Simultaneous Estimation of Target Pose and 3-D Shape using the FastSLAM Algorithm

Sean Augenstein[*] and Stephen M. Rock[†‡]

[*,†] *Aerospace Robotics Lab, Stanford University, 496 Lomita Mall, Stanford, CA, 94305, USA*
[‡] *Monterey Bay Aquarium Research Institute, 7700 Sandholdt Rd, Moss Landing, CA, 95039, USA*

A solution is presented to the problem of estimating recursively the 6-DOF pose and 3-D shape of a target, using insights from the SLAM community. The algorithm in this paper is referred to as 'SLAM-inspired Pose Estimation and Reconstruction' (or 'SPEAR'), and is based on the FastSLAM algorithm. In particular, FastSLAM's use of Rao-Blackwellized particle filters is emulated. SPEAR tracks the target with measurements from cameras; this paper focuses on the monocular case. No *a priori* knowledge of the target is required.

This paper discusses the end-to-end implementation of SPEAR in detail, including the algorithm itself, feature initialization steps, and the image processing step (which uses SIFT). Also, specifics are given on what is needed computationally to perform SPEAR in real-time.

The rationale for applying Rao-Blackwellized particle filters is discussed. Compared with a solely EKF-based approach to solving SPEAR, particle filters allow for more efficient measurement updates and are not constrained by the assumption of unimodal probability distributions.

Results are given for several targets in the lab and in the field, demonstrating successful estimation of each target's pose along with the 3-D locations of features on the body.

I. Introduction

This paper presents an efficient particle filter-based approach to estimating relative pose and shape of an object of interest ('the target'). In particular, the algorithm is called SLAM-inspired Pose Estimation and Reconstruction (henceforth referred to as 'SPEAR'), and can be considered a solution to the recursive online Structure from Motion (SFM) problem. With a monocular camera and with no *a priori* information about the target, the algorithm determines probabilistically the target's kinematic state $\overline{s}$ (orientation, angular rates, position, and velocity) and structure $\mathbf{X}$ (body locations of point features).

The motivation for this research is the problem of autonomously docking a chaser vehicle with an unknown tumbling body. A real-time estimate of the target's pose and 3-D shape is a prerequisite for autonomous control algorithms that would fly the chaser to a prescribed location on the target. Examples of such bodies are a satellite that has lost attitude control and requires repair (illustrated in Figure 1), or an undersea mooring mounted on a tether and subject to ocean currents.

The SPEAR algorithm is inspired by the structural similarities between the Simultaneous Localization and Mapping (SLAM) field and the pose estimation and reconstruction task described above. SLAM is concerned with the problem of determining own-robot location/pose along with a map of the robot's surroundings.[1] Where SLAM algorithms are concerned with processing measurements to form a map of the external world, SPEAR processes measurements to form a 3-D 'map' (i.e. the shape) of a distinct external target. Where SLAM algorithms are concerned with the kinematic states of the own-robot making the measurements, SPEAR estimates the relative kinematic states of the external target.

Specifically, SPEAR is based on FastSLAM,[2,3] a SLAM algorithm utilizing Rao-Blackwellized particle filters.[4] There are two benefits. First, compared with common alternatives such as extended Kalman filters

[*]PhD Candidate, Aeronautics & Astronautics, Stanford University, AIAA Student Member
[†]Professor, Aeronautics & Astronautics, Stanford University, AIAA Fellow
[‡]Adjunct, Monterey Bay Aquarium Research Institute

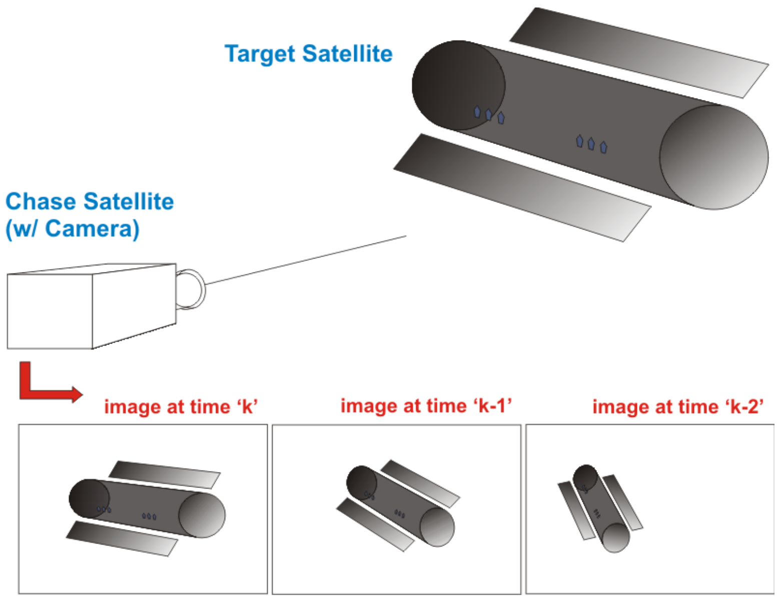

Figure 1. Tumbling Satellite being Imaged by Chase Vehicle

(EKFs), particle filters make less constraints on the probability distribution over the states. A multi-modal, highly non-Gaussian distribution is admissible. Second, the Rao-Blackwellization of the algorithm greatly reduces the computational complexity, by exploiting the factorization of the feature location estimation.[5] EKF-based SLAM requires $O(N^2)$ operations to perform measurement updates, whereas FastSLAM can be done in $O(M \log N)$ operations, where N is the number of features, and M the number of particles.[2] It appears that a Rao-Blackwellized particle filter has not been previously used for SFM.

The details of the vision front end to the algorithm are also described in this paper. The SIFT feature algorithm[6] is used for detecting point features on the target. SIFT was chosen for its invariance properties, allowing for tracking/matching of the features through a variety of a transformations. It should be stressed that the actual SPEAR algorithm itself is independent of the particular choice of image point features used, provided that correspondence is taken care of upstream of the particle filter. Figure 2 shows the block diagram of the end-to-end pose estimation and shape reconstruction process.

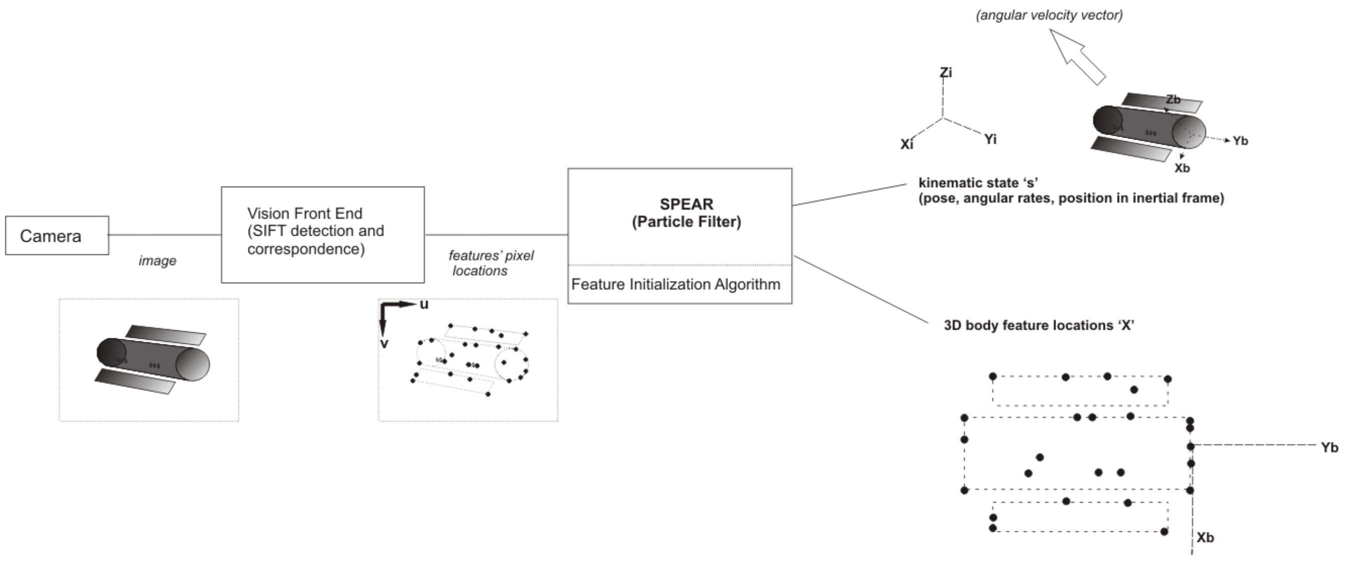
(angular velocity vector)
Zi
Xi
Yi
Zb
Yb
Xb
kinematic state 's'
(pose, angular rates, position in inertial frame)
Camera
image
Vision Front End
(SIFT detection and correspondence)
features' pixel locations
u
v
SPEAR
(Particle Filter)
Feature Initialization Algorithm
3D body feature locations 'X'
Yb
Xb

Figure 2. Block Diagram of Overall SPEAR Process

II. Background

The algorithm described here is applicable broadly because it assumes nothing about the target and employs only a very basic sensor for the chaser. Related work has been done with some form of prior knowledge of the target, such as fiducials (markers on the target at known locations)[7] or a known 3-D model.[8] Also, the problem has been approached with solutions that utilize richer sensors (Lidar or stereo vision).[9] These sensors give range and bearing information, while monocular vision is bearings-only.

The problem of bearings-only reconstruction of an object up to a scale factor, along with estimation of the relative poses of the camera, falls under the category of Structure-from-Motion (SFM). Much of the SFM field is devoted towards offline, batch processing to find an optimal solution for the shape and poses based on all the images acquired.[10] While the solution obtained is more accurate, an offline batch algorithm is of no use in a real-time estimation/control loop.

There are also SFM algorithms that recursively process each new image online, and might be applied in a real-time situation. The dominant approach to performing recursive SFM has been to use extended Kalman filters (EKFs).[11–13] However, the selection of an EKF makes a constraining assumption on the probability distribution of the states of the filter, namely, that they are Gaussian-like (i.e., unimodal). In addition, many of these methods have various computational drawbacks, such as requiring inversion of large matrices and/or requiring a batch process for initialization.

More recently, research has been done on particle filter-based SFM,[14–16] which relaxes the assumption of a Gaussian posterior. Some of these methods use fiducials for state initialization[15] and periodic state correction.[16] Some are motivated by issues of unknown feature correspondence between images[14] and have not run in real-time. Another recently proposed SFM method, which can run in real-time, focuses on scoring potential state hypotheses via the RANSAC algorithm;[17] this can be considered analogous to the weighting step in particle filter methods.

It should be noted that SPEAR is largely unrelated to the fields of dynamic mapping or moving object detection. Dynamic mapping primarily focuses on occupancy grid-based methods, to classify and separate static regions from moving objects.[5,18] In contrast, the algorithm described here is feature-based. Research into moving object detection focuses on complementing the SLAM problem. Dynamic objects are identified, so that they are not treated as part of the static map when the standard SLAM algorithm is solved;[19] shape reconstruction of the moving objects is not performed.

III. Estimator States

There are two sets of variables being estimated: the kinematic state $\overline{s}$ (which evolves over time), and the 3-D location of the features in the body frame, $\mathbf{X}$ (constants, since a rigid body is assumed). For notation, the discrete time steps are indexed by k, and the features are distinguished by index j. Explicitly, the kinematic state vector at time k is:

$$\overline{s}_k = \begin{bmatrix} \phi & \theta & \psi & \omega_{x_R} & \omega_{y_R} & \omega_{z_R} & x_t & y_t & z_t & \dot{x}_t & \dot{y}_t & \dot{z}_t \end{bmatrix}_k^T \tag{1}$$

The Euler angles ϕ, θ, and ψ give the rotation to the body frame from the reference frame; ω_{x_R}, ω_{y_R}, and ω_{z_R} form the target's angular rate vector, expressed in the reference frame; x_t, y_t, z_t, $\dot{x}_t$, $\dot{y}_t$, $\dot{z}_t$ are the position and velocity of the target's center of rotation in the reference frame. The reference frame can be either the camera's frame (in which case the position is in non-dimensional coordinates), or an inertial frame if position/orientation sensors are available on the observing vehicle.

The matrix of feature locations on the body is:

$$\mathbf{X} = \begin{bmatrix} X_1 & X_2 & \cdots & X_j & X_{j+1} & \cdots \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & & x_j & x_{j+1} & \\ y_1 & y_2 & \cdots & y_j & y_{j+1} & \cdots \\ z_1 & z_2 & & z_j & z_{j+1} & \end{bmatrix} \tag{2}$$

IV. Estimator Models

The kinematic states are propagated forward in time according to the motion model $f(\overline{s})$:

$$\overline{s}_k = f(\overline{s}_{k-1}) + \overline{n}_p$$

$$\begin{bmatrix} \phi \\ \theta \\ \psi \\ \omega_{x_R} \\ \omega_{y_R} \\ \omega_{z_R} \\ x_t \\ y_t \\ z_t \\ \dot{x}_t \\ \dot{y}_t \\ \dot{z}_t \end{bmatrix}_k = \begin{bmatrix} \phi_{(k-1)} + \Delta t * \dot{\phi}_{(k-1)} \\ \theta_{(k-1)} + \Delta t * \dot{\theta}_{(k-1)} \\ \psi_{(k-1)} + \Delta t * \dot{\psi}_{(k-1)} \\ \omega_{x_R(k-1)} \\ \omega_{y_R(k-1)} \\ \omega_{z_R(k-1)} \\ x_{t(k-1)} + \Delta t * \dot{x}_{t(k-1)} \\ y_{t(k-1)} + \Delta t * \dot{y}_{t(k-1)} \\ z_{t(k-1)} + \Delta t * \dot{z}_{t(k-1)} \\ \dot{x}_{t(k-1)} \\ \dot{y}_{t(k-1)} \\ \dot{z}_{t(k-1)} \end{bmatrix} + \overline{n}_p \tag{3}$$

where

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}_k = \begin{bmatrix} 1 & tan(\theta_k)sin(\phi_k) & tan(\theta_k)cos(\phi_k) \\ 0 & cos(\phi_k) & -sin(\phi_k) \\ 0 & \frac{sin(\phi_k)}{cos(\theta_k)} & \frac{cos(\phi_k)}{cos(\theta_k)} \end{bmatrix} \mathbf{R}^{B/R} \begin{bmatrix} \omega_{x_R(k)} \\ \omega_{y_R(k)} \\ \omega_{z_R(k)} \end{bmatrix} \tag{4}$$

In Eq. 3, $\overline{n}_p$ is additive noise, with Gaussian distribution $\mathcal{N}(0, P_{[4:12,4:12]})$ on the rates, position, and velocity components of $\overline{s}_k$.

A Fisher distribution ($\mathcal{F}(\overline{\mu}_{Fisher}, \kappa_{Fisher})$), a probability model for directions in space,[20] is used as the basis for applying noise on the Euler angle components of $\overline{s}_k$. This ensures perturbations to the target's 3-D orientation are identically distributed over the entire range of Euler angle parameters. The $\overline{\mu}_{Fisher}$ parameter is the unit vector that is the center of the directional distribution (analogous to the mean in a Gaussian distribution). The κ_{Fisher} parameter represents the 'tightness' of the directional distribution (analogous to the inverse of the covariance in a Gaussian distribution).

In application here, $\overline{\mu}_{Fisher}$ is set to the unit vector pointing from the target's center to the camera. Furthermore, the standard Fisher distribution is modified in that perturbations are added to rotate the target about $\overline{\mu}_{Fisher}$. Without this, a Fisher distribution would not induce in-plane rotational noise.

In Eq. 4, $\mathbf{R}^{B/R}$ is the rotation matrix which rotates vectors from the reference frame R into the body frame B.

The measurements ($\overline{z}$) are pixel coordinates (u, v). A pinhole measurement model $g(\overline{s}, X_j)$ is assumed:

$$\overline{z}_j = \begin{bmatrix} u_j \\ v_j \end{bmatrix} = g(\overline{s}, X_j) + \overline{n}_c = \frac{foc}{z^C_{j/cam}} \begin{bmatrix} x \\ y \end{bmatrix}^C_{j/cam} + \overline{n}_c \tag{5}$$

In Eq. 5, $\overline{n}_c$ is additive Gaussian noise with distribution $\mathcal{N}(0, r^2 I)$.

The relationship between the feature location w.r.t. the camera in the camera frame ($\overline{X}^C_{j/cam}$) and the feature location w.r.t. the target in the target's body frame ($X_j = \overline{X}^B_{j/target}$) is given by:

$$\begin{aligned} \overline{X}^C_{j/cam} &= \mathbf{R}^{C/I} * \overline{X}^I_{j/cam} \\ &= \mathbf{R}^{C/I} * (\overline{X}^I_{j/target} + \bar{t}_{target/cam}) \\ &= \mathbf{R}^{C/I} * (\mathbf{R}^{I/B} * \overline{X}^B_{j/target} + \bar{t}_{target/cam}) \end{aligned} \tag{6}$$

In Eq. 6, $\mathbf{R}^{R/B}$ is the rotation matrix which rotates vectors from the body frame B into the body frame R, and $\mathbf{R}^{C/R}$ is the rotation matrix which rotates vectors from the reference frame R into the camera frame C (if these frames are not the same).

V. Particle Filters

This section describes the rationale behind applying a particle filter to the problem posed in this paper. Additionally, it describes how the estimator states discussed in Section III are represented in the particle filter.

V.A. Motivation

When estimating the target's kinematic state $\overline{s}$, the strength of a particle filter over an EKF is that the probability distribution is not restricted to being unimodal, quasi-Gaussian. This is important in the context of the bearings-only SPEAR algorithm. The posterior is evolving based on bearings-only measurements, and these can give rise to multiple distinct regions of high likelihood in the distribution.

For example, consider the scenario illustrated in Figure 3. This shows a slowly rotating target with 4 features at equal depth from the camera at time 0.

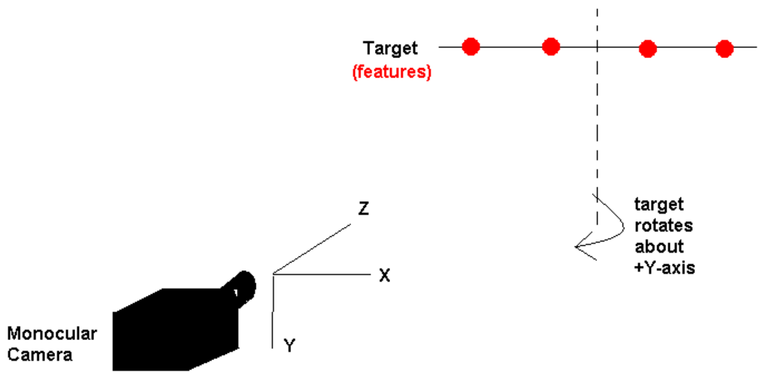

Figure 3. Target Motion Scenario

Figure 4 shows the particle cloud evolve over time; the dark dots are the prior particles, and the light dots are the re-sampled particles. As can be seen, two distinct groups emerge from the initial distribution, each representing a distinct, high likelihood hypothesis of the angle of the target. Figure 5 shows the same results, as plots of particle weights versus hypothesis at images 6, 9, 12, and 15. This illustrates the fact that while the posterior probability is *high* the target has rotated to some angle about either +Y or -Y, the probability is *low* the target has not rotated at all.

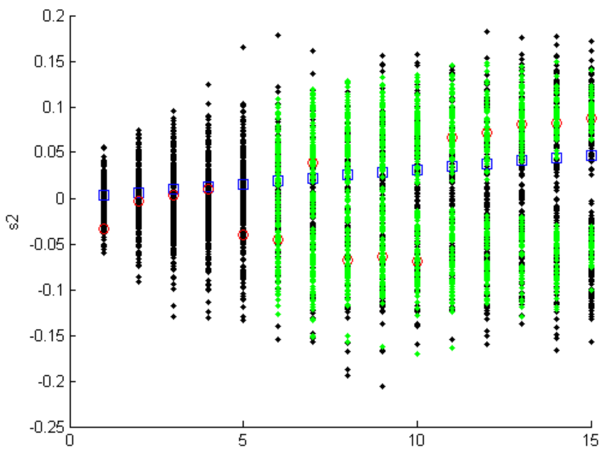

Figure 4. Posterior Evolution over Time

A second reason for choosing particle filters for the target reconstruction/ pose estimation problem is computational. Particle filters (specifically, Rao-Blackwellized particle filters or RBPFs) allow for efficient

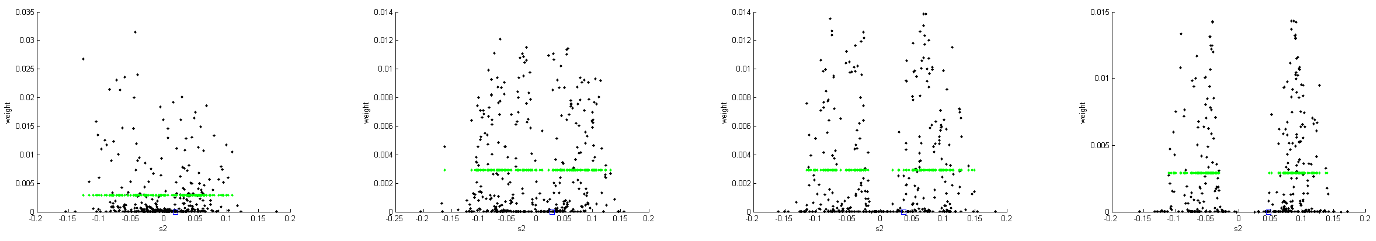

Figure 5. Posterior Distributions

tracking of a large set of target features X_j. A key idea from the FastSLAM literature and earlier work is that, conditioned on the target's kinematic state $\overline{s}$, the features' position uncertainties are independent of each other.[2,5] That is:

$$p(X_1, X_2, X_3|\overline{s}) = p(X_1|\overline{s})p(X_2|\overline{s})p(X_3|\overline{s}) \tag{7}$$

Since each particle i contains its own hypothesis of the kinematic state $\overline{s}^{[i]}$, it implicitly conditions its estimates of X_j on $\overline{s}^{[i]}$. Thus it is possible, within a given particle, to track each feature independent of the other features.

RBPFs exploit this partitioning of the state-space.[4] Instead of maintaining a very high-dimensional particle filter over the entire state-space, the particles only sample over the kinematic state $\overline{s}$; the distributions on the feature locations X_j are conditional on $\overline{s}$ and maintained analytically. In this paper, the analytic method for maintaining $p(\mathbf{X}|\overline{s}^{[i]})$ is via individual EKFs for each feature. This yields a significant computational benefit when the number of features N gets large. Instead of having to maintain a single large $3N$x$3N$ covariance matrix for the feature position estimates, MN 3x3 covariance matrices (each associated to a specific particle and feature) are maintained (where M is the number of particles).

V.B. Representation of States

The particle filter consists of many particles (referred to by index i), each containing a hypothesis for the kinematic state at the latest time step k, $\overline{s}_k^{[i]}$.

As discussed in Section V, within each particle, each feature can be estimated independently of the others. This algorithm duplicates the approach of FastSLAM, which has each particle maintain a set of small EKFs, one for each feature. In other words, within each particle i, the distribution $p(X_j|\overline{s}^{[i]})$ is assumed to be Gaussian, $\mathcal{N}(\overline{\mu}_j^{[i]}, \Sigma_j^{[i]})$.

So each particle contains N EKFs; that is, for particle i, every feature j has a mean $\overline{\mu}_j^{[i]}$ and covariance $\Sigma_j^{[i]}$ to estimate its true 3-D location in the body frame (X_j).

VI. Algorithm

At every time-step, the particle filter proceeds through three distinct updates: a measurement update, a weighting/resampling of the particles based on their likelihoods, and a motion update to predict the kinematic state at the next time step. These are described in turn.

VI.A. Measurement Updates

This process is done for every feature observed, for each particle (because each particle maintains its own estimates of the feature locations). To update a given feature's estimate for a given particle, the measurement model $g(\overline{s}, X_j)$ is used to predict the next measurement of that feature, and calculate its Jacobians (recall i indexes the particles and j indexes the features):

$$\hat{\overline{z}}_j^{[i]} = g(\overline{s}^{[i]}, \overline{\mu}_j^{[i]}) \tag{8}$$

$$G_{X_j} = \left.\frac{\partial g}{\partial X_j}\right|_{\left(\overline{s}^{[i]}, \overline{\mu}_j^{[i]}\right)} \tag{9}$$

$$G_s = \frac{\partial g}{\partial \overline{s}}\Big|_{\left(\overline{s}^{[i]}, \overline{\mu}_j^{[i]}\right)} \tag{10}$$

This is followed by a standard EKF measurement update, performed on the feature's mean and covariance (Eqns. 11-14). Note that r is the noise standard deviation. This parameter is a measure of the importance with which the latest measurement is treated.

$$Q = \left(G_s P {G_s}^T + G_{X_j}\Sigma_j^{[i]}{G_{X_j}}^T + r^2 I\right) \tag{11}$$

$$K = \Sigma_j^{[i]}{G_{X_j}}^T Q^{-1} \tag{12}$$

$$\overline{\mu}_j^{[i]} = \overline{\mu}_j^{[i]} + K(\overline{z}_j - \hat{\overline{z}}_j^{[i]}) \tag{13}$$

$$\Sigma_j^{[i]} = \left(I - K G_{X_j}\right)\Sigma_j^{[i]} \tag{14}$$

VI.B. Particle Weighting and Resampling

At each time step, the particle filter updates an importance weighting for each of the particles. This is a measure of the likelihood of the particle, given the measurements observed up to the given time step. The importance weighting of particle i is denoted $w^{[i]}$. There are many choices available for weighting measures. For example, inlier/outlier counts based on pixel error thresholding have been used in previous SFM particle filters.[14,15]

Given the choice made above to model $p(X_j|\overline{s}^{[i]})$ as 'Gaussian-like' ($\mathcal{N}(\overline{\mu}_j^{[i]}, \Sigma_j^{[i]})$), a natural choice of weighting is the Gaussian likelihood (as is done in the original FastSLAM algorithm[2,3]). The measurement of feature j, z_j, has likelihood denoted $w_j^{[i]}$, and calculated by:

$$w_j^{[i]} = |2\pi\mathbf{Q}|^{-\frac{1}{2}} exp\left\{-\frac{1}{2}(\overline{z}_j - \hat{\overline{z}}_j^{[i]})^T\mathbf{Q}^{-1}(\overline{z}_j - \hat{\overline{z}}_j^{[i]})\right\} \tag{15}$$

The overall likelihood of a specific particle, w_i, is the cumulative likelihoods of that particle's measurements:

$$w^{[i]} = \prod_j w_j^{[i]} \tag{16}$$

Periodically, the particles are resampled, in order to remove the least likely hypotheses, and focus on the most likely. This algorithm resamples based on a calculation of effective sample size N_{eff}:[4]

$$N_{eff} = \frac{1}{\sum_i (w^{[i]})^2} \tag{17}$$

When N_{eff} is less than half the total number of particles, resampling takes place. In order to resample efficiently, low-variance resampling is used.[3] After resampling, all particles are set to equal weight.

VI.C. Motion Updates

For each particle i, the proposed kinematic state $\overline{s}_{(k+1)}^{[i]}$ for the next time-step is drawn from the following proposal distribution:

$$\overline{s}_{(k+1)}^{[i]} \sim \begin{bmatrix} \mathcal{F}(\mu_{prop_{[7:9]}}^{[i]}, (\Sigma_{prop_{[1,1]}}^{[i]})^{-1}) \\ \mathcal{N}(\mu_{prop_{[4:12]}}^{[i]}, \Sigma_{prop_{[4:12,4:12]}}^{[i]}) \end{bmatrix} \tag{18}$$

This proposal distribution is a modified Fisher distribution over the Euler angles (as discussed in Section IV), and a Gaussian distribution over the other components of the kinematic state vector. It takes into account a motion model prediction ($f(\overline{s}_k^{[i]})$), *as well as* the latest measurements taken ($\overline{z}_{j(k+1)}$).[21] The covariance ($\Sigma_{prop}^{[i]}$) and mean ($\mu_{prop}^{[i]}$) of the proposal distribution are:

$$\Sigma_{prop}^{[i]} = \left[\sum_j \tilde{G}_s^{[i]^T} \left(Q_j^{[i]} \right)^{-1} \tilde{G}_s^{[i]} + P^{-1} \right]^{-1} \tag{19}$$

$$\mu_{prop}^{[i]} = \Sigma_{prop}^{[i]} \sum_j \tilde{G}_s^{[i]^T} \left(Q_j^{[i]} \right)^{-1} \left(\overline{z}_j - \tilde{\overline{z}}_j^{[i]} \right) + f(\overline{s}_k^{[i]}) \tag{20}$$

where:

$$\tilde{\overline{z}}_j^{[i]} = g(f(\overline{s}_k^{[i]}), \overline{\mu}_j^{[i]}) \tag{21}$$

$$\tilde{G}_{X_j} = \left. \frac{\partial g}{\partial X_j} \right|_{\left(f(\overline{s}_k^{[i]}), \overline{\mu}_j^{[i]} \right)} \tag{22}$$

$$\tilde{G}_s = \left. \frac{\partial g}{\partial \overline{s}} \right|_{\left(f(\overline{s}_k^{[i]}), \overline{\mu}_j^{[i]} \right)} \tag{23}$$

$$Q_j^{[i]} = R_{prop} + \tilde{G}_{X_j}^{[i]} \Sigma_j^{[i]} \tilde{G}_{X_j}^{[i]^T} \tag{24}$$

Once the kinematic states have been proposed, the filter then advances to the next time-step and begins the measurement update again.

VII. Feature Initialization

Because the measurement is bearings-only, a feature cannot be initialized with just one measurement. Two or more views of a feature are required before its associated EKF can be initialized.

In selecting the number of views to use at initialization, a balance must be struck between competing objectives. From the standpoint of initialization accuracy, it is better to have more views of a feature, since this is equivalent in general to having a wider 'baseline'. On the other hand, it is desirable to initialize a feature as quickly as possible (i.e., with less views), so that it is mapped and taken into account when weighting the particles. An initialization size of 4 views was empirically chosen for the experiments performed in this paper.

The initialization is performed via least squares minimization. Based on the measurement model, variables are rearranged to obtain a linear equation of the form $B = \mathbf{A}X_j$, where X_j is $\overline{X}_{j/cam}^C$, the 3-D position of the feature in target (body) coordinates. Alternatively, a more accurate nonlinear minimization technique could be used ('bundle-adjustment').[22] The strength of initialization via least squares is its speed.

Expanding Eq. 6, the relationship to transform feature positions into camera coordinates is expressed as:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{j/cam}^{C} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}^{C/I} \left(\begin{bmatrix} r_{11} & r_{21} & r_{31} \\ r_{12} & r_{22} & r_{32} \\ r_{13} & r_{23} & r_{33} \end{bmatrix}^{B/I} \begin{bmatrix} x \\ y \\ z \end{bmatrix}_{j/targ}^{B} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}_{targ/cam} \right) \tag{25}$$

The above matrix equation can be broken into three scalar equations:

$$x_{j/cam}^C = a * x_{j/targ}^B + b * y_{j/targ}^B + c * z_{j/targ}^B + d \tag{26}$$

$$y_{j/cam}^C = e * x_{j/targ}^B + f * y_{j/targ}^B + g * z_{j/targ}^B + h \tag{27}$$

$$z_{j/cam}^C = l * x_{j/targ}^B + m * y_{j/targ}^B + n * z_{j/targ}^B + o \tag{28}$$

The coefficients in Eqns. 26-28 are:

$$\begin{aligned}
a &= r_{11}^{C/I} r_{11}^{B/I} + r_{12}^{C/I} r_{12}^{B/I} + r_{13}^{C/I} r_{13}^{B/I} \\
b &= r_{11}^{C/I} r_{21}^{B/I} + r_{12}^{C/I} r_{22}^{B/I} + r_{13}^{C/I} r_{23}^{B/I} \\
c &= r_{11}^{C/I} r_{31}^{B/I} + r_{12}^{C/I} r_{32}^{B/I} + r_{13}^{C/I} r_{33}^{B/I} \\
d &= r_{11}^{C/I} t_x + r_{12}^{C/I} t_y + r_{13}^{C/I} t_z \\
e &= r_{21}^{C/I} r_{11}^{B/I} + r_{22}^{C/I} r_{12}^{B/I} + r_{23}^{C/I} r_{13}^{B/I} \\
f &= r_{21}^{C/I} r_{21}^{B/I} + r_{22}^{C/I} r_{22}^{B/I} + r_{23}^{C/I} r_{23}^{B/I} \\
g &= r_{21}^{C/I} r_{31}^{B/I} + r_{22}^{C/I} r_{32}^{B/I} + r_{23}^{C/I} r_{33}^{B/I} \\
h &= r_{21}^{C/I} t_x + r_{22}^{C/I} t_y + r_{23}^{C/I} t_z \\
l &= r_{31}^{C/I} r_{11}^{B/I} + r_{32}^{C/I} r_{12}^{B/I} + r_{33}^{C/I} r_{13}^{B/I} \\
m &= r_{31}^{C/I} r_{21}^{B/I} + r_{32}^{C/I} r_{22}^{B/I} + r_{33}^{C/I} r_{23}^{B/I} \\
n &= r_{31}^{C/I} r_{31}^{B/I} + r_{32}^{C/I} r_{32}^{B/I} + r_{33}^{C/I} r_{33}^{B/I} \\
o &= r_{31}^{C/I} t_x + r_{32}^{C/I} t_y + r_{33}^{C/I} t_z
\end{aligned} \tag{29}$$

Eqs. 26, 27, and 28 are plugged into the measurement model (Eq. 5), and rearranged to express it as a linear equation on the feature's body coordinates:

$$\begin{gathered}
\begin{bmatrix} u_j * o - foc * d \\ v_j * o - foc * h \end{bmatrix}_k = \begin{bmatrix} foc * a - u_j * l & foc * b - u_j * m & foc * c - u_j * n \\ foc * e - v_j * l & foc * f - v_j * m & foc * g - v_j * n \end{bmatrix}_k X_j \\
B_k = \mathbf{A_k} X_j
\end{gathered} \tag{30}$$

In Eq. 30, the index k indicates the terms are based on measurement and vehicle data at a specific time. Taking data at multiple time steps, the matrices above are stacked:

$$\begin{gathered}
\begin{bmatrix} B_{k_1} \\ B_{k_2} \\ B_{k_3} \\ \vdots \end{bmatrix} = \begin{bmatrix} \mathbf{A_{k_1}} \\ \mathbf{A_{k_2}} \\ \mathbf{A_{k_3}} \\ \vdots \end{bmatrix} X_j \\
B = \mathbf{A} X_j
\end{gathered} \tag{31}$$

Finally, least squares estimation returns the estimate of feature position:

$$\hat{X}_j = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T B \tag{32}$$

This estimate is used to initialize the mean of the EKF, $\overline{\mu}_j^{[i]}$.

To initialize the covariance of the feature's location ($\Sigma_j^{[i]}$), the Jacobian matrices (G_k) are determined by evaluating Eq. 22 at $\overline{\mu}_j^{[i]}$ for each of the poses $\overline{s}_k^{[i]}$ for which a measurement was recorded. The covariance is then initialized as given in Eq. 33 below.

$$\Sigma_j^{[i]} = r^2 \left(\begin{bmatrix} G_{k_1} & G_{k_2} & G_{k_3} & \cdots \end{bmatrix} \begin{bmatrix} G_{k_1} \\ G_{k_2} \\ G_{k_3} \\ \vdots \end{bmatrix} \right)^{-1} \tag{33}$$

VIII. Vision Front End

The algorithm described in the previous section depends on consistently tracking features over several images. The algorithm assumes all correspondences are correct, so feature correspondence needs to be handled 'upstream' of the filter. Any type of point features can be used, as long as correspondence between images is correctly performed.

For the demonstrations presented in this paper, SIFT features were used.[6] The unique 128 element SIFT vector greatly aids the correspondence problem. SIFTs are somewhat robust to orientation (including out-of-plane) changes, making them useful for tracking on a tumbling body.

When a new image is taken, the latest SIFTs are extracted and correspondence matches are attempted. For each of these observed features, the Euclidean distance (in the 128-dimensional descriptor space) is computed between it and a set of SIFTs from the reference library. These reference SIFTs each correspond to an already observed feature. If the ratio of the descriptor Euclidean distances to the closest reference SIFT and second closest reference SIFT is less than 0.6, then a match is declared. This is the classic SIFT correspondence algorithm, as described in.[6]

In this paper two additional logical checks are incorporated to improve match performance and eliminate false positives. First, only reference SIFTs within a given pixel distance of the new SIFT are considered as candidates for potential correspondence. Second, to be considered a match, the dot product between the new SIFT vector and reference SIFT vector must be greater than a threshold. This ensures that, in regions of only a few reference SIFTs, a match is not selected when none of the reference SIFTs is at all similar to the new SIFT. Good values for these parameters were determined empirically; the pixel distance cutoff was set to 15 pixels, and the dot product threshold was set to 0.8.

A few data structures are necessary in order to track SIFTs efficiently over time, while throwing out those SIFTs that are only observed once or sporadically. Two tables of SIFT vectors are maintained in memory: a table for the latest features observed (the 'new table'), and another table to store the reference SIFT vectors ('the library'). The library also contains a vector of counters, keeping track of the number of time steps since a particular feature was observed last. At the first time step, all the SIFTs from the new table are copied into the library, and the counters for these features are set to 0.

At each successive time step, the counter is incremented. Then, each of the SIFTs that are in the new table is compared against the library to find a match, via the method described above. If a match is found, the SIFT vector overwrites its match in the table, the counter is set to 0, and the pixel coordinates of the SIFT are passed to the SPEAR algorithm.

The counter is used for pruning the library. If every SIFT were kept for comparison to future SIFTs, the library would grow unmanageably large. Instead, a SIFT is removed from the library if it has not be observed in the last 8 images.

IX. Computation Issues

The end goal of the application of this algorithm is a pipeline providing real-time information on the target's pose and shape. Therefore it is essential that both the SIFT detection step, as well as the particle filter, run at adequate rate in software.

SIFT detection can be performed at high rate, utilizing Field Programmable Gate Arrays (FPGAs)[23] or Graphical Processing Units (GPUs).[24]

Particle filters of comparable complexity have been run at 10 Hz,[25] well within feasible limits of operation for the application in this paper.

X. Experiments

To demonstrate the performance of the SPEAR algorithm, several experiments are presented. To analyze the algorithm separately from the vision processing, a synthetic target (with known locations of the body features) is analyzed (X.A). To incorporate the vision processing, a mock target (moving at known rotation rate) is observed over time with a camera in air (X.B). Finally, the algorithm is deployed on a target of interest in the field: a moving underwater target (X.C).

X.A. Synthetic Target

The purpose of the synthetic test is to show the veracity of the shape reconstruction and pose tracking via the algorithm described in section VI. Knowing the true positions of the features on the target allows for testing of the accuracy of the feature reconstruction. The synthetic target is generated in MATLAB. It is a set of 100 points randomly located on a box-shaped object (with side length of 2 units), which is undergoing rotation and translation. Figure 6 shows a 3-D plot of the true box shape (squares), along with the reconstructed feature locations (crosses). Figure 7 shows the root-mean-square error of the feature position estimates versus time, for the maximum likelihood particle. These plots show the accuracy of both the shape reconstruction and its location in the reference frame.

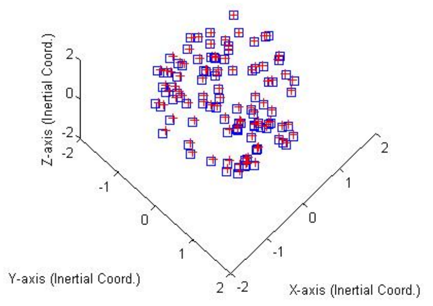

Figure 6. True vs. Reconstructed Synthetic Target

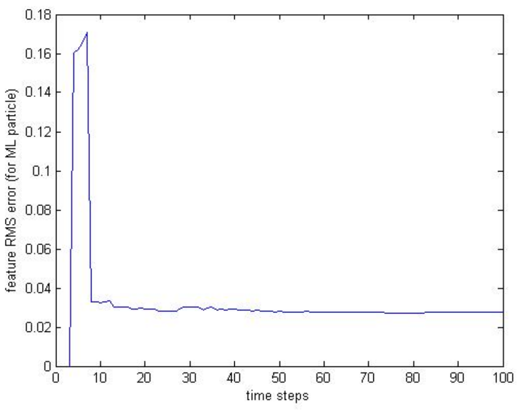

Figure 7. RMS Error of Feature Positions vs. Time (ML)

The other set of variables that is estimated is the kinematic states. Figures 8 and 9 show the synthetic target's orientation and angular rates versus time (blue is truth, green is estimates).

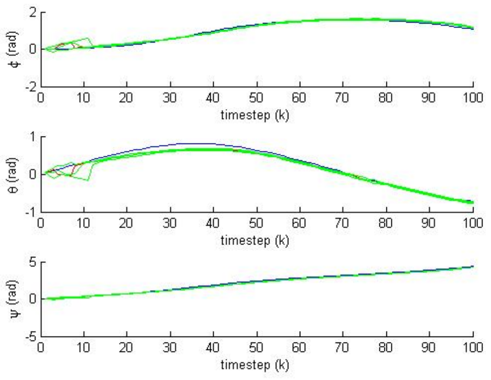

Figure 8. Orientation vs. Time (Synthetic Target)

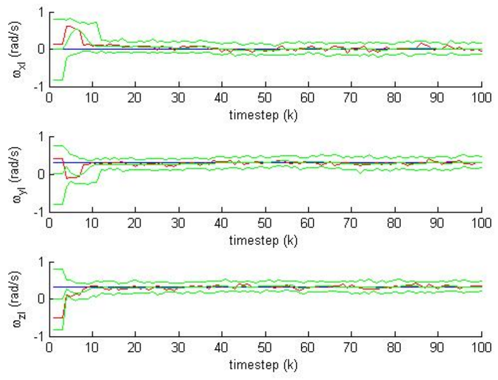

Figure 9. Angular Rate vs. Time (Synthetic Target)

X.B. Physical Target in Air

To test the SPEAR algorithm using actual images, a series of images were taken of a LEGO^{TM}'target'. The target was observed against a black background, and was meant to simulate a tumbling satellite in space. A selection of the target images is shown in Figure 10.

The target was placed on a rotating base, which can be rotated to specific angles. For each image in the sequence, the target was rotated by 1°. Images were taken using a Matrox Morphis framegrabber and a calibrated[26] Sony XC-999 Camera.

Because of the nature of the hardware set-up (i.e., a base which only rotates about one axis, and a fixed camera), the only test possible was a 1-axis rotation. Results are presented below, showing the particle filter's successful reconstruction of the target body (Figure 11).

Figures 12 and 13 are plots of Euler angles and angular rates vs. time (blue is truth, green is estimates). They show SPEAR tracking the correct rotation on one axis.

X.C. Underwater Target

The underwater target demonstration shows the SPEAR algorithm reconstructing and tracking in an actual field application. Field testing was conducted in conjunction with the Monterey Bay Aquarium Research Institute (MBARI). Using a Remotely Operated Vehicle (ROV) (shown in Figure 14), underwater tethered targets were imaged.

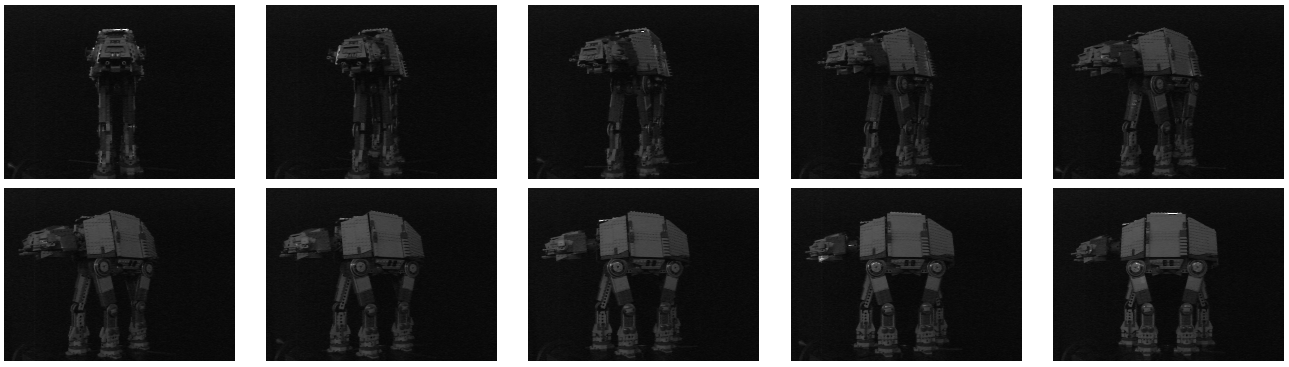

Figure 10. Sample Views from Lab Image Sequence

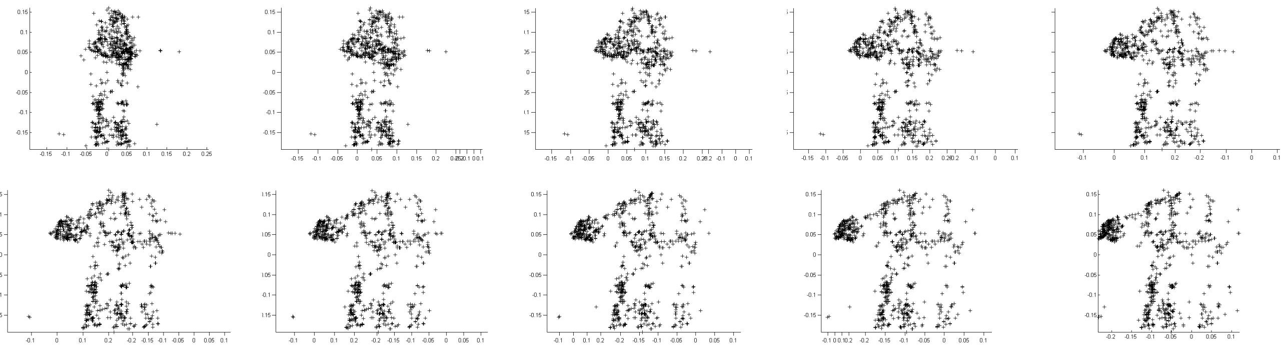

Figure 11. Reconstruction of the Physical Target

An example of an underwater target of interest is shown in Figure 15. No *a priori* information exists, and autonomous proximity operations are desired. The purpose of the field experiments is to track relative pose in the water while simultaneously reconstructing the shape of the target (a precursor to positioning control algorithms).

A set of images was obtained with the main science camera on the ROV, at 10 Hz rate. Samples of the sequence of 150 images, at 1 second intervals, are shown in Figure 16.

Figures 17, 18, 19, 20, and 21 present the results of applying particle filter SPEAR on the set of images above. The reconstructed target is shown in Figure 17; compared with the views of the target shown in Figure 16, it shows accurate representation of the 3-D shape.

The relative pose estimate time history is shown in Figures 18-21. Figure 18 shows the Euler angles, Figure 19 shows the angular rates, Figure 20 shows the position, and Figure 21 shows the velocity.

XI. Summary

A technique has been presented to estimate a target's pose and 3-D shape simultaneously. The algorithm is based on an efficient particle filter algorithm from the SLAM community (FastSLAM). In particular, the algorithm described here was applied to solving the SPEAR problem with a bearings-only measurement using no *a priori* information about the target.

The results show the successful estimation of each target's shape and pose, validating the SPEAR algorithm as an effective tool for the SFM problem.

Acknowledgments

This research was partially funded by a Stanford Graduate Fellowship and NASA grant #Z627401-A. Support was also provided by the Monterey Bay Aquarium Research Institute (MBARI). The authors thank Kiran Murthy, Stephen Russell, Daniel Sheinfeld, Deborah Meduna, and Peter Kimball for useful discussions.

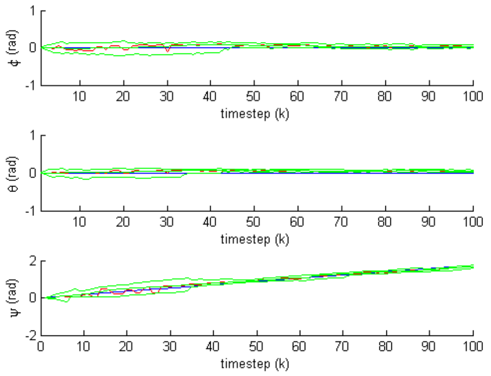

Figure 12. Orientation vs. Time (LEGO Target)

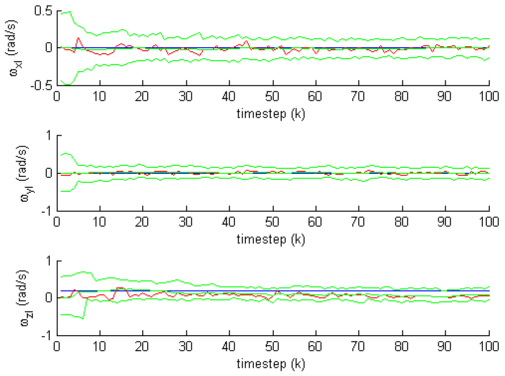

Figure 13. Angular Rate vs. Time (LEGO Target)

Figure 14. ROV *Ventana*

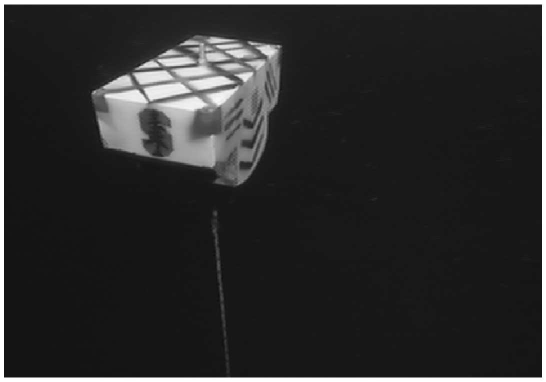

Figure 15. Example of an Underwater Target

References

[1]Dissanayake, M., Newman, P., Clark, S., Durrant-Whyte, H., and Csorba, M., "A solution to the simultaneous localization and map building (SLAM) problem," *IEEE Transactions on Robotics and Automation*, Vol. 17, No. 3, Jun 2001, pp. 229–241.

[2]Montemerlo, M., Thrun, S., Koller, D., and Wegbreit, B., "FastSLAM: a factored solution to the simultaneous localization and mapping problem," *Eighteenth National Conference on Artificial Intelligence*, American Association for Artificial Intelligence, Menlo Park, CA, USA, 2002, pp. 593–598.

[3]Thrun, S., Burgard, W., and Fox, D., *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*, MIT press, Cambridge, Massachusetts, USA, 2005.

[4]Doucet, A., De Freitas, N., and Gordon, N., *Sequential Monte Carlo Methods in Practice*, Springer-Verlag, 2001.

[5]Murphy, K. P., "Bayesian map learning in dynamic environments," *In Neural Info. Proc. Systems (NIPS)*, 1999, pp. 1015–1021.

[6]Lowe, D., "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, Vol. 60, No. 2, 2004, pp. 91–110.

[7]Inaba, N. and Oda, M., "Autonomous satellite capture by a space robot: world first on-orbit experiment on a Japanese robot satellite ETS-VII," *Proceedings of ICRA '00*, Vol. 2, 2000, pp. 1169–1174.

[8]English, C., Ruel, S., Melo, L., Church, P., and Maheux, J., "Development of a practical 3D automatic target recognition and pose estimation algorithm," *Proc. SPIE*, Vol. 5426, 2004, pp. 112–123.

[9]Lichter, M. and Dubowsky, S., "State, shape, and parameter estimation of space objects from range images," *Proceedings ICRA '04*, Vol. 3, April-1 May 2004, pp. 2974–2979 Vol.3.

[10]Ma, Y., Soatto, S., Kosecka, J., and Sastry, S. S., *An Invitation to 3-D Vision: From Images to Geometric Models*, Springer, 2003.

[11]Broida, T., Chandrashekhar, S., and Chellappa, R., "Recursive 3-D motion estimation from a monocular image sequence," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 26, No. 4, Jul 1990, pp. 639–656.

[12]Azarbayejani, A. and Pentland, A., "Recursive estimation of motion, structure, and focal length," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 17, No. 6, Jun 1995, pp. 562–575.

[13]Chiuso, A., Favaro, P., Jin, H., and Soatto, S., "Structure from Motion Causally Integrated Over Time," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24, No. 4, 2002, pp. 523–535.

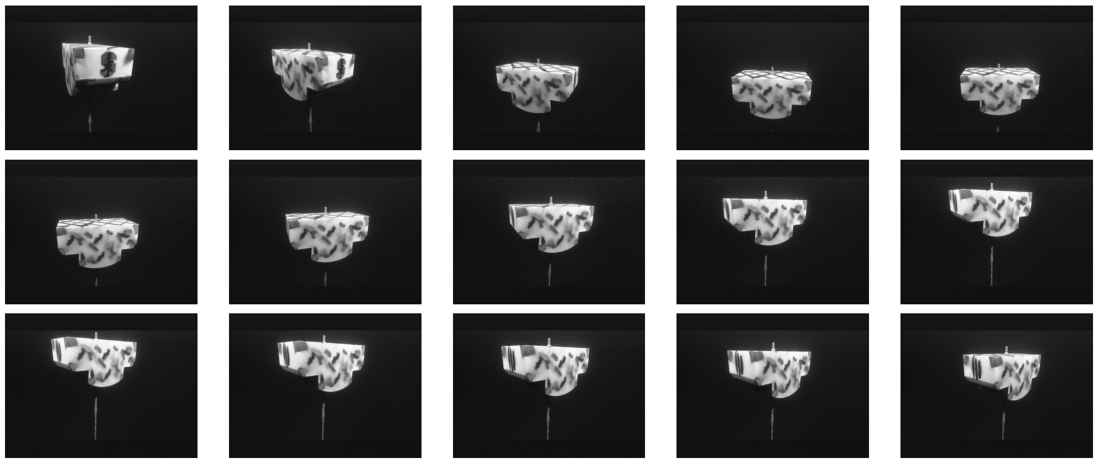

Figure 16. Sample Views from Underwater Image Sequence

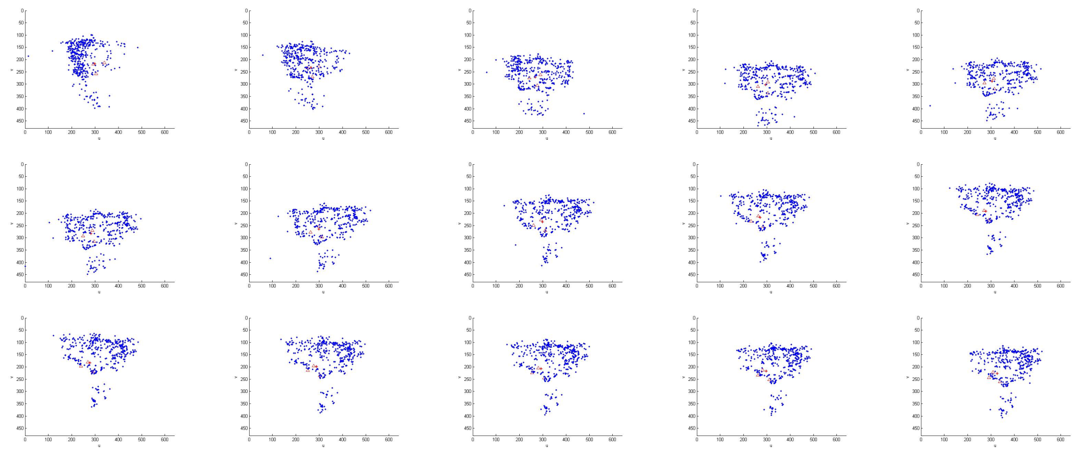

Figure 17. Reconstruction of the Underwater Target

[14]Chang, P. and Hebert, M., "Robust tracking and structure from motion through sampling based uncertainty representation," *Proceedings of ICRA '02*, May 2002.

[15]Pupilli, M. and Calway, A., "Real-time camera tracking using a particle filter," *In Proc. British Machine Vision Conference*, 2005, pp. 519–528.

[16]Marimon, D., Maret, Y., Abdeljaoued, Y., and Ebrahimi, T., "Particle filter-based camera tracker fusing marker and feature point cues," *Proc. SPIE*, Vol. 6508, 2007.

[17]Nister, D., "Preemptive RANSAC for live structure and motion estimation," *Machine Vision and Applications*, Vol. 16, No. 5, 2005, pp. 321–329.

[18]Wolf, D. and Sukhatme, G., "Mobile Robot Simultaneous Localization and Mapping in Dynamic Environments," *Autonomous Robots*, Vol. 19, No. 1, 2005, pp. 53–65.

[19]Wang, C.-C., Thorpe, C., Thrun, S., Hebert, M., and Durrant-Whyte, H., "Simultaneous Localization, Mapping and Moving Object Tracking," *The International Journal of Robotics Research*, Vol. 26, No. 9, 2007, pp. 889–916.

[20]Fisher, N., Lewis, T., and Embleton, B., *Statistical analysis of spherical data*, Cambridge Univ Pr, 1993.

[21]Montemerlo, M., Thrun, S., Koller, D., and Wegbreit, B., "FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges," *International Joint Conference on Artificial Intelligence*, Vol. 18, 2003, pp. 1151–1156.

[22]Deans, M. and Hebert, M., "Experimental Comparison of Techniques for Localization and Mapping using a Bearings Only Sensor," *Proc. of the Seventh International Symposium on Experimental Robotics*, December 2000.

[23]Se, S., Ng, H., Jasiobedzki, P., and Moyung, T., "Vision based modeling and localization for planetary exploration rovers," *In Proc. International Astronautical Congress*, 2004.

[24]Sinha, S., Frahm, J., Pollefeys, M., and Genc, Y., "GPU-based Video Feature Tracking and Matching," *EDGE, Workshop on Edge Computing Using New Commodity Architectures*, Vol. 278, 2006.

[25]Hoffmann, G. M., *Autonomy for Sensor-Rich Vehicles: Interaction between Sensing and Control Actions*, Ph.D. thesis, Dept. Aero. Astro., Stanford Univ., Stanford, CA, 2008.

[26]Bouguet, J., "Camera calibration toolbox for Matlab." Website.

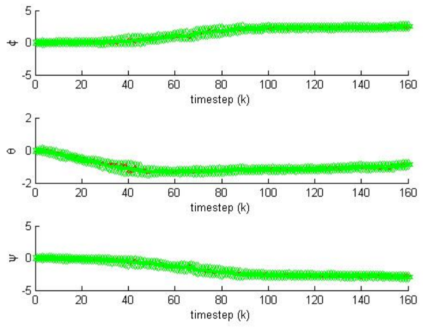

Figure 18. Orientation vs. Time (Underwater Target)

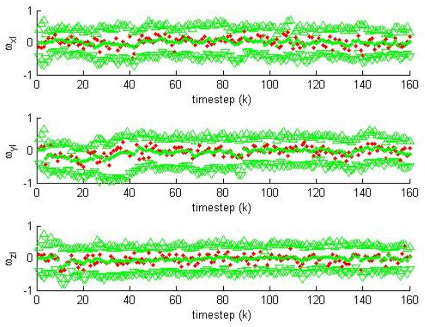

Figure 19. Angular Rate vs. Time (Underwater Target)

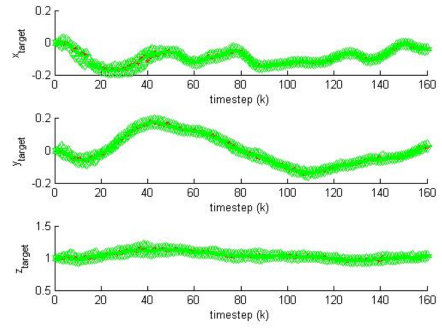

Figure 20. Position vs. Time (Underwater Target)

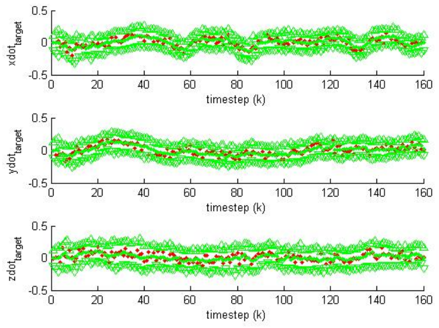

Figure 21. Velocity vs. Time (Underwater Target)