Active Motion-Based Communication for Robots with Monocular Vision

Haruki Nishimura and Mac Schwager

***Abstract*— In this paper, we consider motion as a means of sending messages between robots. We focus on a scenario in which a message is encoded in a sending robot's trajectory, and decoded by a receiver robot equipped with a monocular camera. The relative pose between the robots is unknown. We introduce an online Bayesian estimation algorithm based on the Multi-hypothesis Extended Kalman Filter for the receiving robot to simultaneously estimate its relative pose to the sender, and the trajectory class of the sender. The difficulty in this problem arises from the monocular vision model of the receiver and the unknown relative pose between robots, which brings inherent ambiguity into the trajectory identification, and hence the message decoding. An active vision-based control policy is derived and combined with the Bayesian estimation in order to deal with this difficulty. The policy is constructed online based on Monte Carlo Tree Search and aims at reducing the entropy over the trajectory class distribution. The algorithm has broad applications, e.g., to intent modeling and motion prediction for autonomous driving and autonomous drone operations. Simulation results demonstrate that the proposed estimation algorithm and the control policy result in an accurate trajectory classification.**

I. Introduction

Robots that interact with one another are often assumed to communicate over a wireless network. However, in many instances a wireless network is not available, or cannot be relied upon. In this paper we consider motion-based communication, in which a sender robot encodes a message in its trajectory, which is decoded by a receiver robot. The receiving robot has only a monocular camera with which to observe the motion, and the relative pose between the two robots is unknown. This can make the identification of the trajectory difficult or impossible without an active perception strategy. We present a Bayesian estimation algorithm by which the receiving robot classifies the trajectory from the sending robot, and simultaneously estimates the relative pose between the two robots. We also propose an active perception algorithm based on Monte Carlo Tree Search with Double Progressive Widening. The search is performed in the belief space and drives the receiving robot to move so that it can disambiguate between similar trajectory classes. We use entropy as the measure of uncertainty in the trajectory class distribution. We present theoretical results as well as experimental results evaluated in a realistic simulation environment.

This work was supported in part by the Toyota-SAIL Center for AI Research, ONR grant N00014-16-1-2787, NSF grant CMMI-1562335, and JASSO. We are grateful for this support.

The authors are with the Department of Aeronautics and Astronautics at Stanford University, Stanford, CA 94305, USA. Email: {hnishimura, schwager}@stanford.edu.

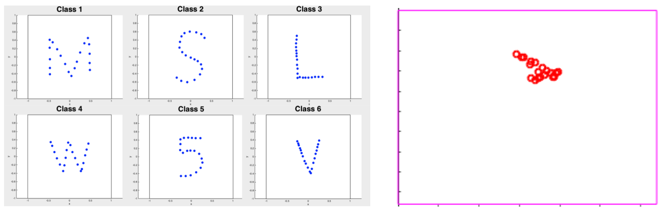


Fig. 1. (Left) A sample 2D trajectory codebook with potential ambiguities between entries. (Right) A noisy camera projection of a complete trajectory. The trajectory may be either an "L" or a "V" from the image. Further inference requires the receiver to move to obtain a more accurate estimate.

Our estimation and control algorithm may be used for robots to literally exchange messages through their trajectories, or more generally, as an abstraction for motion classification and prediction. The sender (whether intentionally or not) sends a "message" with its trajectory, which the receiving robot must "decode." For example, an autonomous car must categorize an observed vehicle as aggressive or defensive based on its observed trajectory. Thus our algorithm has applications to autonomous driving, where vehicles have to predict the motion of other vehicles, pedestrians, and bikers by observing their motion, and to autonomous drones, which have to avoid collisions with other drones and pedestrians in their airspace by observing and predicting their trajectories. The algorithm may also model motion-based communication in animals, for example the "waggle dance" of honey bees.

Previous work [1], [2] has employed control theoretic approaches for the *sending* robot to design a set of distinguishable and energy-optimal trajectories. In constrast, here we consider algorithms for the *receiver* to decode the message. This problem is particularly complicated by the monocular camera of the receiver (which gives only a 2D projection of the sender's trajectory) and the unknown relative pose between the two robots. In this scenario, two different trajectories can look identical from the point of view of the receiver, e.g., an "L" trajectory looks like a "V" trajectory from an oblique angle (see Fig. 1). Although recent work [1], [2], [3], [4], [5] has explored the emerging field of motion-based communication, an effective way to resolve the ambiguity inherent due to the monocular vision and unknown pose has not yet been addressed to the authors' knowledge.

More specifically, the main contributions of this study are two fold. First, we formulate an online estimation framework in which both the sender's message and the receiver's relative pose are sequentially estimated. Based on the Markov assumption, we provide a recursive Bayesian estimation algorithm to handle the multimodal distribution over the joint state of the receiver and the sender. A Gaussian approximation and a model linearization yield a Gaussian mixture representation of the robots' joint state, naturally leading to the Multi-hypothesis Extended Kalman Filter algorithm. Second, we actively control the receiver to move around the sender in order to disambiguate between similar trajectory hypotheses. An information theoretic approach is taken in which we control the receiver to maximally decrease the entropy of the trajectory class distribution. Furthermore, we plan over multiple future poses of the receiver by formulating the optimal control problem as a belief-state Markov deision process and employing the Monte Carlo Tree Search algorithm with Double Progressive Widening. The performance of this control policy is compared to that of a greedy Monte Carlo algorithm and random walk in simulations.

Our algorithms draw inspiration from several existing works in the areas of state estimation, computer vision and planning under uncertainty. For example, general nonlinear filtering with a Gaussian mixture model is introduced by Alspach and Sorenson in [6]. Jochmann et al. discuss its applicability to robot localization [7]. Chen and Liu [8] introduce mixture Kalman filters to handle conditional dynamic linear models. Simultaneous active vision classification and camera pose estimation is proposed by Sipe and Casasent in [9]. Although their estimation scheme has some similarities with ours, it does not deal with motion-based communication, and it does not use an entropy-based control objective. Denzler et al. [10] and Zobel et al. [11] present an entropy-based camera control framework, and derive a closed-form control objective for Gaussian distributions. Seekircher et al. [12] employ Monte Carlo exploration to implement an entropy-based active vision control. These works also do not address communication through motion, but active vision. In terms of the planning algorithms, Monte Carlo Tree Search is an online method for finding optimal decisions [13] and is widely used in various decision processes that involve uncertainty, ranging from game-playing [14] to robotics [15]. The algorithm is anytime, and can be also applied to continuous or large state and action spaces by introducing Double Progressive Widening [16]. There are several existing works that apply Monte Carlo Tree Search to active perception [15], [17]. Our work has similar goals, but we are particularly interested in applications in motion-based communication.

The rest of the paper is organized as follows. In Sec. II we define the robot models and formalize the problem. In Sec. III we derive the recursive Bayesian update formula and provide the Multi-hypothesis Extended Kalman Filter algorithm. A belief initialization algorithm is discussed in Sec. IV. The active-vison control policy for the receiver is formulated in Sec. V. Simulation results in a ROS-Gazebo environment are presented in Sec. VI and conclusions with future directions are given in Sec. VII.

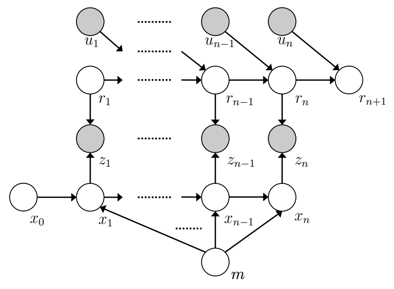


Fig. 2. Bayesian network structure for our trajectory classification and pose estimation problem. The receiver robot observes the camera image z of the sender's position x to estimate the relative pose r and the sender's message m, and applies control u.

II. Preliminaries

The sender robot chooses a pre-specified trajectory to encode a message. The sender then executes this trajectory, while the receiver observes the motion and controls its viewing positions. In this work we let the receiver move and estimate the message simultaneously while the sender executes a single trajectory.

This leads to a sequential state estimation and decision making problem represented by the Bayesian network structure in Fig. 2. We formalize this stochastic process in what follows, dropping the time index for convenience when it does not cause confusion.

A. Trajectory generation

A trajectory of the sender is represented by an initial position $x_0 \in \mathbb{R}^3$ and a set of n stochastic transitions $\{x_k \mid x_{k+1} = f_k(x_k, m) + v_k,\ k = 0, \dots, n-1\}$, where f_k is a deterministic, time-indexed state transition function of the sender, m is a message or a trajectory class and v_k is zero-mean Gaussian white noise. The initial position x_0 is also assumed to be Gaussian distributed. Given a trajectory class $m \in \{1, \dots, M\}$, the knowledge of the corresponding *noiseless* trajectory can be completely described by a set of $n+1$ waypoints $\{\tilde{x}_0, \tilde{x}_1^{(m)}, \dots, \tilde{x}_n^{(m)}\}$. This is expressed in a frame of reference named the trajectory frame and is fixed in inertial space. We assume that the receiver has already acquired the set of ideal trajectories as a "trajectory codebook." This can be done through modeling or learning the deterministic transition function f_k of the sending agent, and is an interesting topic for future research. We also assume that the receiver can identify the time when the sender initiates the trajectory execution.

B. State transition model

Let $s_k = (r_k^{\mathrm{T}}, x_k^{\mathrm{T}})^{\mathrm{T}} \in \mathbb{R}^9$ be the joint robot state of the receiver and the sender at timestep k. Conditional on a trajectory class m, the state transition for the sender is as described in the previous section. The receiver's pose r specifies its relative position and attitude in the trajectory frame

$$r = (\omega^{\mathrm{T}}, t^{\mathrm{T}})^{\mathrm{T}} \in \mathbb{R}^6, \tag{1}$$

where $\omega = (\omega_x, \omega_y, \omega_z)^{\mathrm{T}} \in \mathbb{R}^3$ gives the angle-axis representation of $SO(3)$. The direction of ω specifies the axis around which the receiver's body frame is rotated with respect to the trajectory frame, and the ℓ_2 norm $\|\omega\|$ gives the magnitude of the rotation angle. Similarly, $t = (t_x, t_y, t_z)^{\mathrm{T}} \in \mathbb{R}^3$ represents the translation of the receiver's body in the trajectory frame. The pose r is not known to the receiver or the sender.

The new position t_{k+1} is given by the current position t_k, attitude ω_k and the translational control input $u_{trans} \in \mathbb{R}^3$,

$$t_{k+1} = t_k + \exp([\omega_k]_\times) u_{trans}, \tag{2}$$

where $[\cdot]_\times : \mathbb{R}^3 \mapsto \mathfrak{so}(3)$ is the skew-symmetric matrix operator

$$[\omega]_\times = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix}. \tag{3}$$

The matrix exponential in (2) can be evaluated using the Rodrigues formula,

$$\exp([\omega]_\times) = I_{3\times3} + \frac{\sin\theta}{\theta}[\omega]_\times + \frac{(1-\cos\theta)}{\theta^2}[\omega]_\times^2, \tag{4}$$

where $\theta = \|\omega\|$.

The new attitude ω_{k+1} is related to the current attitude ω_k and the rotational control input $u_{rot} \in \mathbb{R}^3$ as follows,

$$\exp([\omega_{k+1}]_\times) = \exp([\omega_k]_\times)\exp([u_{rot}]_\times). \tag{5}$$

The concatenated control input $u_k = (u_{rot}^{\mathrm{T}}, u_{trans}^{\mathrm{T}})^{\mathrm{T}} \in \mathbb{R}^6$ is represented in the body coordinates of the receiver.

Finally, the state transition for the receiver is also corrupted by zero-mean Gaussian white noise. Along with the state transition noise for the sender, we denote $Q \in {\mathbb{R}_+}^{9\times9}$ as the covariance matrix for the joint state transition.

Note that the receiver will apply an input u_k only after the sender has moved from x_{k-1} to x_k, and the state transition to r_{k+1} is assumed to be finished by the time the sender reaches the next position x_{k+1}.

C. Observation model

The monocular vision of the receiver can be approximated by a simple pinhole camera model. We will provide a brief review of the model that relates a sender's position to an observation. Interested readers are referred to [18], [19] for further details.

1) Pinhole camera projection: Let $z \in \mathbb{R}^2$ represent the observation expressed in the pixel coordinates on the image plane and $x \in \mathbb{R}^3$ be the Cartesian coordinates representing the sender's position. The relationship between a 3D point x and the corresponding 2D point z is given by the following formula

$$\lambda \begin{pmatrix} z \\ 1 \end{pmatrix} = P \begin{pmatrix} x \\ 1 \end{pmatrix}, \tag{6}$$

where λ is a scale factor and $P \in \mathbb{R}^{3\times4}$ is the projection matrix. P can be further decomposed into a product of the camera calibration matrix $K \in \mathbb{R}^{3\times3}$ and the external parameters matrix $T \in \mathbb{R}^{3\times4}$. The camera calibration matrix contains intrinsic parameters of the camera and we assume that it is known to the receiver.

2) External parameters matrix: The external parameters matrix defines the rotation and the translation of the camera frame with respect to the trajectory frame,

$$T = \left(R^c_{traj}\ t^c_{traj}\right), \tag{7}$$

where t^c_{traj} is the translational vector and R^c_{traj} is the 3D rotation matrix represented in the camera coordinates. These parameters can be expressed in terms of the receiver's state $r = (\omega^{\mathrm{T}}, t^{\mathrm{T}})^{\mathrm{T}}$ as follows,

$$R^c_{traj} = R_c^{b\mathrm{T}}\ [\exp([\omega]_\times)]^{\mathrm{T}} \tag{8}$$

$$t^c_{traj} = -R_c^{b\mathrm{T}}\ [\exp([\omega]_\times)]^{\mathrm{T}}\ t - R_c^{b\mathrm{T}}\ t_c^b, \tag{9}$$

where R_c^b and t_c^b define the relative pose between the body and the camera coordinates and are determined by the physical position and attitude of the camera on the receiver robot.

The pinhole camera projection (6) along with the external parameters matrix (7) comprise the nonlinear observation function $h(s)$. The actual observation is also subject to zero-mean Gaussian white noise with covariance $R \in {\mathbb{R}_+}^{2\times2}$, hence the distribution is given by

$$p(z \mid s) = \mathcal{N}(z; h(s), R). \tag{10}$$

D. Repeated trajectory execution

The stochastic process depicted in Fig. 2 represents a single execution of a trajectory. In fact, the process is repeated multiple times, with r_{n+1} in the current iteration being r_1 in the next iteration. Similarly, the sender transitions from x_n to x_1 through x_0. Note that the posterior at the end of the current iteration becomes the prior for the next iteration.

III. Bayesian Filtering

As illustrated in Fig. 2, the trajectory class m and the receiver's state r are correlated via the observation z and the sender's position x. We therefore need to simultaneously estimate m and $s = (r^{\mathrm{T}}, x^{\mathrm{T}})^{\mathrm{T}}$ even though eventually we are more interested in m than s in this problem. Formally, this is equivalent to describing the joint distribution of s and m in the Bayesian framework, which is discussed in this section.

A. Recursive Bayesian Estimation Formula

We are now set to derive the recursive Bayesian estimation formula to determine the posterior distribution over the joint robot state s and the trajectory class m.

Theorem 1 (Recursive Bayesian Update): The posterior of the joint distribution $p(s_{k+1}, m \mid z_{1:k+1}, u_{1:k})$ can be obtained recursively as a function of the class-dependent state transition model $p(s_{k+1} \mid s_k, u_k, m)$ and the observation model $p(z_{k+1} \mid s_{k+1})$, given the Bayesian network structure depicted in Fig. 2 and the prior belief $p(s_k, m \mid z_{1:k}, u_{1:k-1})$.

Proof: From the definition of conditional probability, the posterior can be factored into a product of a continuous probability density function p of the joint state s and a discrete probability mass function P of the trajectory class m

$$p(s_{k+1}, m \mid z_{1:k+1}, u_{1:k}) = p(s_{k+1} \mid z_{1:k+1}, u_{1:k}, m) \times P(m \mid z_{1:k+1}, u_{1:k}) \quad . \tag{11}$$

The first term can be decomposed using the Bayes' rule

$$p(s_{k+1} \mid z_{1:k+1}, u_{1:k}, m) \propto p(z_{k+1} \mid s_{k+1}) \times p(s_{k+1} \mid z_{1:k}, u_{1:k}, m) \quad . \tag{12}$$

The first term $p(z_{k+1} \mid s_{k+1})$ in (12) is the observation model. The second term in (12) can be expressed as a marginalization integral with the class-dependent state transition model $p(s_{k+1} \mid s_k, u_k, m) = p(r_{k+1} \mid r_k, u_k)p(x_{k+1} \mid x_k, m)$

$$p(s_{k+1} \mid z_{1:k}, u_{1:k}, m) = \int p(s_{k+1} \mid s_k, u_k, m) \times p(s_k \mid z_{1:k}, u_{1:k-1}, m) ds_k \quad . \tag{13}$$

Substituting (13) into (12), we see that the first term of the factored posterior in (11) can be specified by the observation model, the state transition model and $p(s_k \mid z_{1:k}, u_{1:k-1}, m)$, which is part of the prior belief factored in the same manner as in (11).

On the other hand, the second term of the factored posterior in (11) can be also decoupled according to the Bayes' rule

$$P(m \mid z_{1:k+1}, u_{1:k}) \propto \ p(z_{k+1} \mid m, z_{1:k}, u_{1:k}) \ P(m \mid z_{1:k}, u_{1:k-1}). \tag{14}$$

$P(m \mid z_{1:k}, u_{1:k-1})$ is the prior belief on the trajectory class m. Now $p(z_{k+1} \mid m, z_{1:k}, u_{1:k})$ can be expressed as a marginal density function of s_{k+1}

$$p(z_{k+1} \mid m, z_{1:k}, u_{1:k}) = \int p(z_{k+1} \mid s_{k+1}) \times p(s_{k+1} \mid z_{1:k}, u_{1:k}, m) ds_{k+1}. \tag{15}$$

The first term in the integral above is the observation model. The second term is equivalent to (13). ■

Remark 1: This result suggests that we can separately update our belief of s conditional on m and the belief of m itself given the state transition model and the observation model. The factorization presented in the proof will allow us to derive closed-form parameter update equations, with certain assumptions imposed on the model as explained in the following section.

B. Multi-hypothesis Extended Kalman Filter

The problem of estimating the joint robot state s and the message m given past observations and control inputs can be viewed as a multi-hypothesis filtering problem. We employ the Multi-hypothesis Extended Kalman Filter algorithm as our filter. The nonlinear state transition and observation functions are linearized around the current estimate and the prior distribution on the joint robot state is assumed to be a mixture of Gaussians. In what follows, we derive the Kalman update equations based on the recursive Bayesian estimation formulae discussed above.

Algorithm 1 Multi-hypothesis Extended Kalman Filter algorithm for trajectory classification and state estimation.

```
INPUT: ${\mu_k^{(m)}, \Sigma_k^{(m)}, \phi_k^{(m)} \mid m = 1, \ldots, M}$, $u_k$, $z_{k+1}$
OUTPUT: ${\mu_{k+1}^{(m)}, \Sigma_{k+1}^{(m)}, \phi_{k+1}^{(m)} \mid m = 1, \ldots, M}$
 1: for each $m \in \{1, \ldots, M\}$ do
 2:     $F_k^{(m)} \leftarrow \frac{\partial}{\partial s} g_k(s, u_k, m)\big|_{\mu_k^{(m)}}$
 3:     $\bar{\mu}_{k+1}^{(m)} \leftarrow g_k(\mu_k^{(m)}, u_k, m)$
 4:     $\bar{\Sigma}_{k+1}^{(m)} \leftarrow F_k^{(m)} \Sigma_k^{(m)} F_k^{(m)\mathrm{T}} + Q$
 5:     $G_{k+1}^{(m)} \leftarrow \frac{\partial}{\partial s} h(s)\Big|_{\bar{\mu}_{k+1}^{(m)}}$
 6:     $H_{k+1}^{(m)} \leftarrow R + G_{k+1}^{(m)} \bar{\Sigma}_{k+1}^{(m)} G_{k+1}^{(m)\mathrm{T}}$
 7:     $K_{k+1}^{(m)} \leftarrow \bar{\Sigma}_{k+1}^{(m)} G_{k+1}^{(m)\mathrm{T}} H_{k+1}^{(m)-1}$
 8:     $\mu_{k+1}^{(m)} \leftarrow \bar{\mu}_{k+1}^{(m)} + K_{k+1}^{(m)} \left(z_{k+1} - h(\bar{\mu}_{k+1}^{(m)})\right)$
 9:     $\Sigma_{k+1}^{(m)} \leftarrow (I - K_{k+1}^{(m)} G_{k+1}^{(m)}) \bar{\Sigma}_{k+1}^{(m)}$
10:     $\phi_{k+1}^{(m)} \leftarrow \mathcal{N}\left(z_{k+1}; h(\bar{\mu}_{k+1}^{(m)}), H_{k+1}^{(m)}\right) \phi_k^{(m)}$
11: end for
12: $\phi_{k+1} \leftarrow normalize(\phi_{k+1})$
```

1) Model linearization: As in the usual EKF update equations, we approximate the nonliner state transition and observation with their first-order Taylor expansions. First, let $g_k(s_k, u_k, m)$ represent the joint robot state transition function defined by (2), (5) and $f_k(x_k, m)$, which is the state transition function for the sender. The linearization of g_k around $\mathbb{E}[s_k \mid z_{1:k}, u_{1:k-1}, m] \triangleq \mu_k^{(m)}$ yields

$$g_k(r_k, u_k, m) \approx g_k(\mu_k^{(m)}, u_k, m) + F_k^{(m)}(s_k - \mu_k^{(m)}) \tag{16}$$

with Jacobian $F_k^{(m)} \triangleq \frac{\partial}{\partial s} g_k(s, u_k, m)\big|_{\mu_k^{(m)}} \in \mathbb{R}^{9\times9}$.

In a similar manner, the observation function is linearized around $\mathbb{E}[s_{k+1} \mid z_{1:k}, u_{1:k}, m] \triangleq \bar{\mu}_{k+1}^{(m)}$. Assuming that the trajectory class is m, the linearized observation function is given by

$$h(s_{k+1}) \approx h(\bar{\mu}_{k+1}^{(m)}) + G_{k+1}^{(m)}(s_{k+1} - \bar{\mu}_{k+1}^{(m)}), \tag{17}$$

where $G_{k+1}^{(m)} \triangleq \frac{\partial}{\partial s} h(s)\big|_{\bar{\mu}_{k+1}^{(m)}} \in \mathbb{R}^{2\times9}$ is the Jacobian matrix.

2) Parameter update equations: In order to derive the Kalman update equations, we assume that the prior belief of s conditioned on m is a Gaussian distribution

$$p(s_k \mid z_{1:k}, u_{1:k-1}, m) \triangleq \mathcal{N}(s_k; \mu_k^{(m)}, \Sigma_k^{(m)}). \tag{18}$$

We also denote by $\phi_k^{(m)}$ the probability of the trajectory class being m given $z_{1:k}$ and $u_{1:k-1}$

$$P(m \mid z_{1:k}, u_{1:k-1}) \triangleq \phi_k^{(m)}. \tag{19}$$

As a result, the marginal distribution over s takes the form of a mixture of Gaussians

$$p(s_k \mid z_{1:k}, u_{1:k-1}) = \sum_{m=1}^{M} \phi_k^{(m)} \mathcal{N}(s_k; \mu_k^{(m)}, \Sigma_k^{(m)}). \quad (20)$$

Given the prior distribution (20), the Bayesian filtering (11)–(15) is equivalent to the parameter update equations given in Algorithm 1. This is consistent with similar Gaussian Mixture Filtering algorithms presented in [6] and [7]. Note that the categorical distribution parameters $\phi_{k+1}^{(m)}$ must be normalized so that $\sum_{m=1}^{M} \phi_{k+1}^{(m)} = 1$ is satisfied.

IV. Belief Initialization

A. Levenberg-Marquart algorithm

Convergence performance of the Extended Kalman Filter algorithm is sensitive to the accuracy of the prior belief. Especially the initial estimate of the receiver's pose is of particular importance; we observed in simulation that a randomized initial value for the pose estimate leads to poor convergence performance of the filter. This stems from the nonlinearity in the state transition model (2), (5). Thus we resort to the Levenberg-Marquardt algorithm [18], a well-known nonlinear least squares method in the computer vision and optimization literature. At the very beginning of the communication process, we let the sender perform a complete trajectory once while the receiver remains stationary at the initial pose and have the receiver collect a set of observations. Based on these observations and the noiseless trajectory waypoints $\{\tilde{x}_0, \tilde{x}_1^{(m)}, \ldots, \tilde{x}_n^{(m)}\}$ for trajectory class m in the codebook, the algorithm seeks for a minimizer of the following least squares

$$\arg\min_{\rho^{(m)}} \sum_{k=0}^{n} \left\| z_k - h\left((\rho^{(m)\mathrm{T}}, \tilde{x}_k^{(m)\mathrm{T}})^{\mathrm{T}} \right) \right\|_2^2. \quad (21)$$

Each run of the Levenberg-Marquardt algorithm itself requires an initial estimate $\rho_0^{(m)}$ and the algorithm runs until a local minimum is found or a desired number of iterations is finished. For a better performance, it is preferred that the initial value $\rho_0^{(m)}$ be a reasonable estimate. We employ the Direct Linear Transformation algorithm to compute this initial estimate from the same set of initial observations and the waypoint data from the codebook. The Direct Linear Transformation algorithm directly estimates the projection matrix P by solving a linear system of equations, from which we can extract the pose information $\rho_0^{(m)}$ using (7), (8) and (9). The details are omitted for brevity and readers are referred to [18], [10] for further information.

B. Prior estimate

After evaluating $\{\rho^{(m)} \mid m = 1, \ldots, M\}$, the prior is specified by the mean $\mu_0^{(m)} = \left(\rho^{(m)\mathrm{T}}, \tilde{x}_0^{\mathrm{T}}\right)^{\mathrm{T}}$, covariance matrix $\Sigma_0^{(m)}$, and the initial trajectory class probability $\phi_0^{(m)} = P(m)$ for each $m \in \{1, \ldots, M\}$. We could feedback the information of how well the nonlinear least squares performed to initialize $\phi_0^{(m)}$. Under the assumption that noise is independent and Gaussian distributed, $\phi_0^{(m)}$ can be evaluated by taking the likelihood of the set of observations using the Bayes' rule;

$$\phi_0^{(m)} \propto \exp\left(-\frac{1}{2\sigma^2} \sum_{k=0}^{n} \left\| z_k - h\left((\rho^{(m)\mathrm{T}}, \tilde{x}_k^{(m)\mathrm{T}})^{\mathrm{T}} \right) \right\|_2^2 \right), \quad (22)$$

in which the residual of the least squares appears and σ is a parameter. In the experiment we used $\sigma = 1e2$.

V. Entropy-based Active Vision Control

Given the current esimate of s and m, our goal is to control the position and the attitude of the receiver in order to correctly classify the trajectory and decode the message. One information theoretic approach is to select u_k from the prespecified control space $\mathcal{U}$ so that the entropy over the trajectory class distribution is maximally decreased. To this end, we formulate the optimal control problem as a belief-state Markov decision process and adapt the Monte Carlo Tree Search algorithm, an approximate dynamic programming method that computes near-optimal solutions in a given amount of time. More specifically, we employ its most well-known variant called Upper Confidence Trees [20] with an extension known as Double Progressive Widening [16].

A. Belief-state Markov decision process for motion-based communication

Markov decision processes provide a general framework for an autonomous agent to act optimally in a fully observable environment in order to maximize its expected cumulative reward, or the value function. In a belief-state Markov decision process, the state space is the entire belief space. A belief is defined by the following tuple in our problem,

$$b = \left((\phi^{(1)}, \mu^{(1)}, \Sigma^{(1)}), \ldots, (\phi^{(M)}, \mu^{(M)}, \Sigma^{(M)}) \right). \quad (23)$$

With this extension the same framework can be applied to problems with partial observability, which is the case in this motion-based communication scenario.

We also need to define the state transition model for the belief state b and the reward model related to the state transition.

1) Belief state transition model: Recall from Algorithm 1 that the belief state transition is completely described by the Multi-hypothesis Extended Kalman Filter equations once it is given the control input u_k and the observation z_{k+1}. There is stochasticity involved in this transition due to the observation. However, we can still specify the distribution from which a new observation is drawn based on the current belief b that has $\{z_{1:k}, u_{1:k}\}$ as evidence variables. Namely,

$$p(z_{k+1} \mid z_{1:k}, u_{1:k}) = \sum_{m=1}^{M} p(z_{k+1} \mid m, z_{1:k}, u_{1:k}) \times P(m \mid z_{1:k}, u_{1:k}), \quad (24)$$

```
Algorithm 2 Sample a new belief state and a reward.
INPUT: current belief $b$, control input $u$.
OUTPUT: new belief $b'$, reward $r$
1: Compute $\bar{\mu}^{(m)}$ and $H^{(m)}$ for each $m$ from $b$.
2: Sample $z \sim \sum_{m=1}^{M} \phi^{(m)} \mathcal{N}\left(z; h(\bar{\mu}^{(m)}), H^{(m)}\right)$
3: Obtain $b'$ from $u$ and $z$ with Algorithm 1.
4: Compute $r = Reward(b, b')$ (27).
```

where $p(z_{k+1} \mid m, z_{1:k}, u_{1:k}) = \mathcal{N}\left(z_{k+1}; h(\bar{\mu}_{k+1}^{(m)}), H_{k+1}^{(m)}\right)$ and $P(m \mid z_{1:k}, u_{1:k}) = \phi_k^{(m)}$. Therefore, we can simulate stochastic belief state transitions by drawing an observation sample from the following Gaussian mixture distribution.

$$z_{k+1} \sim \sum_{m=1}^{M} \phi_k^{(m)} \mathcal{N}\left(z_{k+1}; h(\bar{\mu}_{k+1}^{(m)}), H_{k+1}^{(m)}\right) \tag{25}$$

2) Reward model: The reward needs to be designed so that the autonomous agent exhibits desired behavior. In the motion-based communication problem, we would like to reduce the uncertainty in the trajectory class distribution. Entropy provides a measure of uncertainty in the current belief, which is defined as follows.

$$H[\mathcal{M}] = -\sum_{m=1}^{M} P(m) \log P(m), \tag{26}$$

where $\mathcal{M}$ is the random variable representing the trajectory class.

The reward function $Reward(b, b')$ is determined by the change in entropy between the current belief b and a new belief b'

$$Reward(b, b') = l\left(H[\mathcal{M} \mid b]\right) - l\left(H[\mathcal{M} \mid b']\right), \tag{27}$$

where $l : \mathbb{R}_+ \to \mathbb{R}$ is a strictly monotonically decreasing function. Note that the reward is positive when the new belief b' results in a smaller entropy than b since the receiver aims at maximizing the reward.

In simulation experiments we chose $l(\cdot) = log(\cdot)$. This is to prevent the reward from effectively getting discounted as the entropy is reduced because the entropy of a categorical distribution (26) is always lower-bounded by zero.

The resulting sampling algorithm is summarized in Algorithm 2.

B. Upper Confidence Trees

The Upper Confidence Trees algorithm approximates the state-action value function using Monte Carlo simulations with a generative model that gives reward and state transition samples. It is an online method that only considers a certain depth from the current state. The depth and the number of iterations are limited by the amount of computational resources available before selecting the next action. Starting from the current state as the root node, in each iteration the algorithm gradually grows a search tree. A node in the tree represents a reachable state and an edge corresponds to an action. In our problem, a state is a belief state b and an action is a control input u.

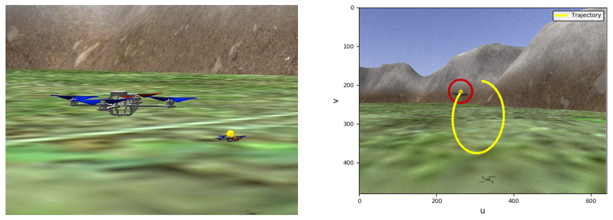


Fig. 3. (Left) Simulation environment with the receiver in the front and the sender in the back. (Right) An image obtained by the forward-facing camera mounted on the receiver. The smooth yellow trajectory is overlaid for clarity.

1) UCB1: The key aspect of the algorithm is the use of a search heuristic called UCB1, which well balances the exploration and exploitation trade-off. At node b control input u is chosen so that it maximizes the following score [16]

$$score_t(b, u) = \frac{totalReward_t(b, u)}{n_t(b, u) + 1} + k_{ucb} \sqrt{\frac{log(t)}{n_t(b, u) + 1}}, \tag{28}$$

where t is the number of times the node b has been visited, $n_t(b, u)$ is the number of times the control input u has been applied at b, and $totalReward_t(b, u)$ is the total cumulative reward obtained so far by executing u at b. k_{ucb} is a parameter that controls the amount of exploration.

2) Double Progressive Widening: The conventional Upper Confidence Trees algorithm assumes small discrete action and state spaces. Nevertheless, the belief-state Markov decision process defined above consists of stochastic transitions in a continuous domain and the probability of reaching the same belief state twice is zero. Couëtoux et al. [16] point out that the algorithm fails in such a case and propose a modification as follows. Instead of always transitioning to a new state using the generative model, a new state is added to the tree only if the number of children is less than or equal to $\lceil C n_t(b, u)^{\alpha} \rceil$, where $C > 0$ and $\alpha \in]0, 1[$ are constants; otherwise an existing child is chosen with probability $n_t(b, u, b')/n_t(b, u)$, where $n_t(b, u, b')$ is the number of transitions observed from b to b' with u. A similar strategy is used to limit the number of actions available if the action space is also large. The k-th action becomes available when $k = \lceil C_u t^{\alpha_u} \rceil$ with constatns $C_u > 0$ and $\alpha_u \in]0, 1[$. This restriction is known as Progressive Widening, and the algorithm is called Double Progressive Widening as it is applied to both the states and the actions.

VI. Experimental Results

A. Simulation setup

We designed our experiment in Gazebo [21], a multi-robot simulator compatible with ROS [22]. Two Hummingbird [23] quadrotors are used in the simulation, whose dynamics were simulated with RotorS [24]. Fig. 3 illustrates the simulation environment. [1]

[1]The supplemental video is available at https://youtu.be/u4HxiA2d2IQ

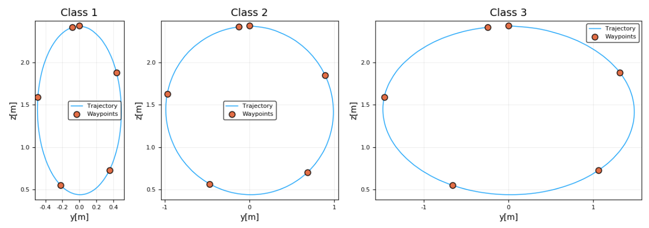


Fig. 4. 2D trajectory codebook used in the experiment. The sender robot moves clockwise. The trajectories are intentionally ambiguous from different observer angles, to make trajectory classification difficult for the receiver.

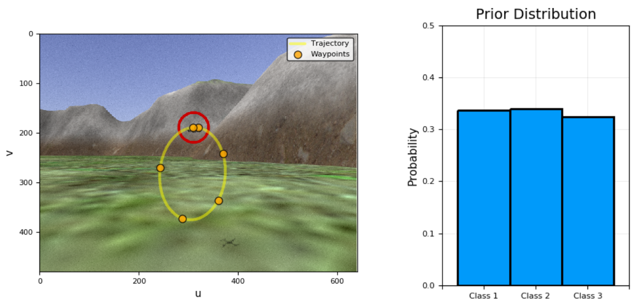


Fig. 5. (Left) A set of waypoint measurements used to initialize the prior. The smooth yellow trajectory and the waypoints are overlaid for clarity. (Right) Prior distribution obtained from the left image.

The trajectory codebook presented in Fig. 4 consists of three different elliptic trajectories. A trajectory is represented by 6 waypoints. The sender takes 7 seconds to transition between successive waypoints, which corresponds to a single timestep. The codebook was deliberately designed to make the classification challenging. Indeed, the prior initialization with the Levenberg-Marquardt algorithm resulted in an ambiguous classification as depicted in Fig. 5.

The parameters used in the Multi-hypothesis Extended Kalman Filter and the Double Progressive Widening algorithms are summarized in Table I. The horizontal and the vertical FoVs of the camera were 80.1 [deg] and 64.5 [deg], respectively. Initially the two quadrotors were 4.2 [m] apart in the horizontal direction. In controlling the receiver, we only allowed lateral translational motion and yaw rotation as possible control inputs: $u_{trans} = (0, u_y, 0)^{\mathrm{T}}$ and $u_{rot} = (0, 0, u_{yaw})^{\mathrm{T}}$ with $u_y \in \{-1, -0.5, 0, 0.5, 1\}[m]$ and $u_{yaw} \in \{-45, -25, -10, 0, 10, 25, 45\}[deg]$. The resulting size of the control space was $|\mathcal{U}| = 35$.

We compared the performance of the proposed control policy with a greedy algorithm and a random policy. The greedy algorithm still employs Monte Carlo simulations to estimate the reward, but it only considers the immediate next action and performs Algorithm 2 ten times per action to select one that has the maximum average reward.

B. Simulation results

The classification and the pose estimation performance was evaluated through 30 simulation episodes with different trajectory classes and priors. In each episode the sender repeated the trajectory execution 6 times, with an interval

TABLE I
PARAMETER VALUES.

Parameter	Value	Parameter	Value
Q	$diag(Q_r, Q_t, Q_x)$	k_{ucb}	10.0
Q_r	$diag(1e^{-3}, 1e^{-3}, 1e^{-2})$	C	3.0
Q_t	$diag(5e^{-2}, 5e^{-2}, 1e^{-3})$	α	0.1
Q_x	$diag(1e^{-4}, 1e^{-3}, 1e^{-3})$	C_u	10.0
R	$diag(10.0, 10.0)$	α_u	0.3

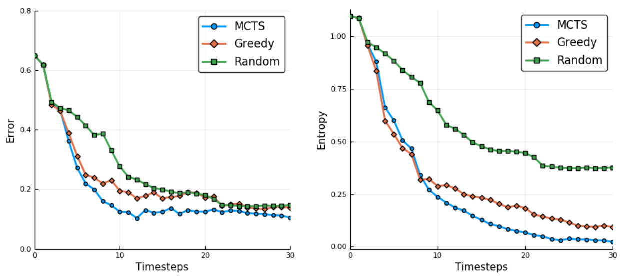
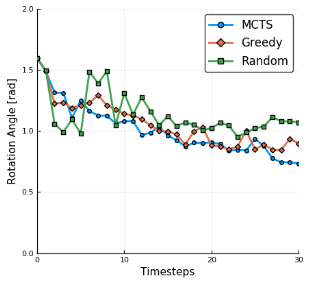


Fig. 6. Classification error (left) and entropy (right) averaged over 30 cases. The MCTS active sensing strategy outperforms greedy and random in both criteria.

time of 7 seconds between each execution. We observed that the MCTS and the greedy policies both actively controlled the yaw rotation as well as the horizontal position, trying to keep the sender inside the FoV of the camera. As can be seen in Fig. 6, the MCTS control policy resulted in the lowest classification error, which is defined as $1 - P(m_{true})$. Even though all the policies achieved similar errors after step 22, there are still notable differences in the entropy, indicating that with the proposed approach the receiver is most confident in its classification.

However this performance improvement was obtained at the expense of computational time. On average Monte Carlo Tree Search with Double Progressive Widening took 4.0 seconds to compute the next action, with the maximum tree depth of 3 and 400 iterations. The greedy algorithm only took 1.6 seconds to perform 350 iterations. Obviously the random policy ran much faster than the other two, taking 0.0024 seconds. The computation was performed on a laptop with Intel Core i7-6700HQ CPU and 16GB RAM. Considering this runtime and the classification performance discussed above, there is a trade-off between the proposed approach and the greedy policy since the performance of the greedy policy is not significantly worse than the proposed method.

The MCTS active sensing strategy also outperformed the other two policies in the pose estimate, even though the filtering algorithm was not able to correct the pose error within 30 steps regardless of the control policy. In Fig. 7 the attitude error is represented by the magnitude of the rotation angle between the true attitude and the mean estimate. The position error is given by the distance between the true position and the mean estimate, which is divided by the initial distance of the two robots to give a normalized unitless distance error. Interestingly, the results suggest that the classification can become correct when the pose estimate is still erroneous.



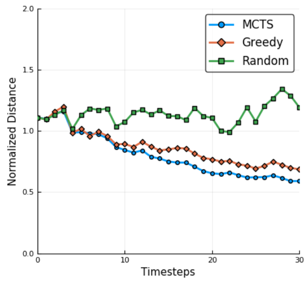


Fig. 7. Attitude error (left) and position error (right) averaged over 30 cases. The MCTS active sensing strategy performs the best for pose estimation, even though it is not explicitly trying to optimize the relative pose estimate.

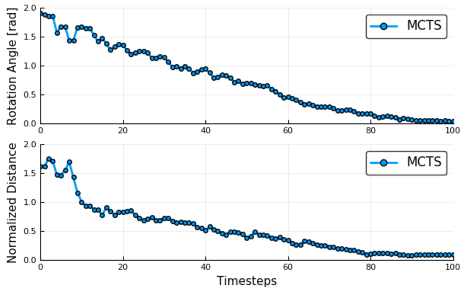


Fig. 8. Attitude error (top) and position error (bottom) for a single long episode with class 1 trajectories. The MCTS active sensing strategy was used to move the receiver. Eventually, the relative pose estimation converges even though the receiver's controller optimizes for trajectory class, not pose error.

In order to confirm that the filter essentially improves the pose estimate over time, we ran an additional episode for 100 steps with the proposed control policy. The result is presented in Fig. 8. Both the attitude error and the position error converged after 100 steps.

VII. Conclusion

We have presented an online trajectory classification algorithm for motion-based communication between two robots. Communication difficulties arising from the monocular vision and unknown pose are addressed, and a Bayesian approach is derived, taking a structure similar to the Gaussian mixture filtering. We also proposed an entropy-based active control policy to reduce classification uncertainty over time by formulating a belief-state Markov decision process. Through simulation studies we demonstrated that the proposed method significantly improves classification accuracy. In future work we will consider improving our active perception method to account for certain constraints such as collision avoidance. We also intend to generalize this active classification framework and apply to other applications such as autonomous driving.

References

[1] J. Baillieul and K. Özcimder, "The control theory of motion-based communication: Problems in teaching robots to dance," in *2012 American Control Conference (ACC)*. IEEE, jun 2012, pp. 4319–4326.

[2] A. Jones and S. Andersson, "A motion-based communication system," in *2013 American Control Conference*. IEEE, jun 2013, pp. 365–370.

[3] D. Raghunathan and J. Baillieul, "Relative motion of robots as a means for signaling," in *Proceedings of the World Congress on Engineering and Computer Science 2008*, San Francisco, USA, 2008.

[4] ——, "Motion based communication channels between mobile robots - a novel paradigm for low bandwidth information exchange," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, oct 2009, pp. 702–708.

[5] ——, "Exploiting information content in relative motion," in *2009 American Control Conference*. IEEE, 2009, pp. 2166–2171.

[6] D. L. Alspach and H. W. Sorenson, "Nonlinear bayesian estimation using gaussian sum approximations," *IEEE Transactions on Automatic Control*, vol. 17, no. 4, pp. 439–448, 1972.

[7] G. Jochmann, S. Kerner, S. Tasse, and O. Urbann, "Efficient multi-hypotheses unscented kalman filtering for robust localization," in *RoboCup 2011: Robot Soccer World Cup XV*. Springer Berlin Heidelberg, jun 2012, vol. 7416 LNCS, pp. 222–233.

[8] R. Chen and J. S. Liu, "Mixture kalman filters," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 62, no. 3, pp. 493–508, aug 2000.

[9] M. A. Sipe and D. P. Casasent, "Best viewpoints for active vision classification and pose estimation," in *Proc. SPIE 3208, Intelligent Robots and Computer Vision XVI: Algorithms, Techniques, Active Vision, and Materials Handling*, D. P. Casasent, Ed. International Society for Optics and Photonics, sep 1997, pp. 382–393.

[10] J. Denzler, M. Zobel, and H. Niemann, "On optimal camera parameter selection in kalman filter based object tracking," *Pattern Recognition*, pp. 17–25, 2002.

[11] M. Zobel, J. Denzler, and H. Niemann, "Entropy based camera control for visual object tracking," in *Proceedings. International Conference on Image Processing*, vol. 3. IEEE, jun 2002, pp. 901–904.

[12] A. Seekircher, T. Laue, and T. Röfer, "Entropy-based active vision for a humanoid soccer robot," in *RoboCup 2010: Robot Soccer World Cup XIV*. Springer Berlin Heidelberg, 2011, vol. 6556 LNCS, pp. 1–12.

[13] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of monte carlo tree search methods," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 1, mar 2012.

[14] S. Gelly, L. Kocsis, M. Schoenauer, M. Sebag, D. Silver, C. Szepesvári, and O. Teytaud, "The grand challenge of computer go: Monte carlo tree search and extensions," *Commun. ACM*, vol. 55, no. 3, pp. 106–113, mar 2012.

[15] G. Best, O. M. Cliff, T. Patten, R. R. Mettu, and R. Fitch, "Decentralised monte carlo tree search for active perception," in *The 12th International Workshop on the Algorithmic Foundations of Robotics*, dec 2016.

[16] A. Couëtoux, J.-B. Hoock, N. Sokolovska, O. Teytaud, and N. Bonnard, "Continuous Upper Confidence Trees," in *2011 International Conference on Learning and Intelligent Optimization*. Springer, Berlin, Heidelberg, 2011, pp. 433–445.

[17] P. Slade, P. Culbertson, Z. Sunberg, and M. J. Kochenderfer, "Simultaneous active parameter estimation and control using sampling-based bayesian reinforcement learning," *CoRR*, vol. abs/1707.09055, 2017. [Online]. Available: http://arxiv.org/abs/1707.09055

[18] V. Lepetit and P. Fua, "Monocular model-based 3d tracking of rigid objects," *Foundations and Trends in Computer Graphics and Vision*, vol. 1, no. 1, pp. 1–89, jan 2005.

[19] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, sep 2004.

[20] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *Proceedings of the 17th European Conference on Machine Learning*, ser. ECML'06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 282–293.

[21] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3. IEEE, 2004, pp. 2149–2154.

[22] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[23] C. R. Inc. Hummingbird unmanned aerial vehicle. https://www.clearpathrobotics.com/hummingbird-unmanned-aerial-vehicle/, visited 2017-09-13.

[24] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "RotorS A Modular Gazebo MAV Simulator Framework," in *Robot Operating System (ROS)*. Springer, Cham, 2016, vol. 625 SCI, pp. 595–625.