6DOF Pose Estimation using 3D Sensors

Bart Verzijlenberg and Michael Jenkin
York University, Department of Computer Science and Engineering
4700 Keele St., Toronto, ON, Canada, M3J 1P3
{bartv, jenkin}@cse.yorku.ca

Abstract—Pose estimation is an important capability for mobile agents. A wide variety of solutions have been proposed, but work in the literature has focused primarily on solutions for robots whose mobility is restricted to the ground plane. In this work we present a framework for 6DOF pose estimation. Normally the increased computational cost associated with this higher dimensional space makes pose estimation intractable. The approach presented here addresses the computational issues associated with the higher dimensional problem by decoupling orientation estimation from position estimation. Assuming that orientation can be estimated separately from position allows efficient methods to be used for the (unimodal) orientation estimate, while more sophisticated methods are used for the position estimate. Although similar to Rao-Blackwellization, the approach is essentially reversed. Results on real and simulated datasets and a comparison with a naïve 6DOF filter are presented.

I. INTRODUCTION

Advanced autonomous robotic platforms are increasingly deployed in outdoor environments; these environments can be considered more complex than the traditional office or lab environment in that they are less structured and allow a wider variety of robot motion. The problems encountered by vehicles like the robot KROY shown in Figure 1 are typical. The device’s pose is given by both its position and orientation. There is no ground plane upon which computations can rely and the vehicle is free to achieve essentially arbitrary poses within the six degree of freedom (DOF) pose space of the vehicle. The lack of a ground plane also reduces the efficacy of powerful kinematic motion models to shape pose estimation distributions.

In order for an autonomous system to plan its path through an operating environment, an estimate of its position and orientation is important, if not critical. Traditional approaches to pose estimation (e.g., [6], [9], [1], [18], [2]) typically focus on 2D environments, such as offices, where the device’s limited mobility simplifies the pose estimation process. Others assume an underlying 2D environment structure in a 3D world (e.g., [17]). Here we present an approach for the efficient global pose estimation of an unconstrained 6DOF device using sophisticated 3D sensors, operating freely in true 3D environments. Unlike a ground contact robot which is typically provided with a flat map of its environment, the 6DOF devices considered here are provided with a fully 3D map. A variety of sophisticated sensors are available for sensing a robot’s 3D environment and such environments are increasingly of interest in robotic applications. In addition to 6DOF robots operating in full 3D environments, there exist a range of intermediate environments which, although not fully 3D, provide many of the complexities of such environments. Although the work presented here concentrates on the sensing abilities of devices such as the AQUA [10] and C2SM [19] robot platforms it has a wider application to 3D environments with robots capable of collecting 3D environmental map data.

II. BACKGROUND

The problem of local pose estimation is typically phrased in terms of a recursive state estimation process and is implemented using either Kalman or particle filtering (see [14], [8], [14] for examples). The Bayes filter [16] provides a general framework within which to recursively estimate the state of a system given a series of measurements and motion commands. A common assumption when using the Bayes filter is the Markov assumption [3], which in this context states that past and future data are (conditionally) independent if the current state is known. Application of the Bayes filter (with the Markov assumption) to the problem of robot localization is known as Markov Localization [3]. The general form for Markov Localization is given by

\[
bel(x_t) = \eta \ p(z_t|x_t) \ \int p(x_t|x_{t-1}, u_{t-1}) \ \text{bel}(x_{t-1}) \ dx_{t-1}.
\]

Here \(bel(x_t)\) is the belief the robot has regarding its state \(x_t\) at time \(t\), conditioned on collected data \(y_t\). The data collected by the robot \(y_t = (u_{t-1}, z_t)\) is composed of a sensing input \(z_t\) and a control command \(u_{t-1}\). In general, the robot’s state \(x_t\) is composed of the robot position and orientation. The belief \(bel(x_0)\) is often initialized either uniformly across the space, or clustered around some \(a\)-priori estimate.

Particle filters have emerged as a very popular approach to the pose estimation problem (e.g., [13], [7], [11]), as they address both memory and computational requirements for 2D robots. Monte-Carlo Localization (MCL) is a particle filtering approach applied to robot localization (see [3], [18] for examples). Particle filters estimate the desired probability distribution using a sample-based approach.

A critical problem in the development of pose estimation algorithms is that the cost of the algorithm typically grows exponentially with the size of the state space. For Kalman filters this growth is seen in the size of the covariance...
resulting in significant computational savings.

Similarity, particle-filter-based approaches must deal with sampling an environment that grows exponentially with the size of the problem. Approaches that are successful in 3DOF pose estimation are generally not attractive for the 6DOF version of the problem. One approach to solving the higher DOF version of the problem is to decompose the problem into two smaller problems and then integrate these results, or conditioning one part of the problem on another.

One approach that has proven particularly successful in this regard is Rao-Blackwellization. Rao-Blackwellization is an approach that improves the performance of a particle filter when the structure of the state-space model allows for some of the parameters to be marginalized out, conditioned on the remaining parameters [5]. Rao-Blackwellized particle filters are used to estimate the posterior using a parametric pdf to represent a subset of the parameters, conditioned on the remaining, particle sampled parameters. It is assumed that the parametric distribution can be computed relatively efficiently. Only the remaining parameters need to be sampled, which can be accomplished using particle filters [3], [5]. An alternative approach popular in the Kalman filter literature is to decompose the problem into independent sub-problems and then to solve these problems independently.

Here we take advantage of the fact that many 6DOF vehicles are equipped with an independent orientation sensor that we can decouple from positional sensors. This enables the pose estimation formalism to be similarly decoupled resulting in significant computational savings.

III. 6DOF GLOBAL POSE ESTIMATION

Vehicles operating in a 3D environment must maintain a host of parameters specifying the vehicle’s state. Instead of using a single, monolithic system to estimate every component of the state, various sub-systems can be used to monitor individual parameters. This allows an individual sensor to be focused on the task it does well, instead of attempting to estimate all parameters.

Let the state of the robot at time $t$ be defined by $X_t = \{x_t, \theta_t\}$, where $x_t$ denotes the position and $\theta_t$ denotes the orientation of the robot. Further, let $Y_t$ denote relevant evidence collected by the robot at time $t$, and $Y_{m:n}$ the evidence between times $m$ and $n$. In very general terms, the belief of the robot can then be stated as

$$Bel(X_t) \triangleq p(X_t|Y_{0:t}) = p(x_t, \theta_t|Y_{0:t})$$ (1)

It is desirable to split this term into separate subcomponents that might be computed more efficiently than attempting to establish the full pdf directly. We can decompose this monolithic pdf using Rao Blackwellization. A typical application of Rao Blackwellization separating the position and orientation (see [5]) results in either

$$p(x_t, \theta_t|Y_{0:t}) = p(\theta_t|x_t, Y_{0:t}) p(x_t|Y_{0:t}) \quad (2)$$

or

$$= p(x_t|\theta_t, Y_{0:t}) p(\theta_t|Y_{0:t}) \quad (3)$$

depending on the assumption as to which can be established independently, position or orientation. Equation 2 would be appropriate if the position likelihood, $p(x_t|Y_{0:t})$, can be decoupled from prior knowledge of the device’s orientation state. Of interest here is the manipulation given in Equation 3. For a device such as KROY, shown in Figure 1, the orientation estimate in $p(\theta_t|x_t, Y_{0:t})$, obtained by onboard orientation sensors (IMU’s), is not generally affected by position (ignoring the effects of external noise sources such as local magnetic forces on orientation sensors). This suggests that a potentially effective mechanism for establishing $p(x_t, \theta_t|Y_{0:t})$ is to decouple $\theta_t$ and $x_t$ using Equation 3. The estimate $p(\theta_t|Y_{0:t})$ can then be used when sampling $p(x_t|\theta_t, Y_{0:t})$. This is the basic approach taken here; the details of the approach are outlined in the following sections.

A. Filter derivation

A recursive filter is desired that computes one portion of the state efficiently while using a sampling approach on the remaining state. Given evidence $Y_{0:t} = \{x_{0:t-1}, z_{1:t}, i_{1:t}\}$, let $u_t$ be the control inputs for the robot at time $t$, $z_t$ be the sensor readings used as evidence for the robot’s position at time $t$, and let $i_t$ be the IMU data used for orientation estimation at time $t$. Assume that control $u_{t-1}$ is executed at state $X_{t-1}$, and that this commanded motion results in state $X_t$. The sensor data $z_t$ and $i_t$ are subsequently collected while the device is in state $X_t$. Note that neither $z_0$ nor $i_0$ are collected, since it is assumed that $u_0$ will be executed prior to the first data collection; similarly, there is no $u_t$.

Consider the $Bel(X_t)$ in Equation 1: after applying the theorem of total probability over $\theta_{t-1}$, the belief (or joint
Finally, applying the theorem of total probability to the posterior distribution
\[ \text{Bel}(X_t) \triangleq p(X_t | Y_{0:t}) = p(x_t, \theta_t | Y_{0:t}) \]
\[ = p(Y_{0:t})^{-1}p(x_t, \theta_t | Y_{0:t}) \]
\[ = p(Y_{0:t})^{-1} \int p(x_t, \theta_t, Y_{0:t} | \theta_t) p(\theta_t) d\theta_t. \]
and applying the chain rule of conditional probability
\[ \text{Bel}(X_t) = \int \frac{p(x_t, \theta_t | Y_{0:t}, \theta_t) p(Y_{0:t} | \theta_t) p(\theta_t) d\theta_t}{p(Y_{0:t})}. \]

Observing that \( p(a|b) = p(b|a)p(a)/p(b) \)
\[ \text{Bel}(X_t) = \int p(x_t, \theta_t | Y_{0:t}, \theta_t) p(\theta_t | Y_{0:t}) d\theta_t. \] (4)

Equation 4 may now be modified by applying the chain rule of probability to expand \( p(x_t, \theta_t | Y_{0:t}, \theta_t) \). Also expanding \( Y_{0:t} \triangleq (u_{0:t-1}, x_{1:t}, z_{1:t}) \) results in:
\[ p(x_t, \theta_t | Y_{0:t}, \theta_t) = p(\theta_t | Y_{0:t}) p(z_{1:t}, x_t, \theta_t, u_{0:t-1}, i_{1:t}, z_{1:t-1}). \]

By writing \( \alpha = (\theta_t, \theta_t-1, u_{0:t-1}, i_{1:t}, z_{1:t-1}) \), and expanding
\[ p(x_t, \theta_t | Y_{0:t}, \theta_t) = p(\theta_t | Y_{0:t}) p(z_{1:t}, x_t, \alpha) p(x_t | \alpha)^{-1}. \] (5)

Expanding \( \alpha \) in \( p(x_t | \alpha) \) allows equation 5 to be rewritten as
\[ p(x_t, \theta_t | Y_{0:t}) = p(\theta_t | Y_{0:t}) p(z_{1:t}, x_t, \alpha) p(z_{1:t} | \alpha)^{-1} \]
\[ = p(\theta_t | Y_{0:t}) p(z_{1:t}, x_t, \alpha) p(z_{1:t} | \alpha)^{-1} p(x_t | \alpha)^{-1}. \] (6)

Where \( \beta = (\theta_t, \theta_t-1, u_{0:t-1}, i_{1:t-1}, z_{1:t-1}) \), and \( \alpha = (\beta, i_t) \).

Expanding \( p(x_t | \beta) \) results in
\[ p(x_t, \beta | Y_{0:t}) = p(\theta_t | Y_{0:t}) p(z_{1:t}, x_t, \alpha) p(z_{1:t} | \alpha)^{-1} \]
\[ = p(\theta_t | Y_{0:t}) p(z_{1:t}, x_t, \alpha) p(z_{1:t} | \alpha)^{-1} p(x_t | \beta)^{-1}. \] (7)

Finally, applying the theorem of total probability to \( p(x_t | \beta) \) in terms of \( x_{t-1} \) results in the following equivalence:
\[ p(x_t, \beta | Y_{0:t}) = p(\theta_t | Y_{0:t}) p(z_{1:t}, x_t) p(z_{1:t} | \alpha)^{-1} \]
\[ p(i_t | \beta)^{-1} \int p(x_t | x_{t-1}, \beta) p(x_{t-1} | \beta) dx_{t-1}. \] (8)

Note that the derivation of Equation 8 relies only on properties of conditional probability. Substituting Equation 8 into Equation 4 obtains
\[ \text{Bel}(X_t) = \int p(\theta_t | Y_{0:t}) p(z_{1:t}, x_t, \alpha) p(z_{1:t} | \alpha)^{-1} \]
\[ p(i_t | \beta)^{-1} \int p(x_t | x_{t-1}, \beta) p(x_{t-1} | \beta) dx_{t-1} \]
\[ p(\theta_t-1 | Y_{0:t}) d\theta_t-1. \] (9)

It is possible to simplify several of these distributions under a Markov assumption. Based on the assumption that the state \{\( x_t, \theta_t \)\} is complete at time \( t \) when considering the full pose, and that \{\( \theta_t \)\} is complete when considering only the orientation, then all data preceding those states may be ignored in the distributions of equation 9. The measurement equations and (associated normalizers) can then be simplified as follows:
\[ p(z_t | x_t, \alpha) p(z_t | \alpha)^{-1} = p(z_t | x_t, \theta_t) p(z_t | \theta_t)^{-1} \] (10)
\[ p(i_t | x_t, \beta) p(i_t | \beta)^{-1} = p(i_t | x_t, \theta_t) p(i_t | \theta_t)^{-1}. \] (11)

Note in Equation 11 that a further simplification is possible if the assumption that \( i_t \) is independent of \( x_t \) holds. Such an assumption is not necessarily true, i.e., magnetic sensors may behave differently near a ship’s hull. Nonetheless, in certain situations it may reduce the required computational effort; this potential simplification is discussed in more detail below.

Further, \( p(x_t | x_{t-1}, \beta) \) contains the complete state \{\( x_{t-1}, \theta_{t-1} \)\}, allowing us to ignore most of the parameters (with the exception of \( u_{t-1} \) since it occurs after the complete state). Although the state \( \theta_t \) occurs after the (assumed) complete state, we ignore it, since the orientation of a device at time \( t \) does not affect the likelihood of having arrived at \( x_{t-1} \). Only \( \theta_{t-1} \) is needed for a complete state in \( p(\theta_t | Y_{0:t}) \), since we assume that orientation may be estimated independently of \( x_{t-1} \). Therefore only the data \{\( u_{t-1}, i_t, z_t \)\} that occurs after the complete state must still be considered. Any evidence related to the device’s position may also be eliminated, again based on the assumption that orientation is not affected by position. As a result we may also ignore \( z_t \), since by definition it contains no information related to the device’s orientation. This then results in:
\[ p(x_t | x_{t-1}, \beta) = p(x_t | x_{t-1}, u_{0:t-1}, z_{1:t-1}, i_{1:t-1}, \theta_{t-1}, \theta_t, i_t) \]
\[ = p(x_t | x_{t-1}, \theta_{t-1}, u_{t-1}) \] (12)
and
\[ p(\theta_t | \theta_{t-1}, Y_{0:t}) = p(\theta_t | \theta_{t-1}, u_{0:t-1}, i_{1:t-1}, z_{1:t-1}) \]
\[ = p(\theta_t | \theta_{t-1}, u_{t-1}, i_t). \] (13)

Substituting these equations back into Equation 9 results in
\[ \text{Bel}(X_t) = \int p(\theta_t | \theta_{t-1}, u_{t-1}, i_t) p(z_t | x_t, \theta_t) p(z_t | \theta_t)^{-1} \]
\[ p(i_t | \theta_t)^{-1} \int p(x_t | x_{t-1}, \theta_{t-1}, u_{t-1}) p(x_{t-1} | \theta_{t-1}, u_{t-1}, i_{1:t-1}, z_{1:t-1}) \]
\[ dx_{t-1} p(\theta_{t-1} | Y_{0:t}) d\theta_{t-1}. \]

The terms \( p(z_t | x_t, \theta_t), p(z_t | \theta_t)^{-1}, p(i_t | x_t, \theta_t) \) and \( p(i_t | \theta_t)^{-1} \) are constant with respect to the integral \( d\theta_{t-1} \) and may therefore be moved out. Moving \( p(\theta_{t-1} | Y_{0:t}) \) into
the inner integral results in:

\[
Bel(X_t) = p(z_t|x_t, \theta_t) p(\theta_t|\theta_{t-1})^{-1} p(i_t|x_t, \theta_t) p(i_t|\theta_t)^{-1} \\
\int p(\theta_t|\theta_{t-1}, u_{t-1}, i_t) \int p(x_t|x_{t-1}, \theta_{t-1}, u_{t-1}) \\
p(x_{t-1}|\theta_{t-1}, u_{0:t-1}, z_{1:t-1}, i_{1:t-1}) \\
p(\theta_{t-1}|Y_{0:t}) \, dx_{t-1} \, d\theta_{t-1}.
\]

Note that the \(u_{t-1}\) term in \(p(x_{t-1}|\theta_{t-1}, u_{0:t-1}, z_{1:t-1}, i_{1:t-1})\) is executed after state \(x_{t-1}\), while the state \(\theta_t\) occurs after \(x_{t-1}\), allowing both to be safely removed. We may then simplify the last two terms of equation 14 as follows:

\[
p(x_{t-1}|\theta_{t-1}, u_{0:t-2}, z_{1:t-1}, i_{1:t-1}) \, dx_{t-1} \, d\theta_{t-1}.
\]

Substituting this equivalence back in results in the desired recursive filter:

\[
Bel(X_t) = p(z_t|\theta_t)^{-1} p(i_t|\theta_t)^{-1} p(i_t|x_t, \theta_t) p(i_t|x_t, \theta_t) \int p(\theta_t|\theta_{t-1}, u_{t-1}, i_t) \int p(x_t|x_{t-1}, \theta_{t-1}, u_{t-1}) \\
Bel(X_{t-1}) \, dx_{t-1} \, d\theta_{t-1}.
\]

B. Simplification

As mentioned earlier (Equation 11), a simplification could be introduced if the assumption \(p(i_t|x_t, \theta_t) = p(i_t|\theta_t)\) is valid, resulting in \(p(i_t|\theta_t) p(i_t|\theta_t)^{-1} = 1\). This situation may arise if the sensors employed are able to purely measure orientation-relevant properties, while never being affected by where the measurements are taken. The experiments conducted for this work make that assumption. For convenience, define the normalizing constant

\[
\eta = \begin{cases} 
  p(z_t|\theta_t)^{-1} & \text{if } p(i_t|x_t, \theta_t) = p(i_t|\theta_t) \\
  p(z_t|\theta_t)^{-1} p(i_t|\theta_t)^{-1} & \text{otherwise}
\end{cases}
\]

The filter can then be rewritten as:

\[
Bel(X_t) = \eta \, p(z_t|x_t, \theta_t) \, [p(i_t|x_t, \theta_t)] \int p(\theta_t|\theta_{t-1}, u_{t-1}, i_t) \\
\int p(x_t|x_{t-1}, \theta_{t-1}, u_{t-1}) \\
Bel(X_{t-1}) \, dx_{t-1} \, d\theta_{t-1}.
\]

In this filter, the distribution \(p(\theta_t|\theta_{t-1}, u_{t-1}, i_t)\) is computed using an Extended Kalman filter based on quaternions [15]. Only a single Kalman filter is maintained for efficiency, although exploring the use of multiple filters is an obvious area for future work. A particle filter is used to compute \(p(x_t|x_{t-1}, \theta_{t-1}, u_{t-1})\), which is conditioned on the previous Kalman filtered orientation estimate. Particle \(p_i = (x_i, \theta_i)\) is composed of the sampled position \(x_i\) and a computed orientation sample based on the Kalman filter. Once an \textit{a-priori} pose estimate has been computed, it is weighted using \(p(z_t|x_t, \theta_t)\) and \(p(i_t|x_t, \theta_t)\). The former computes the likelihood of observing a 3D range scan \(z_t\). The data association process for \(p(z_t|x_t, \theta_t)\) uses an extended version of the NDT. See [20] for details and Figure 2 for a graphical sample based on a particular \(z_t\). Finally, \(\eta\) is used to approximate a normalizing constant, and as discussed, the distribution \(p(i_t|...)\) is ‘optional’ and ignored in this work. Certainly, exploring the use of this distribution is another area for potential future work.

IV. FILTER APPLICATION

The filter developed in the previous section has been applied to several datasets to evaluate its performance. The datasets used for validation vary from game models (for controlled testing) to real world datasets such as vision-based datasets from the AQUA and C2SM projects. Due to space constraints, here we compare the performance of the filter to a naïve 6DOF filter, and show the results of applying the algorithm to a dataset of a subway car [19] (see Figure 3); the results of further experiments are described in [20]. In order to provide a controlled evaluation of the approach, the algorithm is evaluated off-line on collected datasets, while motion through these datasets is simulated or based on egomotion estimates obtained when the data is collected.

A. Experiment Conditions

The environment map used for the experiments reported here was collected using vision based technologies (see [20] for details). Additional noise and error for range image \(z_t\) (composed of range points \(z^j_t\)) is simulated by adding random, uniformly distributed noise to the point clouds that make up the dataset. As an additional corruption, three randomly positioned occlusions are included in each range image, with each occlusion covering 10% of the width and height of the range scan. These occlusions are included to
simulate random failures for portions of the scan caused, for example, by dynamic obstacles such as fish. Further:

- The particle filter for each experiment is initialized randomly using a uniform distribution, with particles covering a volume larger than the model itself. This ensures that the robot’s position is included in the initial distribution.
- The motion model for the particle filter adds additive noise independently to each axis from a normal distribution with the variance set to 0.5 $|u_i|$. This results in tightly clustered particles expanding into a “bubble” around the estimated motion, i.e., $u_t \pm 0.5 |u_t|$. Such a liberal model covers potential errors in the ego motion and also allows particles to move out of local maxima.
- The sampling rate for the IMU (and the corresponding extended Kalman filter update) is 100hz; by comparison, the inertiaCube3 [12] has an update rate of 180hz. Due to the greater computational cost associated with updating the particle filter, it is updated at a reduced rate in these experiments, once per second. Since the algorithm is run off-line, this update process results in the Kalman filter being updated 100 times, followed by a particle filter update.
- The unit $t$ in the graphs refers to the number of seconds since the robot started moving.
- To allow the Kalman filter estimating the orientation of the vehicle to converge, it is assumed that the robot is stationary for 10 seconds prior to moving. These 10 seconds occur prior to time $t = 1$ in the figures.
- All graphs represent results starting at time $t = 1$, after integrating the control $u_0$, sensor measurements $z_1, t_1$ and resampling once.
- The particle cloud mean $\mu_t$ is computed according to

\[
\mu_t = \sum_i x_i w_i,
\]

where $x_i$ is the robot’s position and $w_i$ is the scalar weight associated with the $i$th particle.

B. Comparison with a naïve filter

Decoupling the orientation estimate from the position estimate allows pose estimation to be performed at a reasonable computational cost. The naïve comparison filter estimates all parameters of the state using only particles. As such, the space that needs to be searched by the particle filter is much larger, and naturally requires more particles. Due to the number of particles required by the naïve 6DOF filter, this experiment was conducted on a smaller dataset (a virtually created ‘cave’ map), and the algorithm was given unrestricted time per update.

Running the filter repeatedly and varying the size of the particle cloud reveals the number of particles required to converge consistently, as well as the minimum number of particles with which the filter starts to fail. When using the EKF enabled filter, the 6DOF estimation approach presented here consistently converges with 750 particles (Figure 4). Using 500 particles the filter becomes less stable, and exhibits some occasional failures (10% of experimental runs, where failure is defined as the position estimate being off by at least one meter from the correct estimate at time $t = 50$). On the other hand, disabling the EKF, and instead sampling all six parameters, results in a requirement for a much greater number of particles. When using 150,000 particles the failure rate is around 50%, with the other half correctly converging on the desired pose. Increasing the number of particles to 250,000 results in a 19% failure rate for the naïve filter.

V. SUMMARY

Pose estimation is an important capability for mobile robots; without it, motion planning becomes difficult or even impossible. Although pose estimation is an important task for robots able to move with six degrees of freedom, directly extending the existing work for 2D pose estimation into the 3D realm is not computationally attractive. The increased dimensionality of the problem requires a vastly larger number of particles that typically cannot be effectively computed using current hardware. Reducing the dimensionality of the problem is key to reducing the computational resources required. Some solutions attempt to do so by limiting the motion of the device, or by first computing a 2D pose estimate and then extending that estimate into 3D. Such approaches assume the the motion of the device can be limited, or that a reasonable 2D pose estimate can be computed.
A. Limitations

The focus of this work has been on vehicles with characteristics similar to the AQUA robot. Certain implicit assumptions that are made regarding the robot may not be valid when considering other platforms. For example, the work assumes a vehicle that moves slowly enough such that there is significant scene overlap between successive video frames. This allows for visual odometry (egomotion estimation) to be used for a priori estimation of the change in pose for the robot. Some alternative method must be employed to estimate the change if this work would be applied to estimating the pose of a faster moving vehicle.

The developed filter does not explicitly account for missing data in the models; such as the presence of people, windows, or “holes” in the map. People and objects that move around in the environment cause readings by the range sensor that will not match anything in the “known” environment. This may result in wasting resources on searching for objects that are not found the map. While, to a certain extent, the NDT approach is robust to such missing data, the evaluation of data that won’t match anything in the map is a waste of resources. Extension of the 2D novelty and entropy filters in [4] to 3D could improve some of these results.

REFERENCES