full teleoperation was retained, which resulted in five trials per user for a total of 120 trials. Full data collection details are available in the accompanying paper [Newman et al. 2018].

6.2 Fixation detection

To ease data processing, we begin by dividing the entire eye gaze signal into time periods during which the participant is looking at the same object, which we here call *fixations*. Though the physiological details of that term are ambiguous [Hessels et al. 2018], the details of exactly when a fixation begins or ends is not especially relevant to our algorithm. Furthermore, the distinction between



(a) Forward view. (b) Egocentric view.

Figure 4: Task-oriented eye gaze data was collected from participants who were asked to teleoperate a robot arm to complete a food acquisition task. (a) A forward view of a participant wearing a Pupil Labs Pupil[Pupil Labs, Inc. 2017] eye tracker. (b) A frame of the egocentric video captured by the eye tracker, with a red dot indicating the participant's point of regard is on (or near) the plate.

fixations on static or dynamic objects (often termed *fixations* and *smooth pursuits* respectively) can be collapsed. With those caveats in mind, for fixation segmentation we use the I-BDT algorithm as described in Tafaj et al. [2012], but without the second step for pursuit classification. This algorithm assigns a label of either *sac* or *fix* to each time step based on a two-component Bayesian mixture model, then fuses sequential time steps labeled *fix* into a single fixation. We re-implemented the algorithm in Python (originally in Matlab) and our implementation is available online. [2]

To validate our re-implementation of this algorithm, we compared it to the manually annotated dataset published in Kasneci et al. [2014]. The overall precision for individually labeling time steps as *sac* was 0.725, and the recall 0.921. These results indicate that our re-implementation is somewhat more conservative than the original algorithm: it successfully labels most of the original *sac* time steps (as indicated by the high recall) but also labels additional points as *sac* (as indicated by the relatively low precision). However, this conservatism is compatible with our overall goals: it is more important to successfully subdivide time periods in which different objects are being looked at than it is to ensure that the entire gaze time is contained within a single fixation. In other words, we more reliably detect transitions between objects (as saccades) at the cost of occasionally splitting up a single fixation into multiple parts.

6.3 Semantic labeling

To determine the position of relevant objects, we developed a direct pipeline for explicitly locating the objects of interest as keypoints within the egocentric video. First, we calculate the extrinsic parameters of the camera relative to a constant frame of reference in the scene. Next, we note that the robot and object positions relative to this fixed frame are already available, as they are necessary for computing the manipulation action of interest. Therefore, we can transform the relevant object locations into the egocentric camera frame, and then project them into the image frame.

[2]https://github.com/HARPLab/ibmmpy

To complete this pipeline, the main difficulty is in obtaining the camera extrinsics. To simplify this problem, we use a pre-existing grid of ArUcO tags [Garrido-Jurado et al. 2014] available in the workspace, as well as a prior calibration between the ArUcO grid and the robot base[Wu and Ren 2017]. The extrinsics were then smoothed using a Kalman filter using the `robot_localization` ROS package[Moore and Stouch 2014]. One extension to this work is to consider alternate methods of extrinsic calibration; using monocular SLAM for gaze data has shown some success [Paletta et al. 2013b; Wang et al. 2018]. However, since in this situation the tag grid is already present, more advanced techniques were not necessary.

Fixations were manually labeled with their associated keypoints as determined from watching the egocentric camera. Coders were instructed to label each fixation with the scene keypoint most similar, using clues such as proximity, history of motions, or motion relative to the background scene. Four coders coded the data. Twelve trials (10%) were coded by all coders, and the resulting average pairwise Cohen's kappa (inter-reliability rating) was 0.645, indicating good agreement [Hallgren 2012]. A sample image from the egocentric video, with keypoints, fixation, and feature values overlaid, appears in Fig. 5.

For computing the features, the 2D (pixel) coordinates for all of the relevant positions (fixations and keypoints) were first rectified using to a fisheye camera distortion model which was fit from a static calibration. Then, the undistorted pixel coordinates were transformed into 3D (ray) coordinates using the egocentric camera intrinsic matrix K:

$$\begin{bmatrix} r_x \\ r_y \\ 1 \end{bmatrix} = K^{-1} \begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix}.$$

All feature computation was performed using these ray coordinates.

6.4 Results

Each of the algorithms described in Sec. 4.2 were applied to the labeled data, which consisted of 3861 total labeled fixations. The position classifier yielded an overall accuracy of 65.9%, the true position/velocity an accuracy of 75.8%, and the sequential position/velocity classifier an accuracy of 65.5%. Therefore, adding in the velocity features has the potential to increase the labeling accuracy by ten percentage points; however, the error stackup induced by the feature dependence in the sequential velocity case eliminates the added benefit of the velocity features.

To understand the performance more deeply, we plot the accuracy of each classification algorithm relative to the actual offset of the true label. That is, for each fixation, we compute the distance between the fixation point and the keypoint assigned by the manual labeler. These distances are binned in 0.6° bins (each bin having a minimum of 20 fixations) and the resulting accuracies are show in Fig. 6. As in the synthetic data, the position classifier performs best at very low offsets. When the offsets exceed 5°, the true position/velocity classifier surpasses the position classifier, and the sequential position/velocity classifier does the same at around 7° of offset. These results demonstrate that the velocity features indeed improve the classification performance in the presence of error. Though they may not be necessary for highly accurate datasets,

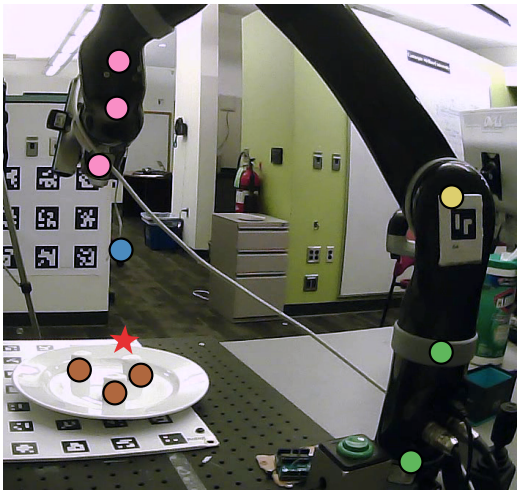


Figure 5: Frame of the egocentric video. The fixations (shown as a red star) are manually classified as one of eleven subcategories, consisting of each of the manipulator joints as well as the goal location. These lower-level keypoints are then grouped into categories (*robot base*, *robot arm*, *robot end effector*, *fork tip*, and *goal morsels*) for representative labeling; categories are indicated by color. All classification is done by keypoint category, where the higher-level keypoint location is determined by the mean locations of its component lower-level keypoints.

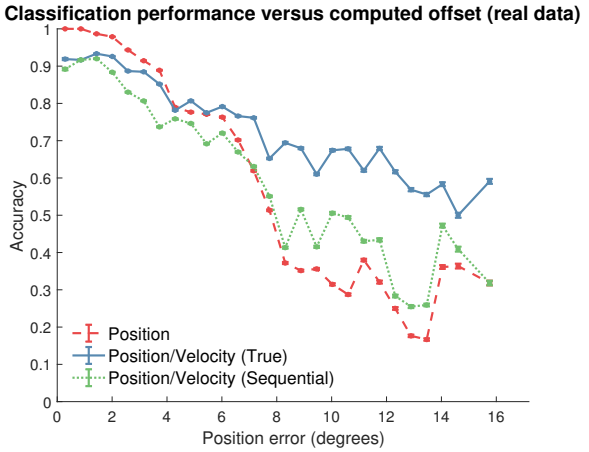


Figure 6: Plot of classification accuracy versus computed degree offset on real data. Colored lines indicate classification approaches (position features only, red; position plus true velocity features, blue; and position plus sequential velocity features, green.) Using velocity features made the classifications more accurate when the error exceeds 7°.

using velocity features for classification improves accuracy in the presence of tracking error.

7 DISCUSSION AND CONCLUSIONS

In this work, we present the idea of using velocity features to improve the quality of semantic gaze labeling in the presence of errors. To demonstrate their effectiveness, we use a simple classifier based on these features on both synthetic data and real data collected during a manipulation task. In both cases, the addition of these velocity features significantly improves the quality of the labeling process. While more sophisticated, data-intensive approaches may yield better results, adding this relatively simple additional feature category can mitigate the effect of systemic errors.

A simple technique that can mitigate errors in 3D gaze analysis is particularly valuable for collecting real-world data in general environments. While lab conditions allow for rigorous calibration procedures and frequent resets to ensure high accuracy, that standard is difficult to achieve with untrained users or in natural situations. Even if high accuracy were achievable, marginal gains in accuracy comparable to those enabled by these velocity features may require substantial resources in computation, training, or calibration to achieve elsewhere in the gaze tracking system. In contrast, using a relatively simple system to process this gaze data reduces the accuracy requirements of the underlying tracker, enabling even low-cost systems to be used for complicated tasks.

One challenge of the approach presented here is that it requires a finite set of enumerable locations in the world that the user may be looking at. In certain tasks, such as a simple manipulation action when only a few objects are relevant, this assumption is likely to be valid. Moreover, in a scenario such as robotic manipulation in which other parts of the tasks require knowledge of object positions, these keypoints may already be present in the system. In addition, a labeling strategy using these features that gives a score can be thresholded to detect gaze events targeted at non-tracked objects, which may be inherently interesting (for example, they may represent a task failure or an unexpected state). For other tasks, such as recognizing novel objects during driving or understanding reading behavior, a small set of keypoints may not be relevant, so alternate approaches (like automated object recognition [Auepanwiriyakul et al. 2018; Orlov et al. 2018]) may need to be used. Even in these cases, however, applying the velocity-based classification idea to reduce slow or constant errors may improve performance. Furthermore, an extension of this work could consider automatically learning relevant keypoints from visual saliency or object recognition and feature clustering; for such a project, velocity features can reduce the impact of not just calibration error but slow frame or object motions.

Another assumption made by this approach is that a system can accurately divide the gaze into periods of time focused on a single semantic unit (here called fixations) and that the object is relatively stationary during that time. If the objects move substantially during a fixation, the features can wander. This problem may be ameliorated by deliberately considering the internal motion and adding additional matching features indicating the degree of correlation of the internal motion of a keypoint with the internal motion of the object, as has been explored elsewhere [Bernhard et al. 2014; Vidal et al. 2013].

One limitation to consider is, as discussed earlier, the effect of error stackup: when computing the velocity features at some specific

time t, if the label for the previous timestep ℓ_{t-1} was computed incorrectly, all of the velocity comparisons are incorrect. In this work, we chose to fuse the position and velocity features at each timestamp, which provides a continuous corrective signal. However, numerous other approaches are available. For example, a system could periodically discard the velocity labels and reset using only the position labels. Alternatively, if an expensive but more accurate checking process (such as asking the user to look at a known point) is available, this sensor can be used to periodically reset the labels.

One direction for improving classification results that explicitly addresses the problem of error stackup is to combine the features presented here with more sophisticated, learning-based classifiers. For example, Bernhard et al. [2014] presents a framework for combining multiple features using Bayesian probability metrics, and Mantiuk et al. [2013] describes using these probability updates in a hidden Markov model. We have been careful to present the contribution in this paper as velocity *features*; the classification algorithms described in Sec. 4.2 are deliberately simplified to show the power of the features in isolation. Future work will explore combining these velocity features with sequential labeling techniques like hidden Markov models, as well as combining them with other featurizations for better overall recognition.

In this work, we presented a set of features to use to better solve the semantic gaze recognition problem, especially when considering the challenges that appear when using 3-dimensional, real-world gaze. Accurately solving this problem is one important component to using eye gaze for real-world systems.

ACKNOWLEDGMENTS

This work was supported by the Paralyzed Veterans of America and the National Science Foundation (IIS-1755823). Thanks to Siddharth Girdhar, Michael Huang, and Roman A. Kaufman for assistance in data labeling; to Robbie Paolini for assistance and code for robot calibration; and to the anonymous reviewers whose feedback substantially improved this paper.

REFERENCES

Reuben M. Aronson and Henny Admoni. 2018. Gaze for Error Detection During Human-Robot Shared Manipulation. In *Fundamentals of Joint Action workshop, Robotics: Science and Systems*.

Reuben M. Aronson, Thiago Santini, Thomas. C. Kübler, Enkelejda Kasneci, Siddhartha Srinivasa, and Henny Admoni. 2018. Eye-Hand Behavior in Human-Robot Shared Manipulation. In *ACM/IEEE International Conference on Human-Robot Interaction*.

Rowel Atienza and Alexander Zelinsky. 2005. Intuitive Human-Robot Interaction Through Active 3D Gaze Tracking. In *Robotics Research. The Eleventh International Symposium*, Paolo Dario and Raja Chatila (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 172–181.

Chaiyawan Auepanwiriyakul, Alex Harston, Pavel Orlov, Ali Shafti, and A. Aldo Faisal. 2018. Semantic Fovea: Real-time Annotation of Ego-centric Videos with Gaze Context. In *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications (ETRA '18)*. ACM, New York, NY, USA, Article 87, 3 pages. https://doi.org/10.1145/3204493.3208349

Matthias Bernhard, Efstathios Stavrakis, Michael Hecher, and Michael Wimmer. 2014. Gaze-to-Object Mapping During Visual Search in 3D Virtual Environments. *ACM Trans. Appl. Percept.* 11, 3, Article 14 (Aug. 2014), 17 pages. https://doi.org/10.1145/2644812

C. Braunagel, E. Kasneci, W. Stolzmann, and W. Rosenstiel. 2015. Driver-activity recognition in the context of conditionally autonomous driving. In *IEEE 18th International Conference on Intelligent Transportation Systems*. IEEE, 1652–1657.

Anne-Marie Brouwer and David C. Knill. 2007. The role of memory in visually guided reaching. *Journal of Vision* 7, 5 (06 2007), 6–6. https://doi.org/10.1167/7.5.6

S. Garrido-Jurado, R. Muñoz Salinas, F.J. Madrid-Cuevas, and M.J. Marín-Jiménez. 2014. Automatic Generation and Detection of Highly Reliable Fiducial Markers Under Occlusion. *Pattern Recogn.* 47, 6 (June 2014), 2280–2292. https://doi.org/10.1016/j.patcog.2014.01.005

Kakeru Hagihara, Keiichiro Taniguchi, Irshad Abibouraguimane, Yuta Itoh, Keita Higuchi, Jiu Otsuka, Maki Sugimoto, and Yoichi Sato. 2018. Object-wise 3D Gaze Mapping in Physical Workspace. In *Proceedings of the 9th Augmented Human International Conference (AH '18)*. ACM, New York, NY, USA, Article 25, 5 pages. https://doi.org/10.1145/3174910.3174921

Kevin A Hallgren. 2012. Computing Inter-Rater Reliability for Observational Data: An Overview and Tutorial. *Tutorials in quantitative methods for psychology* 8, 1 (2012), 23–34. http://www.ncbi.nlm.nih.gov/pubmed/22833776

Mary Hayhoe and Dana Ballard. 2005. Eye movements in natural behavior. *Trends in Cognitive Sciences* 9, 4 (2005), 188–194. https://doi.org/10.1016/j.tics.2005.02.009

Mary M. Hayhoe, Anurag Shrivastava, Ryan Mruczek, and Jeff B. Pelz. 2003. Visual memory and motor planning in a natural task. *Journal of Vision* 3, 1 (02 2003), 6–6. https://doi.org/10.1167/3.1.6

Laura Herlant, Rachel Holladay, and Siddhartha Srinivasa. 2016. Assistive Teleoperation of Robot Arms via Automatic Time-Optimal Mode Switching. In *ACM/IEEE International Conference on Human-Robot Interaction*.

Roy S. Hessels, Diederick C. Niehorster, Marcus Nyström, Richard Andersson, and Ignace T. C. Hooge. 2018. Is the eye-movement field confused about fixations and saccades? A survey among 124 researchers. *Royal Society Open Science* 5, 8 (aug 2018), 180502. https://doi.org/10.1098/rsos.180502

Roland S Johansson, Gö Ran Westling, Anders Bäckström, and J Randall Flanagan. 2001. Eye–Hand Coordination in Object Manipulation. *The Journal of Neuroscience* 21, 17 (2001), 6917–6932.

Marcel A. Just and Patricia A. Carpenter. 1980. A theory of reading: From eye fixations to comprehension. (1980), 329–354.

Enkelejda Kasneci, Gjergji Kasneci, Thomas C. Kübler, and Wolfgang Rosenstiel. 2014. The applicability of probabilistic methods to the online recognition of fixations and saccades in dynamic scenes. In *Proceedings of the Symposium on Eye Tracking Research and Applications - ETRA '14*. ACM Press, New York, New York, USA, 323–326. https://doi.org/10.1145/2578153.2578213

Michael Land, Neil Mennie, and Jennifer Rusted. 1999. The Roles of Vision and Eye Movements in the Control of Activities of Daily Living. *Perception* 28, 11 (1999), 1311–1328. https://doi.org/10.1068/p2935 PMID: 10755142.

Michael F. Land and Mary Hayhoe. 2001. In what ways do eye movements contribute to everyday activities? *Vision Research* 41, 25 (2001), 3559–3565.

S. Li, X. Zhang, and J. D. Webb. 2017. 3-D-Gaze-Based Robotic Grasping Through Mimicking Human Visuomotor Function for People With Motion Impairments. *IEEE Transactions on Biomedical Engineering* 64, 12 (Dec 2017), 2824–2835. https://doi.org/10.1109/TBME.2017.2677902

R. Mantiuk, B. Bazyluk, and R. K. Mantiuk. 2013. Gaze-driven Object Tracking for Real Time Rendering. *Computer Graphics Forum* 32, 2pt2 (2013), 163–173. https://doi.org/10.1111/cgf.12036

Neil Mennie, Mary Hayhoe, and Brian Sullivan. 2007. Look-ahead fixations: anticipatory eye movements in natural tasks. *Experimental Brain Research* 179, 3 (01 May 2007), 427–442. https://doi.org/10.1007/s00221-006-0804-0

T. Moore and D. Stouch. 2014. A Generalized Extended Kalman Filter Implementation for the Robot Operating System. In *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)*. Springer.

Benjamin A. Newman, Reuben M. Aronson, Siddhartha S. Srinivasa, Kris Kitani, and Henny Admoni. 2018. HARMONIC: A Multimodal Dataset of Assistive Human-Robot Collaboration. *ArXiv e-prints* (July 2018). arXiv:cs.RO/1807.11154

Pavel Orlov, Ali Shafti, Chaiyawan Auepanwiriyakul, Noyan Songur, and A. Aldo Faisal. 2018. A Gaze-contingent Intention Decoding Engine for Human Augmentation. In *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications (ETRA '18)*. ACM, New York, NY, USA, Article 91, 3 pages. https://doi.org/10.1145/3204493.3208350

Lucas Paletta, Katrin Santner, Gerald Fritz, Albert Hofmann, Gerald Lodron, Georg Thallinger, and Heinz Mayer. 2013a. FACTS - A Computer Vision System for 3D Recovery and Semantic Mapping of Human Factors. In *Computer Vision Systems*, Mei Chen, Bastian Leibe, and Bernd Neumann (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 62–72.

Lucas Paletta, Katrin Santner, Gerald Fritz, Heinz Mayer, and Johann Schrammel. 2013b. 3D Attention: Measurement of Visual Saliency Using Eye Tracking Glasses. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems (CHI EA '13)*. ACM, New York, NY, USA, 199–204. https://doi.org/10.1145/2468356.2468393

Thies Pfeiffer. 2012. Measuring and Visualizing Attention in Space with 3D Attention Volumes. In *Proceedings of the Symposium on Eye Tracking Research and Applications (ETRA '12)*. ACM, New York, NY, USA, 29–36. https://doi.org/10.1145/2168556.2168560

Thies Pfeiffer and Patrick Renner. 2014. EyeSee3D: A Low-cost Approach for Analyzing Mobile 3D Eye Tracking Data Using Computer Vision and Augmented Reality Technology. In *Proceedings of the Symposium on Eye Tracking Research and Applications (ETRA '14)*. ACM, New York, NY, USA, 369–376. https://doi.org/10.1145/2578153.2628814

Thies Pfeiffer, Patrick Renner, and Nadine Pfeiffer-Leßmann. 2016. EyeSee3D 2.0: Model-based Real-time Analysis of Mobile Eye-tracking in Static and Dynamic Three-dimensional Scenes. In *Proceedings of the Ninth Biennial ACM Symposium*

on Eye Tracking Research & Applications (ETRA '16). ACM, New York, NY, USA, 189–196. https://doi.org/10.1145/2857491.2857532

Pupil Labs, Inc. 2017. Pupil Labs - Pupil. Retrieved Jan 5, 2018 from https://pupil-labs.com/pupil/

Karishma Singh, Mahmoud Kalash, and Neil Bruce. 2018. Capturing Real-world Gaze Behaviour: Live and Unplugged. In *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications (ETRA '18)*. ACM, New York, NY, USA, Article 20, 9 pages. https://doi.org/10.1145/3204493.3204528

Enkelejda Tafaj, Gjergji Kasneci, Wolfgang Rosenstiel, and Martin Bogdan. 2012. Bayesian Online Clustering of Eye Movement Data. In *Proceedings of the Symposium on Eye Tracking Research and Applications (ETRA '12)*. ACM, New York, NY, USA, 285–288. https://doi.org/10.1145/2168556.2168617

Mélodie Vidal, Andreas Bulling, and Hans Gellersen. 2013. Pursuits: Spontaneous Interaction with Displays Based on Smooth Pursuit Eye Movement and Moving Targets. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '13)*. ACM, New York, NY, USA, 439–448. https://doi.org/10.1145/2493432.2493477

Haofei Wang, Jimin Pi, Tong Qin, Shaojie Shen, and Bertram E. Shi. 2018. SLAM-based Localization of 3D Gaze Using a Mobile Eye Tracker. In *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications (ETRA '18)*. ACM, New York, NY, USA, Article 65, 5 pages. https://doi.org/10.1145/3204493.3204584

L. Wu and H. Ren. 2017. Finding the Kinematic Base Frame of a Robot by Hand-Eye Calibration Using 3D Position Data. *IEEE Transactions on Automation Science and Engineering* 14, 1 (Jan 2017), 314–324. https://doi.org/10.1109/TASE.2016.2517674