Hindawi Publishing Corporation
Journal of Sensors
Volume 2016, Article ID 3891865, 23 pages
http://dx.doi.org/10.1155/2016/3891865

Research Article

FastSLAM Using Compressed Occupancy Grids

Christopher Cain[1] and Alexander Leonessa[2]

[1]*SEW-EURODRIVE, Lyman, SC 29365, USA*
[2]*Center for Dynamic Systems Modeling and Control, Department of Mechanical Engineering, Virginia Tech, Blacksburg, VA 24061, USA*

Correspondence should be addressed to Alexander Leonessa; aleoness@vt.edu

Received 21 October 2015; Accepted 8 May 2016

Academic Editor: Maan E. El Najjar

Robotic vehicles working in unknown environments require the ability to determine their location while learning about obstacles located around them. In this paper a method of solving the SLAM problem that makes use of compressed occupancy grids is presented. The presented approach is an extension of the FastSLAM algorithm which stores a compressed form of the occupancy grid to reduce the amount of memory required to store the set of occupancy grids maintained by the particle filter. The performance of the algorithm is presented using experimental results obtained using a small inexpensive ground vehicle equipped with LiDAR, compass, and downward facing camera that provides the vehicle with visual odometry measurements. The presented results demonstrate that although with our approach the occupancy grid maintained by each particle uses only 40% of the data needed to store the uncompressed occupancy grid, we can still achieve almost identical results to the approach where each particle filter stores the full occupancy grid.

1. Introduction

One of the most important problems in the development of an unmanned vehicle (UMV) is giving it the ability to be placed in a completely unknown environment and allowing it to determine (1) its surroundings, based on what it "sees" using available sensors, and (2) its position, based on how far it has moved and what it sees. In the field of robotics this problem is referred to as Simultaneous Localization and Mapping (SLAM) or Concurrent Mapping and Localization (CML). In this paper we develop a method of solving the SLAM problem which implements a representation of the world using a compressed occupancy grid [1].

The approach we develop is an extension of the FastSLAM algorithm [2] which is one of the first solutions to the SLAM problem that makes use of a particle filter. More specifically the FastSLAM algorithm uses a Rao-Blackwellized [3, 4] particle filter to generate a position estimate of the UMV while each particle in filter maintains its copy of the occupancy grid. Based on the expected size of the environment in which a UMV operates and the level of detail required, the size of each occupancy grid can become extremely large (matrices holding hundreds of thousands of entries). Furthermore, as discussed further in Section 3.1.2, the probability that the estimate produced by the FastSLAM algorithm is accurate increases as the number of particles in the particle filter increases. For these reasons the amount of computer resources required to perform the algorithm can grow quickly. This problem is exacerbated in small form factor and low cost UMVs as the computing resources available to vehicles of this type are much less than the resources available to larger and more expensive vehicles. The approach that we develop extends the FastSLAM algorithm [2] to use compressed occupancy grids to overcome the memory restrictions that can occur when storing full occupancy grids without making drastic changes to the original algorithm.

The remainder of this paper is organized as follows. The mathematical notations used in this paper along with some mathematical preliminaries are presented in Section 2. The updated FastSLAM algorithm that makes use of compressed occupancy grids is introduced in Section 3. In Section 4 four separate approaches for reconstructing the compressed occupancy grid are examined. Section 5 presents a set of performance improvements to some components of the

algorithm. In Section 6 experimental results are provided and concluding remarks are presented in Section 7.

2. Mathematical Preliminaries

In much of the existing literature, the SLAM problem is addressed in a probabilistic sense. In many cases we would like to estimate the probability, also referred to as the distribution, of some random variable $x \in \mathbb{R}$ and we denote the distribution of x as $p(x)$. In many cases we use some additional information $y \in \mathbb{R}$ to tell us something about the random variable x. In this situation $p(x)$ is referred to as the *prior probability* and it is all that we know about the probability of x without the inclusion of y, in many cases this is shortened to *prior*. The distribution of x with the inclusion of the data y is denoted as $p(x \mid y)$ and referred to as the *posterior probability* and in many cases is just referred to as the *posterior*.

In SLAM we are attempting to estimate the pose $\boldsymbol{\xi}_k$ of a UMV and a map $\mathbf{M}_k$ of the environment that surrounds the UMV at some time step $k > 0$. In this paper we treat the world in which the UMV operates as a two-dimensional plane; thus $\boldsymbol{\xi}_k \triangleq [x_k \;\; y_k \;\; \theta_k]^{\mathrm{T}}$, where $[x_k \;\; y_k] \in \mathbb{R}^2$ are the horizontal and vertical positions of the vehicle in some frame of reference and $\theta_k \in (-\pi, \pi]$ is the heading of the UMV with respect to the positive horizontal axis in the frame of reference. From our use of occupancy grids, the estimated map is represented by a matrix $\mathbf{M}_k \in \mathbb{R}^{r \times c}$, where $r \in \mathbb{N}$ are the number of rows in the grid and $c \in \mathbb{N}$ are the number of columns. We assume that the environment that the UMV operates is static; therefore during the time in which the UMV is performing the SLAM algorithm the environment does not change. Based on this assumption, to simplify the notation, the occupancy grid is denoted with $\mathbf{M}$. The estimate produced by SLAM, in many cases, makes use of sensor measurements and control inputs. The set of sensor measurements $\mathbf{z}_{1:k}$ denotes the full set of sensor measurements for $k > 0$; thus $\mathbf{z}_{1:k} \triangleq [\mathbf{z}_1, \ldots, \mathbf{z}_k]$, where $\mathbf{z}_i \in \mathbb{R}^m$, $i = 1, 2, \ldots, k$, and $m \in \mathbb{N}$ is the number of measurements in $\mathbf{z}_i$. In the same way the full set of control inputs are defined as $\mathbf{u}_{1:k} \triangleq [\mathbf{u}_1, \ldots, \mathbf{u}_k]$, where $\mathbf{u}_i \in \mathbb{R}^c$, $i = 1, 2, \ldots, k$, and $c \in \mathbb{N}$ is the size of the control vector $\mathbf{u}_i$. With the notation defined and the required mathematical preliminaries provided, the FastSLAM algorithm that makes use of compressed occupancy grids is presented in the following section.

3. FastSLAM with Compressed Occupancy Grids

In general the goal of SLAM is to estimate the pose $\boldsymbol{\xi}_k$ of a UMV and the map, $\mathbf{M}$, of the unknown environment in which the UMV is operating at some time step $k > 0$. We present an extension of the FastSLAM algorithm that uses compressed occupancy grids to store the map. Before presenting our approach, an overview of the FastSLAM algorithm that uses standard occupancy grids (FastSLAM OG) is provided.

3.1. FastSLAM with Occupancy Grids. As opposed to many SLAM solutions that attempt to solve the online SLAM problem that is estimating the posterior that represents the pose of a UMV and the map of the environment based on a set of control inputs and sensor measurements, the FastSLAM approach attempts to estimate the full SLAM posterior. The full SLAM posterior is given as

$$p\left(\boldsymbol{\xi}_{1:k}, \mathbf{M} \mid \mathbf{u}_{1:k}, \mathbf{z}_{1:k}\right), \tag{1}$$

which is the distribution that represents the trajectory of UMV $\boldsymbol{\xi}_{1:k}$ and the map of the environment $\mathbf{M}$ based on the set of control inputs $\mathbf{u}_{1:k}$ and observations $\mathbf{z}_{1:k}$. By estimating the full SLAM posterior, (1) can be factored, using the property of conditional independence, into a pair of simpler terms which is given by

$$\begin{aligned} p\left(\boldsymbol{\xi}_{1:k}, \mathbf{M} \mid \mathbf{u}_{1:k}, \mathbf{z}_{1:k}\right) &= p\left(\boldsymbol{\xi}_{1:k} \mid \mathbf{u}_{1:k}, \mathbf{z}_{1:k}\right) \\ &\quad \times p\left(\mathbf{M} \mid \boldsymbol{\xi}_{1:k}, \mathbf{u}_{1:k}, \mathbf{z}_{1:k}\right), \end{aligned} \tag{2}$$

where $p(\boldsymbol{\xi}_{1:k} \mid \mathbf{u}_{1:k}, \mathbf{z}_{1:k})$ is the distribution that represents the UMV trajectory and $p(\mathbf{M} \mid \boldsymbol{\xi}_{1:k}, \mathbf{u}_{1:k}, \mathbf{z}_{1:k})$ is the distribution that represents the map of the environment. The FastSLAM approach estimates the factored SLAM posterior (2) using a Rao-Blackwellized particle filter. The Rao-Blackwellized particle filter is a version of a sampling importance resampling (SIR) [5] particle filter and it is the SIR particle filter that provides the format of the FastSLAM solution. The Rao-Blackwellized particle filter estimates the distribution representing a portion of the state, which in FastSLAM is the UMV trajectory $\boldsymbol{\xi}_{1:k}$, using a particle filter, and each particle in the particle filter maintains its own estimate of the remainder of the state, which is the map $\mathbf{M}$. As a result, each particle in the particle filter is composed of a UMV pose and a map, which the FastSLAM OG algorithm represents using an occupancy grid; thus the pth particle at time k is defined as

$$\mathcal{X}_k^{[p]} \triangleq \left[\boldsymbol{\xi}_k^{[p]} \;\; \mathbf{M}_k^{[p]}\right], \tag{3}$$

where $\boldsymbol{\xi}_k^{[p]}$ is the pose estimate of the pth particle at time k, $\mathbf{M}_k^{[p]}$ is the occupancy grid of the environment maintained by the particle as it has been generated through time step k, and $p \in [1, n]$ is the index corresponding to the pth particle where n is the total number of particles used in the particle filter. The FastSLAM approach is implemented in a four-step procedure based on the use of a SIR particle filter to estimate (1) and an overview of the algorithm is presented in Algorithm 1 for time k.

3.1.1. Motion Model Update. In the first step of the algorithm (Line (4) in Algorithm 1) a new particle pose is generated and added to a set of temporary particles, $\overline{\mathcal{X}}_k$. The new pose for the pth particle is generated by sampling from the probabilistic motion model of the UMV:

$$\boldsymbol{\xi}_k^{[p]} \sim p\left(\boldsymbol{\xi}_k \mid \boldsymbol{\xi}_{k-1}^{[p]}, \mathbf{u}_k\right). \tag{4}$$

(1) **procedure** FastSLAMOG($\mathcal{X}_{k-1}, \mathbf{u}_k, \mathbf{z}_k$)
(2) $\overline{\mathcal{X}}_k \leftarrow \{\}$
(3) **for** $p \leftarrow 1, n$ **do**
(4) $\boldsymbol{\xi}_k^{[p]} \sim p(\boldsymbol{\xi}_k \mid \boldsymbol{\xi}_{k-1}^{[p]}, \mathbf{u}_k)$
(5) $w^{[p]} \leftarrow \eta p(\mathbf{z}_k \mid \boldsymbol{\xi}_k^{[p]}, \mathbf{M}_{k-1}^{[p]})$
(6) $\mathbf{M}_k^{[p]} = \text{OGUpdate}(\boldsymbol{\xi}_k^{[p]}, \mathbf{M}_{k-1}^{[p]}, \mathbf{z}_k)$
(7) $\overline{\mathcal{X}}_k^{[p]} \leftarrow \left[\boldsymbol{\xi}_k^{[p]} \quad \mathbf{M}_k^{[p]} \quad w^{[p]}\right]$
(8) **end for**
(9) $\mathcal{X}_k \leftarrow \{\}$
(10) **for** $p = 1, n$ **do**
(11) select j with probability proportional to $w^{[p]}$
(12) $\mathcal{X}_k^{[p]} \leftarrow \overline{\mathcal{X}}_k^{[j]}$
(13) **end for**
(14) **end procedure**

Algorithm 1: FastSLAM OG algorithm.

The probabilistic motion model of the UMV generates a new pose based on the state transition model of the UMV, the pose of the pth particle at time step $k-1$, the current control input $\mathbf{u}_k$, and the characteristics of the noise on $\mathbf{u}_k$. The result is that the particles in $\overline{\mathcal{X}}_k$ are distributed according to

$$p\left(\boldsymbol{\xi}_{1:k} \mid \mathbf{u}_{1:k}, \mathbf{z}_{1:k-1}\right), \tag{5}$$

which is referred to as the *proposal distribution*. The actual distribution which we are attempting to estimate is referred to as the *target distribution* and given as

$$p\left(\boldsymbol{\xi}_{1:k} \mid \mathbf{u}_{1:k}, \mathbf{z}_{1:k}\right). \tag{6}$$

The proposal distribution is transformed to the target distribution in the last stage of the algorithm, the resampling stage (Lines (10)–(13) in in Algorithm 1). Before the resampling process can occur, $\mathbf{z}_k$ must be incorporated into the estimate, which is achieved in the calculation of the particle's importance weight.

3.1.2. Importance Weight Calculation. In the second step of the algorithm an importance weight for each particle is generated. The need for an importance weight for each particle comes from the use of a SIR particle filter and, as seen in the previous section, we are sampling particles from a distribution different from the one that we are attempting to estimate. According to [6], if we are unable to directly sample from the distribution that we are attempting to estimate, then the importance weight for each particle is given as

$$w_k^{[p]} = \frac{\text{target distribution}}{\text{proposal distribution}}, \tag{7}$$

and particles are drawn from $\overline{\mathcal{X}}_k$ with replacement and added to $\mathcal{X}_k$ with a probability proportional to $w_k^{[p]}$. When drawing particles from $\overline{\mathcal{X}}_k$ with replacement, each particle that is added to $\mathcal{X}_k$ remains in $\overline{\mathcal{X}}_k$ so that a given particle that is selected has the potential of being added to $\mathcal{X}_k$ multiple times. After the resampling process $\mathcal{X}_k$ will approximate the target distribution and the quality of the approximation will improve, the number of particles increases.

Using (7) along with the previously discussed distributions (5) and (6), the importance weight of each particle simplifies to [7]

$$w_k^{[p]} = \frac{p\left(\boldsymbol{\xi}_{1:k} \mid \mathbf{u}_{1:k}, \mathbf{z}_{1:k}\right)}{p\left(\boldsymbol{\xi}_{1:k} \mid \mathbf{u}_{1:k}, \mathbf{z}_{1:k-1}\right)} = \eta p\left(\mathbf{z}_k \mid \boldsymbol{\xi}_k^{[p]}, \mathbf{M}_{k-1}^{[p]}\right), \tag{8}$$

where $p(\mathbf{z}_k \mid \boldsymbol{\xi}_k^{[p]}, \mathbf{M}_{k-1}^{[p]})$ is the probabilistic measurement model of the sensor being used by the UMV and $\eta > 0$ is a normalizing factor. The probabilistic measurement model is dependent on the perceptual sensor being used by the UMV and it calculates the probability that the sensor measurement would make based on the current pose estimate and map maintained by the particle. This form of importance weight incorporates the sensor observations into the estimate which were not included in the pose sampling procedure. Once the importance weight has been calculated for each particle then the occupancy grid maintained by each particle is updated using the new particle pose and current sensor measurement.

3.1.3. Occupancy Grid Update. Once the importance weight for each particle has been calculated, the third step of the algorithm updates the occupancy grid of each particle using the current sensor measurement and pose estimate. Occupancy grids decompose an unknown environment into a finite set of cells where each cell holds the probability that the location in the world enclosed by the cell is occupied by an object. From [8], the occupancy grid for a given particle at time k is obtained in a probabilistic sense by calculating the posterior over maps based on the history of sensor measurements and trajectory of a particle:

$$p\left(\mathbf{M}_k^{[p]} \mid \boldsymbol{\xi}_{1:k}^{[p]}, \mathbf{z}_{1:k}\right). \tag{9}$$

By decomposing the environment into cells, the posterior calculated by the occupancy grid can be restated as a product of cellular posteriors:

$$p\left(\mathbf{M}_k^{[p]} \mid \boldsymbol{\xi}_{1:k}^{[p]}, \mathbf{z}_{1:k}\right) = \prod_{i,j} p\left(\mathbf{M}_k^{[p]}(i,j) \mid \boldsymbol{\xi}_{1:k}^{[p]}, \mathbf{z}_{1:k}\right), \tag{10}$$

where $p(\mathbf{M}_k^{[p]}(i,j) \mid \boldsymbol{\xi}_{1:k}^{[p]}, \mathbf{z}_{1:k})$ contains the probability that the location in the environment represented by the cell at (i,j) is occupied based on the measurement history and particle trajectory. A solution to a problem of this form is the binary Bayes filter [8]. The binary Bayes filter stores the probability of occupancy for each cell in log odds form to prevent truncation errors that can occur during the update process. The log odds form of the variable $x \in \mathbb{R}$ is given as

$$l(x) \triangleq \log \frac{x}{1-x}, \tag{11}$$

and the value of x can be recovered from the log odds form as

$$x \triangleq 1 - \frac{1}{1 + \exp(l(x))}. \tag{12}$$

(1) **procedure** OGUpdate($\boldsymbol{\xi}_k, \mathbf{M}_{k-1}, \mathbf{z}_k$)
(2) **for** $i = 1, h$ **do**
(3) **for** $j = 1, w$ **do**
(4) **for** $l = 1, m$ **do**
(5) **if** $\mathbf{M}_{k-1}(i, j)$ lies in the perceptual range of $\mathbf{z}_k(l)$ **then**
(6) $$\mathbf{M}_k(i, j) \longleftarrow \mathbf{M}_{k-1}(i, j) + \log \frac{p(\mathbf{M}(i, j) \mid \boldsymbol{\xi}_k, \mathbf{z}_k(l))}{1 - p(\mathbf{M}(i, j) \mid \boldsymbol{\xi}_k, \mathbf{z}_k(l))} - l_0$$
(7) **end if**
(8) **end for**
(9) **end for**
(10) **end for**
(11) **end procedure**

Algorithm 2: Standard occupancy grid update algorithm.

The binary Bayes filter based occupancy grid update modifies the probability of occupancy of each cell in the grid when an observation provides information about that location. As a consequence, for locations that have been observed, the probability of occupancy quickly approaches 0 for free cells and 1 for occupied cells. While the addition of new information to the estimate makes the probability quickly approach the limits, the probability of occupancy of each cell should never equal 0 or 1 as this would assume perfect knowledge about the occupancy of the cell. Depending on the data type selected to store the probability, the precision required to store the value can quickly be exceeded resulting in the probability being rounded to either 0 or 1. Once this rounding has occurred any new information learned about the occupancy of a cell is lost. To overcome this problem the binary Bayes filter stores the probability of occupancy in log odds form which takes the potential probability values in $[0, 1]$ and spreads them over the range $(-\infty, \infty)$. By spreading the small range of probability values over this much larger range, the data type being used to store the probability has more precision to store the probability thus preventing the truncation/rounding from occurring. The occupancy grid update algorithm based on the binary Bayes filter is provided in Algorithm 2.

In Algorithm 2, $p(\mathbf{M}(i, j) \mid \boldsymbol{\xi}_k, \mathbf{z}_k(l))$ is the inverse sensor model of the sensor being used to generate the map and it returns the probability that the cell at (i, j) is occupied based on the lth sensor measurement and the pose estimate of the particle. The value of l_0 in Algorithm 2 is referred to as the prior of occupancy and is defined as

$$l_0 \triangleq \log \frac{p(\mathbf{M}(i, j))}{1 - p(\mathbf{M}(i, j))} \tag{13}$$

and gives the probability that a cell is occupied before any sensor measurements are integrated into the map. Typically $p(\mathbf{M}(i, j))$ in (13) is selected to be 0.5.

3.1.4. Resampling. After the occupancy grid for each particle in $\overline{\mathcal{X}}_k$ has been updated, the final step of the algorithm is the resampling process. During the resampling process particles are drawn with replacement from $\overline{\mathcal{X}}_k$ with a probability proportional to $w^{[p]}$. This resampling step takes the set of temporary particles that are distributed according to (5) and ensures that there is a high probability that $\mathcal{X}_k$ is distributed according to (6).

One of the problems with this approach is the large amount of memory required to run the algorithm. The estimate generated by the algorithm improves as the number of particles increases; thus we like to have as many particles as possible. However, each particle must keep a copy of the occupancy grid and each occupancy grid can be quite large. It is easy to see that while we would like to run the particle filter with as many particles as possible we can be limited by the fact that each particle must maintain a full copy of the occupancy grid. To address this problem we propose an updated version of the algorithm that decreases the amount of memory required for each particle by storing a compressed version of the occupancy grid as opposed to the full uncompressed occupancy grid.

3.2. FastSLAM with Compressed Occupancy Grids. Before discussing the updated SLAM algorithm, we will first address the compressed form of the occupancy grid. To compress $\mathbf{M}_k^{[p]}$ we first reshape the occupancy grid so that $r \times c$ matrix becomes a column vector containing $n = rc$ elements and we refer to the vector form of the occupancy grid as $\mathbf{m}_k^{[p]} \triangleq \text{vec}(\mathbf{M}_k^{[p]})$. The reshaping process is performed in such a way that the cell in the original occupancy grid located at (i, j) can be accessed in the vector form at l, where $l = ic + j$. In order to compress the occupancy grid without major degradation, we assume that the occupancy grid can be represented in an alternate basis in which it is sparse. The relationship between the two different representations is given as

$$\mathbf{m}_k^{[p]} = \boldsymbol{\Phi} \mathbf{c}_k^{[p]}, \tag{14}$$

where $\boldsymbol{\Phi}$ is a transformation matrix that defines the relationship and $\mathbf{c}_k^{[p]}$ is the set of coefficients that represent the occupancy grid in the alternate basis. In order for the occupancy grid to be compressible, $\mathbf{c}_k^{[p]}$ must be sparse [9]; that is $\#\{\text{supp}(\mathbf{c}_k^{[p]})\} < n$, where $\text{supp}(\cdot)$ is the support operator and # denotes the cardinality of the set. This means

(1) **procedure** FastSLAMCOG($\mathcal{X}_{k-1}, \mathbf{u}_k, \mathbf{z}_k$)
(2) $\quad\overline{\mathcal{X}}_k \leftarrow \{\}$
(3) $\quad$**for** $p \leftarrow 1, n$ **do**
(4) $\quad\quad\boldsymbol{\xi}_k^{[p]} \sim p(\boldsymbol{\xi}_k \mid \boldsymbol{\xi}_{k-1}^{[p]}, \mathbf{u}_k)$
(5) $\quad\quad\widehat{\mathbf{c}}_k^{[p]} = \text{UNCOMPRESS}(\mathbf{y}_{k-1}^{[p]})$
(6) $\quad\quad w^{[p]} \leftarrow \eta p(\mathbf{z}_k \mid \boldsymbol{\xi}_k^{[p]}, \boldsymbol{\Phi}\widehat{\mathbf{c}}_k^{[p]})$
(7) $\quad\quad\boldsymbol{\alpha}_k^{[p]} = \text{AlphaConstruction}(\mathbf{m}_{k-1}^{[p]}, \mathbf{z}_k)$
(8) $\quad\quad\mathbf{y}_k^{[p]} = \mathbf{H}[\widehat{\mathbf{c}}_k^{[p]} + \boldsymbol{\Phi}^{\mathrm{T}}[\boldsymbol{\alpha}_k^{[p]} - \boldsymbol{\beta}]]$
(9) $\quad\quad\overline{\mathcal{X}}_k^{[p]} \leftarrow \left[\boldsymbol{\xi}_k^{[p]} \quad \mathbf{y}_k^{[p]} \quad w^{[p]}\right]$
(10) $\quad$**end for**
(11) $\quad\mathcal{X}_k \leftarrow \{\}$
(12) $\quad$**for** $j = 1, n$ **do**
(13) $\quad\quad$sample with replacement from select i with probability proportional to $w^{[p]}$
(14) $\quad\quad\mathcal{X}_k^{[p]} \leftarrow \overline{\mathcal{X}}_k^{[p]}$
(15) $\quad$**end for**
(16) **end procedure**

Algorithm 3: FastSLAM algorithm with compressed occupancy grids.

that, in order for us to be able to compress the occupancy grid, we must find a basis in which the coefficient vector is composed of a large number of zeros. If this condition is met then the occupancy grid for a given particle can be compressed using

$$\mathbf{y}_k^{[p]} = \mathbf{H}\mathbf{c}_k^{[p]}, \tag{15}$$

where $\mathbf{H}$ is a matrix that performs the compression by selecting a subset of coefficient data that is smaller than the amount of data in the complete signal.

The FastSLAM COG algorithm is a modification of Algorithm 1. Since we are now maintaining the compressed form of the occupancy grid, ith particle in the particle set is defined as

$$\mathcal{X}_k^{[p]} \triangleq \left[\boldsymbol{\xi}_k^{[p]} \quad \mathbf{y}_k^{[p]}\right]. \tag{16}$$

As with FastSLAM OG, FastSLAM COG is performed in several steps and an overview of the updated algorithm is presented in Algorithm 3. As a result of using compressed occupancy grids the only changes that must be addressed are the steps of the algorithm that require the occupancy grid, the importance weight calculation, and occupancy grid update.

3.2.1. Importance Weight Calculation. As discussed in Section 3.1.2, the importance weight for each particle is the normalized probabilistic measurement model for each particle. The probabilistic measurement model is a measure of how well the current set of sensor measurements matches up with what we expect based on the current pose and map of the particle. In calculating the probabilistic measurement model, the algorithm must have access to the raw occupancy grid values so that an expected sensor measurement can be generated. Based on this requirement, the compressed occupancy grid coefficients are first uncompressed. For now a general reconstruction operator denoted as $\mathbf{H}_r$ is used; however the details of several reconstruction methods are explored in detail in Section 4. Once the compressed coefficients have been reconstructed, the occupancy grid can be extracted using (14). The importance weight calculation can be found in Algorithm 3 on Line (6).

3.2.2. Compressed Occupancy Grid Update. As previously seen in Algorithm 2, the standard approach to updating the occupancy grid is to examine every cell in the occupancy grid and if a given cell falls within the perceptual range of the sensor attached to the UMV then the corresponding cell of the occupancy grid is updated according to

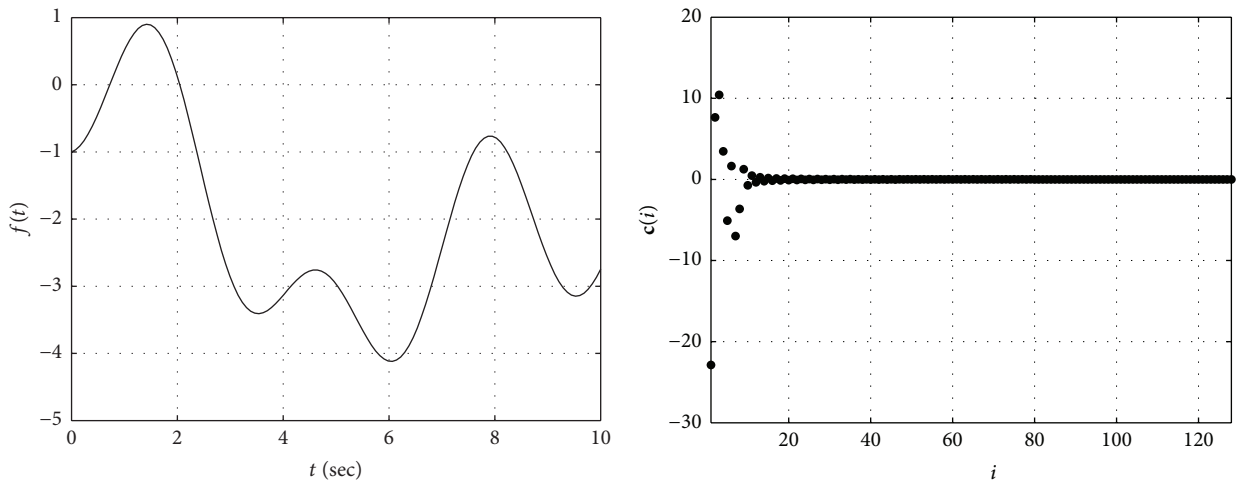
$$\mathbf{M}_k^{[p]}(i,j) = \mathbf{M}_{k-1}^{[p]}(i,j) + \log\frac{p\left(\mathbf{M}(i,j) \mid \boldsymbol{\xi}_k^{[p]}, \mathbf{z}_k(l)\right)}{1 - p\left(\mathbf{M}(i,j) \mid \boldsymbol{\xi}_k^{[p]}, \mathbf{z}_k(l)\right)} - l_0. \tag{17}$$

Using Algorithm 2 as our basis, we rewrite the occupancy grid update in vector form. If we assume that the full occupancy grid is stored in vector form, then the occupancy grid update can be performed using

$$\mathbf{m}_k^{[p]} = \mathbf{m}_{k-1}^{[p]} + \boldsymbol{\alpha}_k^{[p]} - \boldsymbol{\beta}, \tag{18}$$

where $\boldsymbol{\alpha}_k^{[p]}$ is constructed using Algorithm 4 and $\boldsymbol{\beta}$ is vector with every element containing l_0.

Algorithm 4 first initializes $\boldsymbol{\alpha}_k^{[p]}$ using l_0. This ensures that cells that do not fall within the perceptual range of the sensor remain unchanged by the algorithm. For cells that fall into the perceptual range of the sensor, their $\boldsymbol{\alpha}$ value is set as the log odds form of the inverse sensor model and it is this value that either increases or decreases the likelihood that the cell is occupied. Equation (18) updates the occupancy grid in vector form; however we need to modify it so that it updates the occupancy grid which is stored using an alternative basis.



(a) (b)

Figure 1: (a) The test function given by (24) and (b) the test function represented in the discrete cosine basis.

(1) **procedure** AlphaConstruction($\mathbf{m}, \mathbf{z}_k$)
(2) **for** $i = 1, n$ **do**
(3) $\boldsymbol{\alpha}(i) = l_0$
(4) **end for**
(5) **for** $i = 1, n$ **do**
(6) **for** $j = 1, m$ **do**
(7) **if** $\mathbf{m}(i)$ in the perceptual range of $\mathbf{z}_k(j)$ **then**
(8) $$\boldsymbol{\alpha}(i) = \log \frac{\left(\mathbf{m}(i) \mid \boldsymbol{\xi}_{1:k}^{[p]}, \mathbf{z}_k(j)\right)}{1 - p\left(\mathbf{m}(i) \mid \boldsymbol{\xi}_{1:k}^{[p]}, \mathbf{z}_k(j)\right)}$$
(9) **end if**
(10) **end for**
(11) **end for**
(12) **end procedure**

Algorithm 4: Construction of $\boldsymbol{\alpha}^{[p]}$.

Incorporating (14), the vector form of the occupancy grid update in the alternate basis becomes

$$\mathbf{c}_k^{[p]} = \mathbf{c}_{k-1}^{[p]} + \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\alpha}_k^{[p]} - \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\beta}, \tag{19}$$

where $\boldsymbol{\Phi}^{\mathrm{T}}$ is used to convert $\boldsymbol{\alpha}_k^{[p]}$ and $\boldsymbol{\beta}$ to the alternate basis. In (19), $\boldsymbol{\Phi}^{\mathrm{T}}$ is used to change basis as opposed to $\boldsymbol{\Phi}^{-1}$ because we assume that our choice of basis is orthonormal so $\boldsymbol{\Phi}^{-1} = \boldsymbol{\Phi}^{\mathrm{T}}$; however if the alternative basis that is selected is not orthonormal then $\boldsymbol{\Phi}^{-1}$ should replace $\boldsymbol{\Phi}^{\mathrm{T}}$ in (19). We can now simplify (19) by grouping the two components that are multiplied by $\boldsymbol{\Phi}^{\mathrm{T}}$ to reduce the number of matrix-vector multiplications; this yields

$$\mathbf{c}_k^{[p]} = \mathbf{c}_{k-1}^{[p]} + \boldsymbol{\Phi}^{\mathrm{T}} \left[\boldsymbol{\alpha}_k^{[p]} - \boldsymbol{\beta}\right]. \tag{20}$$

Equation (20) updates the occupancy grid in the alternate basis; hence the final step is to incorporate compression into the update. The occupancy grid maintained by each particle contains the compressed set of basis of coefficients; thus (20) can be updated using (15) to yield

$$\mathbf{c}_k^{[p]} = \mathbf{H}_r \mathbf{y}_{k-1}^{[p]} + \boldsymbol{\Phi}^{\mathrm{T}} \left[\boldsymbol{\alpha}_k^{[p]} - \boldsymbol{\beta}\right]. \tag{21}$$

Finally, the basis coefficients used to represent the occupancy grid must be recompressed in order to be stored by the particle. Including the final compression the compressed occupancy grid update can be performed according to

$$\mathbf{y}_k^{[p]} = \mathbf{H}\left[\mathbf{H}_r \mathbf{y}_{k-1}^{[p]} + \boldsymbol{\Phi}^{\mathrm{T}} \left[\boldsymbol{\alpha}_k^{[p]} - \boldsymbol{\beta}\right]\right], \tag{22}$$

which is seen in Algorithm 3 Line (8).

4. Compressed Signal Reconstruction Methods

As previously described we would like to store a compressed version of the occupancy grid as opposed to the complete occupancy grid to decrease the memory requirements of the algorithm. In order to make use of the compressed occupancy grid, a method for reconstructing the full occupancy grid from the compressed version must be selected. In this section several reconstruction methods are examined with the goal of selecting an approach to be used in FastSLAM COG. In FastSLAM COG we are attempting to compress the vector form of an occupancy grid; however in this section we will examine methods for reconstructing a generic discrete signal $\mathbf{x} \in \mathbb{R}^n$ that has been compressed.

We represent $\mathbf{x}$ using some alternate basis in which it is sparse and the relationship between $\mathbf{x}$ and its sparse representation is given by

$$\mathbf{x} = \boldsymbol{\Phi}\mathbf{c}, \tag{23}$$

where $\mathbf{c} \in \mathbb{R}^n$ is a vector of coefficients that represent $\mathbf{x}$ in the alternate basis and $\boldsymbol{\Phi} \in \mathbb{R}^{n \times n}$ is a matrix of basis functions that determines the relationship between $\mathbf{x}$ and $\mathbf{c}$. The discrete signal used in this section for testing is displayed in Figure 1(a) and the set of coefficients used to represent

the signal in the discrete cosine basis [10] can be seen in Figure 1(b). The test signal is generated by evaluating

$$f(t) = \sin(t) - \cos(2t) - 3\sin(0.25t), \tag{24}$$

at 128 evenly spaced points in [0, 10].

If $\mathbf{c}$ is sparse then $\mathbf{x}$ can be compressed; this sparseness can be seen by examining Figure 1(b) and noticing that a large number of the coefficients are zero. The compression is performed by selecting a subset of the coefficient information using

$$\mathbf{y} = \mathbf{H}\mathbf{c}, \tag{25}$$

where $\mathbf{H} \in \mathbb{R}^{m\times n}$ is the compression matrix and $m \in \mathbb{N}$ is the size of the compressed signal with $m < n$.

4.1. l_2 Reconstruction (l_2). One of the most simple methods for reconstructing a compressed signal is to reconstruct the signal while minimizing the error between the original and reconstructed signals as measured by l_2 norm, where l_2 norm of a signal $\mathbf{x} \in \mathbb{R}^n$ is defined as

$$\|\mathbf{x}\|_2 \triangleq \sqrt{\sum_{i=1}^{n} \mathbf{x}(i)^2}. \tag{26}$$

In examining (25), if the size of the compressed signal is the same as the original then the reconstruction can be performed according to

$$\mathbf{c} = \mathbf{H}^{-1}\mathbf{y}. \tag{27}$$

In reality the system is underdetermined due to $m < n$. A common solution to problems of this form uses the Moore-Penrose pseudoinverse of $\mathbf{H}$.

The Moore-Penrose pseudoeinverse [11, 12] is used to find the solution that minimizes l_2 error for systems of the form

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{28}$$

where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$, and $\mathbf{A} \in \mathbb{R}^{m\times n}$. The solution to the system found using the pseudoinverse is defined as

$$\mathbf{z} = \mathbf{A}^{\dagger}\mathbf{b}, \tag{29}$$

where $\mathbf{A}^{\dagger}$ is the pseudoinverse of $\mathbf{A}$ and it minimizes l_2 error so that

$$\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2 \geq \|\mathbf{A}\mathbf{z} - \mathbf{b}\|_2, \tag{30}$$

for all $\mathbf{x} \in \mathbb{R}^n$ if $\mathbf{x}$ is defined as

$$\mathbf{x} = \mathbf{A}^{\dagger}\mathbf{b} + \left(\mathbf{I} - \mathbf{A}^{\dagger}\mathbf{A}\right)\mathbf{w}, \tag{31}$$

for any vector $\mathbf{w} \in \mathbb{R}^n$.

Using the pseudoinverse of $\mathbf{H}$ a set of reconstructed coefficients is generated according to

$$\widehat{\mathbf{c}} = \mathbf{H}^{\dagger}\mathbf{y}, \tag{32}$$

and the reconstructed value of the signal is generated from

$$\widehat{\mathbf{x}} = \mathbf{\Phi}\widehat{\mathbf{c}}. \tag{33}$$

The results of l_2 reconstruction using four separate amounts of signal information, 25%, 50%, 75%, and 95%, are presented in Figure 2. The form selected for $\mathbf{H}$ needed to obtain such level of compression is an orthonormal Gaussian random matrix as it possesses the RIP property whose importance will be discussed in the following section.

4.2. Compressed Sensing Reconstruction (CS). The second reconstruction algorithm selected is based on an approach referred to in the literature as compressed sensing [13]. Compressed sensing allows for sparse signals to be reconstructed using fewer measurements that would be required according to the Nyquist/Shannon sampling principle. In compressed sensing the full set of sparse coefficients can be reconstructed from the compressed set by solving

$$\begin{aligned} \min \quad & \|\widehat{\mathbf{c}}\|_0 \\ \text{subject to} \quad & \left\|\mathbf{y} - \mathbf{H}\widehat{\mathbf{c}}\right\|_2^2 \leq \varepsilon, \end{aligned} \tag{34}$$

for some sufficiently small ε. In (34) $\|\widehat{\mathbf{c}}\|_0 \triangleq \#\{\text{supp}(\widehat{\mathbf{c}})\}$ is a count of the number of nonzero elements in the vector $\widehat{\mathbf{c}}$. Solving (34) is NP-hard and the solution cannot be found efficiently. A key finding of compressed sensing is that if the measurement matrix, $\mathbf{H}$, is constructed in such a way that it obeys the Restricted Isometry Property (RIP), which is defined as

$$(1-\delta)\|\mathbf{c}\|_2^2 \leq \|\mathbf{H}\mathbf{c}\|_2^2 \leq (1+\delta)\|\mathbf{c}\|_2^2, \tag{35}$$

form some small $\delta \geq 0$, then $\|\cdot\|_0$ in (34) can be replaced by $\|\cdot\|_1$ resulting in

$$\begin{aligned} \min \quad & \|\widehat{\mathbf{c}}\|_1 \\ \text{subject to} \quad & \left\|\mathbf{y} - \mathbf{H}\widehat{\mathbf{c}}\right\|_2^2 \leq \varepsilon. \end{aligned} \tag{36}$$

As opposed to being NP-hard to solve, (36) is a constrained convex optimization problem and can be solved efficiently using one of several available software packages with Lasso [14], NESTA [15], and l1 magic [16] being some examples. As in l_2 case, once the complete set of coefficients have been estimated using the compressed set, an estimate of the original signal is generated according to (33).

In order to examine the performance of the compressed sensing reconstruction approach, the test signal is reconstructed using the same four amounts of data, 25%, 50%, 75%, and 95%, with the same orthonormal Gaussian random matrix being used to perform the compression. Equation (36) is solved using the NESTA software library and the results of the reconstruction are shown in Figure 3.

4.3. Compressed Sensing Embedded Kalman Filtering (CSEKF). In the previous section a reconstruction approach was introduced that allows for the reconstruction of compressed signals using less data than expected based on

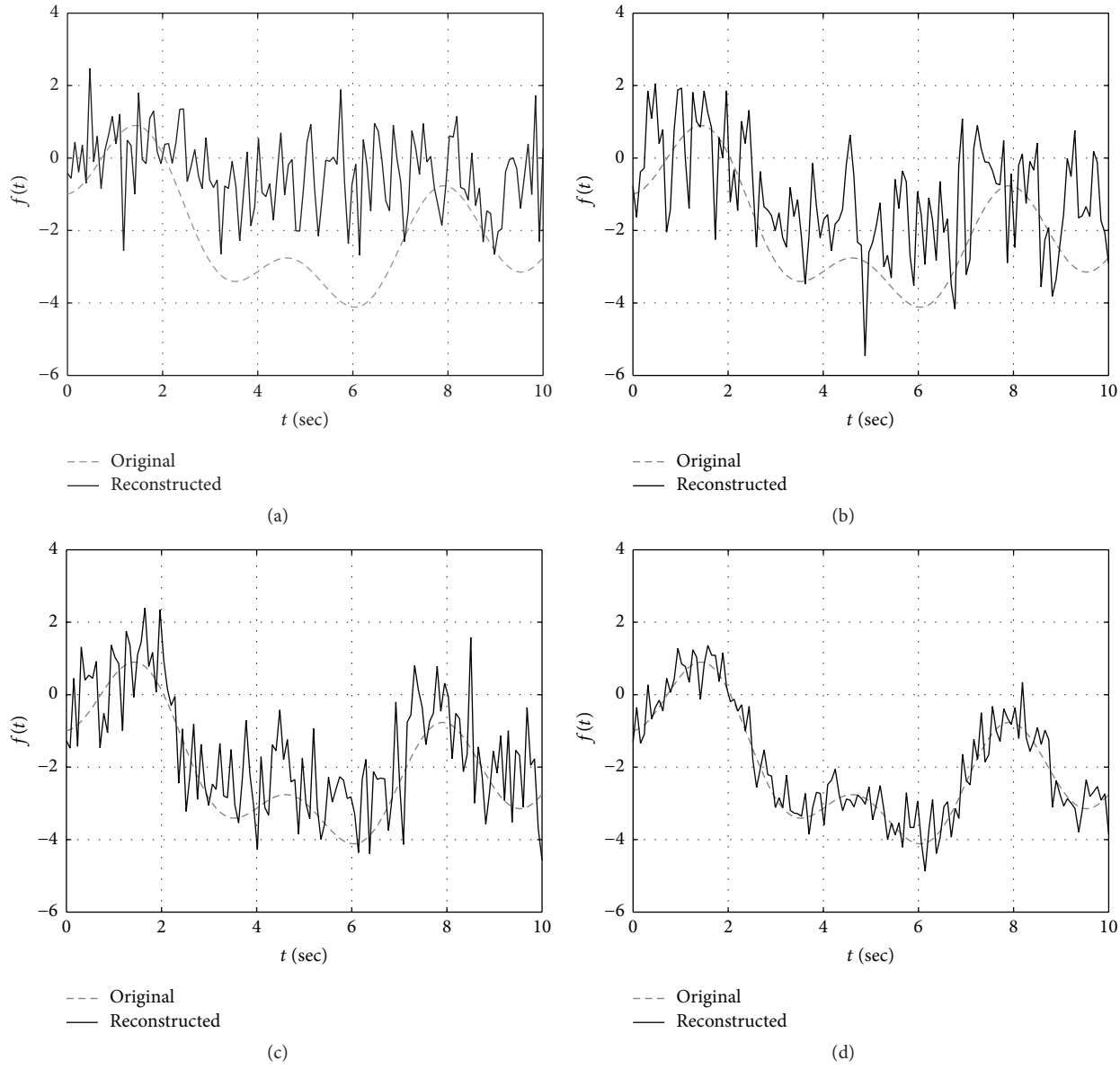


FIGURE 2: The test function reconstructed by minimizing l_2 error for compressed size of (a) 25%, (b) 50%, (c) 75%, and (d) 95% of the original data.

the Nyquist/Shannon sampling principle. As seen in Figure 3 this method does a better job at reconstructing a sparse signal than the approach that reconstructs the signal by minimizing l_2 norm. The authors of [17] introduce an approach that performs the compressed sensing reconstructing using an augmented Kalman filter (KF) [18] which is easy to implement, compared to the compressed sensing approach. The compressed sensing approach must solve a complex convex minimization problem, where problems of this type are typically being solved using an external software package. In order to use the KF it is assumed that the full set of coefficients evolve from time step to time step according to

$$\mathbf{c}_k = \mathbf{A}_k \mathbf{c}_{k-1} + \boldsymbol{\varepsilon}_k, \tag{37}$$

where $\mathbf{A}_k \in \mathbb{R}^{n \times n}$ is the state transition matrix of the system which describes how the signal coefficients change between time steps, and $\boldsymbol{\varepsilon}_k \in \mathbb{R}^n$ is a zero mean Gaussian vector with covariance $\mathbf{R}_k \geq 0$. The compressed form of the coefficients is generated according to

$$\mathbf{y}_k = \mathbf{H}_k \mathbf{c}_k + \boldsymbol{\delta}_k, \tag{38}$$

where $\mathbf{H}_k \in \mathbb{R}^{m \times n}$ is the same compression matrix discussed in (25) and $\boldsymbol{\delta}_k \in R^m$ is a zero mean Gaussian vector with covariance $\mathbf{Q}_k > 0$.

The CSEKF algorithm first performs a standard KF estimation that is carried out in a two parts, a *prediction* phase and a *correction* phase. The KF estimates $p(\mathbf{c}_k \mid \mathbf{y}_k)$ as a Gaussian distribution that can be represented using a mean

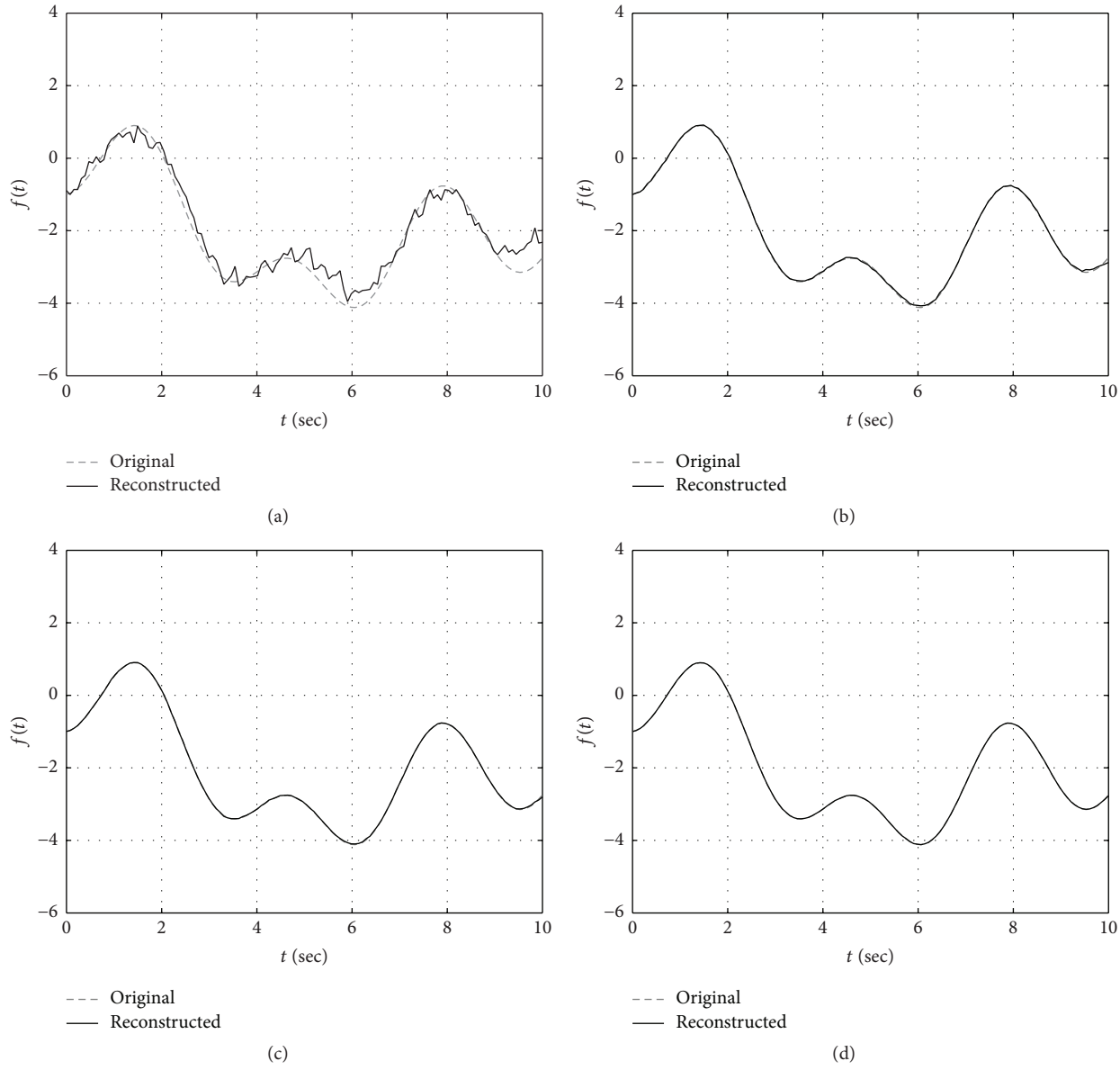


Figure 3: The test function reconstructed by solving the compressed sensing convex optimization problem for compressed size of (a) 25%, (b) 50%, (c) 75%, and (d) 95% of the original data.

vector $\widehat{\mathbf{c}}_k$ and covariance matrix $\boldsymbol{\Sigma}_k$. The KF generates the estimate using a standard set of five equations [18]:

$$\overline{\widehat{\mathbf{c}}}_k = \mathbf{A}_k \widehat{\mathbf{c}}_{k-1}, \tag{39}$$

$$\overline{\boldsymbol{\Sigma}}_k = \mathbf{A}_k \boldsymbol{\Sigma}_{k-1} \mathbf{A}_k^{\mathrm{T}} + \mathbf{R}_k, \tag{40}$$

$$\mathbf{K}_k = \overline{\boldsymbol{\Sigma}}_k \mathbf{H}_k^{\mathrm{T}} \left(\mathbf{H}_k \overline{\boldsymbol{\Sigma}}_k \mathbf{H}_k^{\mathrm{T}} + \mathbf{Q}_k \right)^{-1}, \tag{41}$$

$$\widehat{\mathbf{c}}_k = \overline{\widehat{\mathbf{c}}}_k + \mathbf{K}_k \left(\mathbf{y}_k - \mathbf{H}_k \overline{\widehat{\mathbf{c}}}_k \right), \tag{42}$$

$$\boldsymbol{\Sigma}_k = \left(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k \right) \overline{\boldsymbol{\Sigma}}_k, \tag{43}$$

where $\mathbf{I}$ in (43) is m dimensional identity matrix. The estimate produced by the KF minimizes the error, using l_2 norm, between the estimate and the actual set of coefficients

$$\min \quad \left\| \mathbf{c}_k - \widehat{\mathbf{c}}_k \right\|_2^2; \tag{44}$$

however, this is not what we are trying to solve according to the compressed sensing approach. The CSEKF presented by the authors replaces the classic compressed sensing convex optimization problem (36) with the dual problem:

$$\begin{aligned} \min \quad & \left\| \mathbf{c}_k - \widehat{\mathbf{c}}_k \right\|_2^2 \\ \text{subject to} \quad & \left\| \widehat{\mathbf{c}}_k \right\|_1 \leq \varepsilon'. \end{aligned} \tag{45}$$

They solve this constrained optimization problem by iteratively applying the pseudoobservation method presented

in [19] to the estimate generated by the KF. According to this approach a pseudoobservation is generated using the constraint being applied to the estimate which is defined as

$$\widehat{\mathbf{z}}_k = \left\|\widehat{\mathbf{c}}_k\right\|_1 - \varepsilon', \tag{46}$$

where the constraining value ε' is now treated as a Gaussian noise on the pseudomeasurement with a covariance of $\mathbf{P}$. Using the definition of $\|\cdot\|_1$ the pseudomeasurement can be written as

$$\widehat{\mathbf{z}}_k = \overline{\mathbf{H}}_k \widehat{\mathbf{c}}_k - \varepsilon', \tag{47}$$

where the observation matrix of the pseudoobservation is defined as

$$\overline{\mathbf{H}} = \left[\operatorname{sign}\left(\widehat{\mathbf{c}}_k(1)\right) \ \cdots \ \operatorname{sign}\left(\widehat{\mathbf{c}}_k(n)\right)\right],$$
$$\operatorname{sign}(x) = \begin{cases} 1 & x \geq 0 \\ -1 & \text{otherwise.} \end{cases} \tag{48}$$

The constraint is then iteratively applied τ times to the estimate produced by the KF using the standard KF correction equations (41)–(43). First, a new Kalman matrix is generated using the observation matrix of the pseudomeasurement and the covariance of the constraint:

$$\overline{\mathbf{K}} = \boldsymbol{\Sigma}_k \overline{\mathbf{H}}_k^{\mathrm{T}} \left(\overline{\mathbf{H}}_k \boldsymbol{\Sigma}_k \overline{\mathbf{H}}_k^{\mathrm{T}} + \mathbf{P}\right)^{-1}. \tag{49}$$

Next, the mean of the estimate is updated using the standard mean update equation with the assumption that the measured value is 0. Using this assumption the mean update equation can be simplified to

$$\widehat{\mathbf{c}}_k = \left(\mathbf{I} - \overline{\mathbf{K}}\,\overline{\mathbf{H}}\right)\widehat{\mathbf{c}}_k. \tag{50}$$

Finally, the covariance of the estimate is constrained using the standard covariance measurement update equation

$$\boldsymbol{\Sigma}_k = \left(\mathbf{I} - \overline{\mathbf{K}}\,\overline{\mathbf{H}}_k\right)\boldsymbol{\Sigma}_k. \tag{51}$$

The full algorithm that combines all of these separate components to reconstruct the compressed signal using the CSEKF is shown in Algorithm 5.

The reconstructed test function for the same four test percentages as in the previous two algorithms is shown in Figure 4. The algorithm is implemented as shown in Algorithm 5 with the tuning parameters being chosen as $\mathbf{Q}_k = 0.01^2$, $\mathbf{R}_k = 0$, $\mathbf{P} = 100^2$, and $\tau = 1000$. Since the input vector is static the initial estimate of the coefficient vector was zero, $\widehat{\mathbf{c}}_{k-1} = 0^n$, the state transition matrix is chosen as $\mathbf{A}_k = \mathbf{I}^n$, and the initial covariance matrix of the estimate is chosen to be $\boldsymbol{\Sigma}_{k-1} = \mathbf{I}^n$.

4.4. Kalman Filter Constrained with Quasinorm (CSEKF-p). In the previous section an approach was presented that solves a problem similar to the standard compressed sensing problem (36) using a KF that is constrained by iteratively applying a pseudoobservation to the estimate generated by the KF. The standard approach to solving the compressed sensing problem replaces $\|\cdot\|_0$ in (34) with $\|\cdot\|_1$. A new approach has been presented by the authors of [20] that replaces the zero norm with p-norm $\|\cdot\|_p$ which for $0 < p < 1$ is defined as

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{n} |\mathbf{x}(i)|^p\right)^{1/p}. \tag{52}$$

This approach has been shown to yield better accuracy than using $\|\cdot\|_1$ in some cases. Using this approach, we would like to constrain the estimate produced by the KF with the quasinorm $\|\cdot\|_p$ as opposed to $\|\cdot\|_1$.

Just as in the previous section the initial estimate is produced using the standard KF equations. The pseudoobservation used to apply the constraint to the KF estimate using $\|\cdot\|_p$ is defined as

$$\widehat{\mathbf{z}}_k = \left\|\widehat{\mathbf{c}}_k\right\|_p - \varepsilon', \tag{53}$$

where ε' is again treated as noise on the pseudoobservation with a covariance of $\mathbf{P}$. Unlike in the previous case where the constraint was linear and could be applied using the pseudoobservation matrix, the constraint applied by p-norm is nonlinear and can be rewritten as

$$\widehat{\mathbf{z}}_k = \mathbf{h}\left(\widehat{\mathbf{c}}_k\right) - \varepsilon', \tag{54}$$

where

$$\mathbf{h}\left(\widehat{\mathbf{c}}_k\right) = \left\|\widehat{\mathbf{c}}_k\right\|_p. \tag{55}$$

Since the constraint using p-norm is nonlinear the constraint is applied to the KF estimate by iteratively performing a measurement update using (54) and the Extended Kalman Filter (EKF) [21] update equations. Just as in the linear case the first step in applying the constraint is to construct the Kalman matrix according to

$$\overline{\mathbf{K}} = \boldsymbol{\Sigma}_k \overline{\mathbf{H}}_k^{\mathrm{T}} \left(\overline{\mathbf{H}}_k \boldsymbol{\Sigma}_k \overline{\mathbf{H}}_k^{\mathrm{T}} + \mathbf{P}\right)^{-1}, \tag{56}$$

(1) **procedure** CSEKF($\widehat{\mathbf{c}}_{k-1}, \boldsymbol{\Sigma}_{k-1}, \mathbf{A}_k, \mathbf{R}_k, \mathbf{H}_k, \mathbf{Q}_k, \mathbf{y}, \tau, \mathbf{P}$)
(2) $\quad\widehat{\mathbf{c}}_k \leftarrow \mathbf{A}_k \widehat{\mathbf{c}}_{k-1}$
(3) $\quad\boldsymbol{\Sigma}_k \leftarrow \mathbf{A}_k \boldsymbol{\Sigma}_{k-1} \mathbf{A}_k^{\mathrm{T}} + \mathbf{R}_k$
(4) $\quad\mathbf{K} = \boldsymbol{\Sigma}_k \mathbf{H}^{\mathrm{T}} (\mathbf{H}_k \boldsymbol{\Sigma}_k \mathbf{H}_k^{\mathrm{T}} + \mathbf{Q}_k)^{-1}$
(5) $\quad\widehat{\mathbf{c}}_k + \mathbf{K}(\mathbf{y} - \mathbf{H}_k \widehat{\mathbf{c}}_k)$
(6) $\quad$**for** $i = 1, \tau - 1$ **do**
(7) $\quad\quad\overline{\mathbf{H}} = \left[\operatorname{sign}(\widehat{\mathbf{c}}_k(1)) \ \cdots \ \operatorname{sign}(\widehat{\mathbf{c}}_k(n))\right]$
(8) $\quad\quad\overline{\mathbf{K}} = \boldsymbol{\Sigma}_k \overline{\mathbf{H}}^{\mathrm{T}} (\overline{\mathbf{H}} \boldsymbol{\Sigma}_k \overline{\mathbf{H}}^{\mathrm{T}} + \mathbf{P})^{-1}$
(9) $\quad\quad\widehat{\mathbf{c}}_k = (\mathbf{I} - \overline{\mathbf{K}}\,\overline{\mathbf{H}})\widehat{\mathbf{c}}_k$
(10) $\quad\quad\boldsymbol{\Sigma}_k = (\mathbf{I} - \overline{\mathbf{K}}\,\overline{\mathbf{H}})\boldsymbol{\Sigma}_k$
(11) $\quad$**end for**
(12) **end procedure**

Algorithm 5: Compresed sensing embedded Kalman filtering reconstruction algorithm.

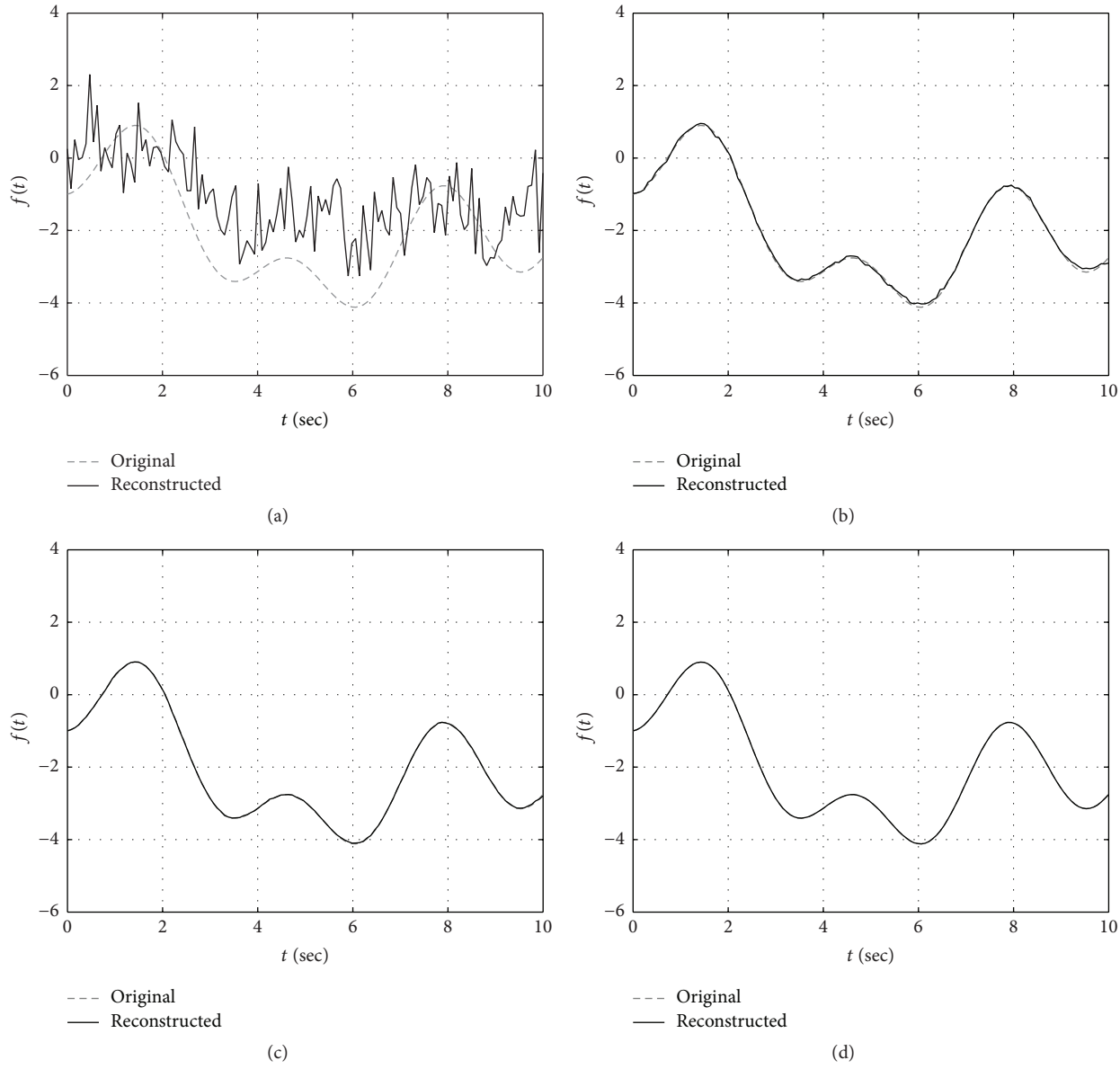


Figure 4: The test function reconstructed using the compressed sensing Kalman filter for compressed size of (a) 25%, (b) 50%, (c) 75%, and (d) 95% of the original data.

where $\overline{\mathbf{H}}$ is now the Jacobian of $\mathbf{h}(\cdot)$ and is defined as

$$\overline{\mathbf{H}}(i) = \begin{cases} \left(\sum_{i=1}^{n} \left|\widehat{\mathbf{c}}_k^*(i)\right|^p\right)^{1/(p-1)} \left|\widehat{\mathbf{c}}_k^*(i)\right|^{p-1}, & \widehat{\mathbf{c}}_k^*(i) \geq 0, \\ -\left(\sum_{i=1}^{n} \left|\widehat{\mathbf{c}}_k^*(i)\right|^p\right)^{1/(p-1)} \left|\widehat{\mathbf{c}}_k^*(i)\right|^{p-1}, & \widehat{\mathbf{c}}_k^*(i) < 0, \end{cases} \quad (57)$$

where $\widehat{\mathbf{c}}_k^* = \widehat{\mathbf{c}}_k$. The constraint is applied to the mean of the estimate using the standard EKF mean measurement update equation, again with the assumption that the measured value is zero, which when simplified is written as

$$\widehat{\mathbf{c}}_k = \widehat{\mathbf{c}}_k - \overline{\mathbf{K}} \left\|\widehat{\mathbf{c}}_k\right\|_p. \quad (58)$$

The final step in applying the constraint is to update the covariance of the estimate using the standard EKF covariance measurement update equation

$$\boldsymbol{\Sigma}_k = \left(\mathbf{I} - \overline{\mathbf{K}}\,\overline{\mathbf{H}}_k\right)\boldsymbol{\Sigma}_k. \quad (59)$$

The complete algorithm for reconstructing a compressed signal using the estimate produced by the KF constrained using p-norm is shown in Algorithm 6.

The reconstruction results for the compressed sensing Kalman filter using a quasinorm to constrain the estimate are presented in Figure 5 for the same four data sizes used for the three previous algorithms. A parameter that greatly effects the performance of the algorithm is the value chosen for p. To determine the best value of p the algorithm was

(1) **procedure** CSEKFP($\widehat{\mathbf{c}}_{k-1}, \mathbf{\Sigma}_{k-1}, \mathbf{A}_k, \mathbf{R}_k, \mathbf{H}_k, \mathbf{Q}_k, \mathbf{y}, \tau, \mathbf{P}, p$)
(2) $\quad \widehat{\mathbf{c}}_k \leftarrow \mathbf{A}_k \widehat{\mathbf{c}}_{k-1}$
(3) $\quad \mathbf{\Sigma}_k \leftarrow \mathbf{A}_k \mathbf{\Sigma}_{k-1} \mathbf{A}_k^{\mathrm{T}} + \mathbf{R}_k$
(4) $\quad \mathbf{K} = \mathbf{\Sigma}_k \mathbf{H}^{\mathrm{T}} (\mathbf{H}_k \mathbf{\Sigma}_k \mathbf{H}_k^{\mathrm{T}} + \mathbf{Q}_k)^{-1}$
(5) $\quad \widehat{\mathbf{c}}_k + \mathbf{K}(\mathbf{y} - \mathbf{H}_k \widehat{\mathbf{c}}_k)$
(6) $\quad$ **for** $i = 1, \tau - 1$ **do**
(7) $\quad\quad$ **for** $j = 1, n$ **do**
(8) $\quad\quad\quad$ **if** $\widehat{\mathbf{c}}_k(j) \geq 0$ **then**
(9) $\quad\quad\quad\quad \overline{\mathbf{H}}(j) = \left(\sum_{i=1}^{n} |\widehat{\mathbf{c}}_k^*(i)|^p\right)^{1/(p-1)} |\widehat{\mathbf{c}}_k^*(i)|^{p-1}$
(10) $\quad\quad\quad$ **else**
(11) $\quad\quad\quad\quad \overline{\mathbf{H}}(i) = -\left(\sum_{i=1}^{n} |\widehat{\mathbf{c}}_k^*(i)|^p\right)^{1/(p-1)} |\widehat{\mathbf{c}}_k^*(i)|^{p-1}$
(12) $\quad\quad\quad$ **end if**
(13) $\quad\quad$ **end for**
(14) $\quad\quad \overline{\mathbf{H}} = \left[\mathrm{sign}(\widehat{\mathbf{c}}_k(1)) \quad \cdots \quad \mathrm{sign}(\widehat{\mathbf{c}}_k(n))\right]$
(15) $\quad\quad \overline{\mathbf{K}} = \mathbf{\Sigma}_k \overline{\mathbf{H}}^{\mathrm{T}} (\overline{\mathbf{H}} \mathbf{\Sigma}_k \overline{\mathbf{H}}^{\mathrm{T}} + \mathbf{P})^{-1}$
(16) $\quad\quad \widehat{\mathbf{c}}_k = (\mathbf{I} - \overline{\mathbf{K}}\,\overline{\mathbf{H}}) \widehat{\mathbf{c}}_k$
(17) $\quad\quad \mathbf{\Sigma}_k = (\mathbf{I} - \overline{\mathbf{K}}\,\overline{\mathbf{H}}) \mathbf{\Sigma}_k$
(18) $\quad$ **end for**
(19) **end procedure**

Algorithm 6: Sparse signal reconstruction algorithm using the KF constrained using the quasinorm.

run for a single data percentage, 75%, for 20 equally spaced values of $p \in (0, 1)$. A plot of the mean square error (MSE) between the compressed signal and original signal can be seen in Figure 6. From this graph the value of p was chosen to be 0.4. The remainder of the tuning parameters for the algorithm were selected to be the same as those used for the CSEKF reconstruction.

4.5. Algorithm Comparison. Before deciding which reconstruction algorithm should be used in the FastSLAM COG algorithm we examine the performance of each of the algorithms in two key areas: accuracy and run time. First, we examine how well each algorithm reproduces the original signal based on the amount of information that is stored in the compressed signal. To compare how well each algorithm reconstructs the signal, each algorithm was run for 20 equally spaced percentages in (0%, 100%]. The results for each of the algorithms are shown in Figure 7 which plots the MSE as a function of the amount of data stored. As seen in the results, in general each of the compressed sensing type approaches produces less of an error for a given data percentage than l_2 approach.

The second key characteristic of each algorithm examined is the time it takes for each algorithm to run. During the error performance testing described above, the time taken to run each algorithm was recorded. The run time as a function of the amount of compressed data used to represent the signal can be seen in Figure 8. As seen in the timing results the two algorithms that use the augmented Kalman filter have run times that are significantly larger than the other approaches. This increased run time is due to the number of matrix-matrix multiplications required by the Kalman filter which are computationally heavy, from [22] having $\mathcal{O}(n^{\log_2 7})$. The remaining two algorithms, l_2 and compressed sensing, can be implemented completely using only matrix-vector multiplications, which can be performed more quickly than matrix-matrix multiplications.

5. Performance Improvements

Our overall goal is to augment the FastSLAM algorithm to use compressed occupancy grids. In order for us to accomplish this task the reconstruction approach selected must run thousands or hundreds of thousands of times during the operation of a UMV. By using a Rao-Blackwellized particle filter, we can increase the probability that our SLAM posterior estimate matches the true SLAM posterior by increasing the number of particles maintained by the particle filter. As seen in Algorithm 3 the reconstruction process must be performed by each particle at each iteration; thus the execution time of the reconstruction process is important in the overall performance of the algorithm.

In Section 4 four different algorithms were presented that can be used to reconstruct an occupancy grid from the compressed form. From the timing results (Figure 8), the two approaches that make use of the Kalman filter (CSEKF and CSEKF-p) take much longer time to run than l_2 and CS. The longer run times of the Kalman filter based approaches are due to the large number of matrix-matrix multiplications. Based on their performance the CSEKF and CSEKF-p algorithms will not be used in FastSLAM COG. We will now present a method that can be used to improve the accuracy of the remaining two algorithms and a method that improves the run time of l_2 approach.

5.1. Updated Compression Step. As seen in Section 3.2 the compressed occupancy grid is generated by selecting a subset

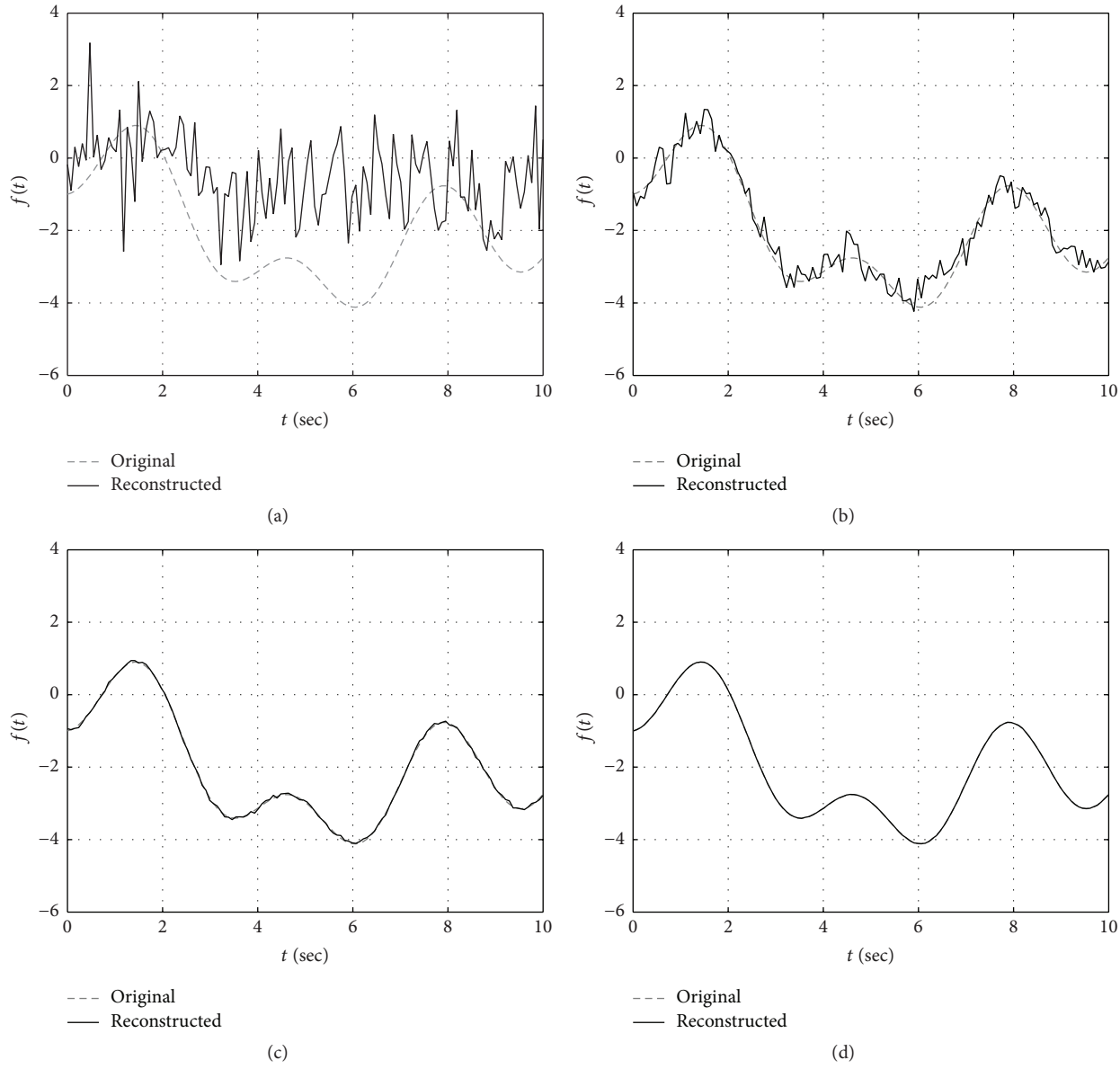


Figure 5: The test function reconstructed using the quasinorm constrained Kalman filter for compressed size of (a) 25%, (b) 50%, (c) 75%, and (d) 95% of the original data.

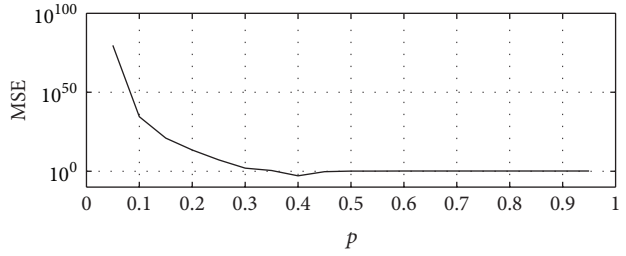


Figure 6: MSE for different values of p for a single compressed percentage 75%.

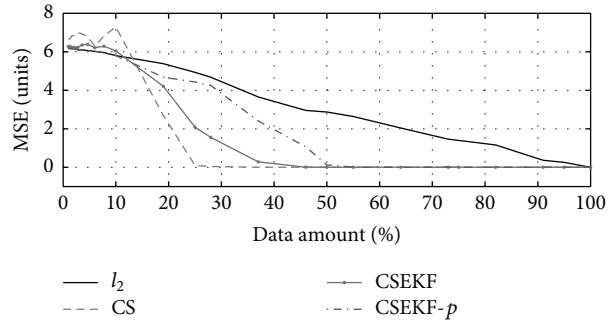


Figure 7: MSE as a function of signal data used for each of the four reconstruction algorithms.

of the information stored in the coefficients. This compression is performed according to (15) which makes use of $\mathbf{H}$ to perform the compression. As stated in Section 4.2 we would like for this compression matrix to obey the RIP property.

Several common matrices that are used in compressed sensing applications are the discrete Fourier transform [23],

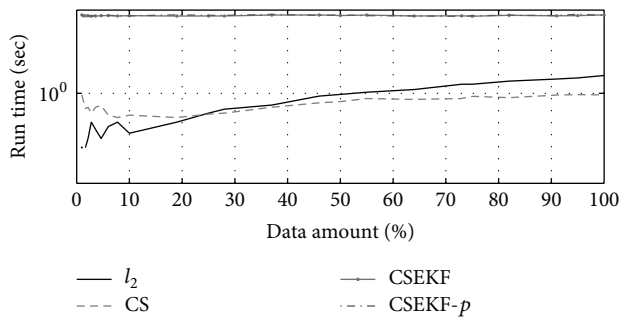


Figure 8: Run time as a function of compressed percentage for each of the four reconstruction algorithms.

the discrete cosine transform [10], the Hadamard transform [24], and noiselet transform [25]. Each of these matrices is constructed in such a way that each of the elements stored in the compressed vector stores a small amount of information about each of the coefficients used to represent the original signal and this information can be recovered by knowing $\mathbf{H}$. These types of matrices are useful when there is no knowledge about the sparse signal that is being compressed; however we are constructing the sparse signal thus we have full knowledge of the signal before the compression is performed. Using this knowledge we present an alternative form of $\mathbf{H}$ that allows for better reconstruction using a smaller amount of information. The form of $\mathbf{H}$ selected comes from the definition of the square error defined according to l_2 norm. For some discrete signal $\mathbf{x} \in \mathbb{R}^n$ the square of the error between the signal and the approximation of the signal $\widehat{\mathbf{x}}$ generated using $m < n$ elements is defined as

$$\|\mathbf{x} - \widehat{\mathbf{x}}\|_2^2 = \langle \mathbf{x} - \widehat{\mathbf{x}} \mid \mathbf{x} - \widehat{\mathbf{x}} \rangle, \tag{60}$$

where $\langle \cdot \mid \cdot \rangle$ is the standard inner product. The discrete signal $\mathbf{x} \in \mathbb{R}^n$ can be represented in some alternate basis so that a single element can be expressed as

$$\mathbf{x}(i) = \sum_{j=1}^{n} \mathbf{\Phi}(i, j)\, \mathbf{c}(j). \tag{61}$$

Recalling that $\widehat{\mathbf{y}} \in \mathbb{R}^m$ denotes the compressed set of coefficients (25), a single element of the compressed signal can be expressed in the alternate basis as

$$\widehat{\mathbf{x}}(i) = \sum_{j=1}^{m} \widehat{\mathbf{\Phi}}(i, j)\, \widehat{\mathbf{c}}(j), \tag{62}$$

where

$$\begin{aligned} \widehat{\mathbf{\Phi}} &= \mathbf{\Phi}\mathbf{H}^{+}, \\ \widehat{\mathbf{c}} &= \mathbf{y}. \end{aligned} \tag{63}$$

While this is a valid approach for compressing a signal, we would like to simplify the compression process for performance reasons. Instead of generating a new set of basis function it is easier to keep the basis functions the same at all times and just select a subset of the coefficients to represent the signal. In order to decide which coefficients should be selected that minimize the error based on l_2 norm, we define several sets. The set $\boldsymbol{\zeta} = \{x \mid x \in \mathbb{N}, 1 \le x \le n\}$ is the full set of indices of the discrete vector being compressed. The set $\boldsymbol{\lambda} \in \mathbb{N}^m$ is the set of indices that are used to represent the compressed signal, $\boldsymbol{\lambda} \subset \boldsymbol{\zeta}$. Finally, the set $\boldsymbol{\gamma} \in \mathbb{N}^{n-m}$ is the set of indices of coefficients not used in the compressed signal, $\boldsymbol{\gamma} = \{x \mid x \in \boldsymbol{\zeta} \text{ and } x \notin \boldsymbol{\lambda}\}$. Using these sets and the fact that a constant basis is used to represent the uncompressed and compressed signal an element in the reconstructed signal can be written as

$$\widehat{\mathbf{x}}(i) = \sum_{j=1}^{m} \mathbf{\Phi}(i, \boldsymbol{\lambda}(j))\, \mathbf{c}(\boldsymbol{\lambda}(j)). \tag{64}$$

The square of the error in l_2 norm for a single element becomes

$$\begin{aligned} \|\mathbf{x}(i) - \widehat{\mathbf{x}}(i)\|_2^2 &= \langle \mathbf{x}(i) - \widehat{\mathbf{x}}(i) \mid \mathbf{x}(i) - \widehat{\mathbf{x}}(i) \rangle \\ &= \left\langle \sum_{j=1}^{n-m} \mathbf{\Phi}(i, \boldsymbol{\gamma}(j))\, \mathbf{c}(\boldsymbol{\gamma}(j)) \mid \right. \\ &\left. \sum_{k=1}^{n-m} \mathbf{\Phi}(i, \boldsymbol{\gamma}(k))\, \mathbf{c}(\boldsymbol{\gamma}(k)) \right\rangle = \sum_{j=1}^{n-m} \sum_{k=1}^{n-m} \mathbf{c}(\boldsymbol{\gamma}(j))\, \mathbf{c}(\boldsymbol{\gamma}(k)) \\ &\times \langle \mathbf{\Phi}(i, \boldsymbol{\gamma}(j)) \mid \mathbf{\Phi}(i, \boldsymbol{\gamma}(k)) \rangle = \sum_{j=1}^{n-m} \mathbf{\Phi}(i, \boldsymbol{\gamma}(j))^2 \\ &\cdot \mathbf{c}(\boldsymbol{\gamma}(j))^2, \end{aligned} \tag{65}$$

and if we assume that the set of basis function is orthonormal, which is common, then l_2 error for the entire compressed signal becomes

$$\|\mathbf{x} - \widehat{\mathbf{x}}\|_2^2 = \sum_{i=1}^{n} \sum_{j=1}^{n-m} \mathbf{c}(\boldsymbol{\gamma}(j))^2. \tag{66}$$

This tells us that the best way to minimize the error in the reconstructed signal is to select the coefficients that have the largest magnitudes. Using this knowledge a new compression algorithm that does not require matrix-vector multiplications and selects only m coefficients with the largest magnitudes was developed and shown in Algorithm 7.

To examine how the use of the new compression algorithms affects the performance of the reconstruction algorithms, the occupancy grid shown in Figure 9(a) is reconstructed from the compressed form. The compression matrix selected for the baseline results is different from the orthonormal Gaussian random matrix used in Section 4; in its place a normalized Hadamard matrix was used. Each of the two algorithms was then run using each of the two compression methods for 20 separate compression percentages in $(0, 100\%)$. The performance results of each of the two algorithms are provided in Figure 10. As seen from the results for each of the two reconstruction algorithms the occupancy grid is reconstructed with a smaller MSE using the

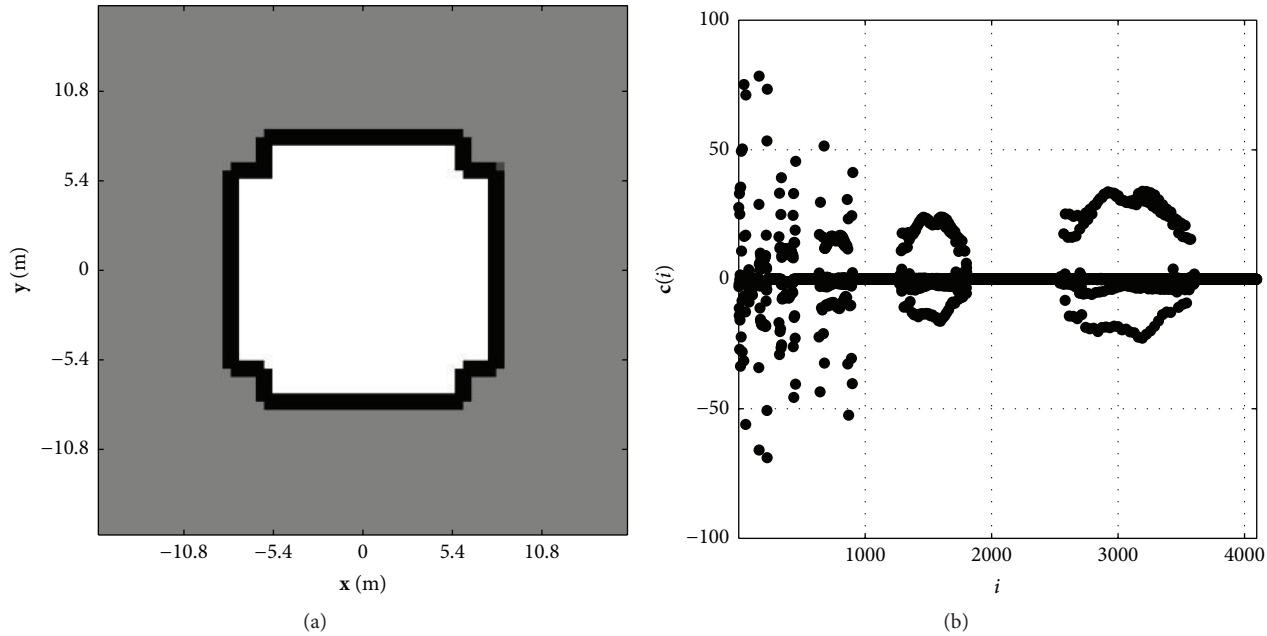


Figure 9: An example occupancy grid for an unknown environment (a) and the set of coefficients used to represent the occupancy grid when represented using the Haar wavelet basis (b).

```
(1) procedure FastCompression(c, m)
(2)     x = sort(abs(c), "descend")
(3)     for i = 1, m do
(4)         i(i) = x(i)
(5)         y(i) = c(x(i))
(6)     end for
(7) end procedure
```

Algorithm 7: Updated compression algorithm that represents the compressed occupancy grids with a set of coefficients and corresponding indices.

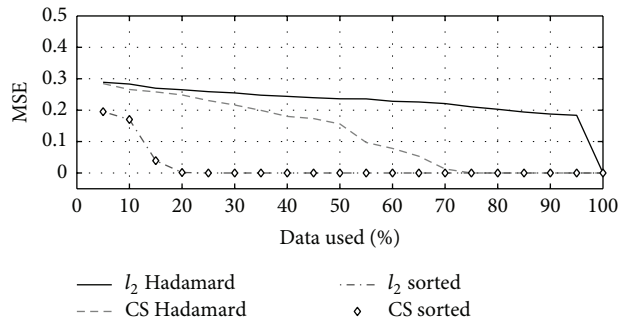


Figure 10: Performance results of each of the reconstruction methods using a normalized Hadamard matrix and a selection matrix to keep m most significant coefficients.

compression matrix that keeps m most significant coefficients as opposed to the compression step using the normalized Hadamard matrix using the same amount of data.

5.2. Updated l_2 Reconstruction. In Section 4.1 a method for reconstructing a compressed signal was introduced that minimizes the error according to l_2 norm. The method makes uses of the Moore-Penrose pseudoinverse to reconstruct the set of coefficients from their compressed form according to

$$\widehat{\mathbf{c}} = \mathbf{H}^{\dagger}\mathbf{y}. \tag{67}$$

This method of reconstructing the signal is made up of two steps where the performance of each step is based on the size of the signal (n). The first step in the process is the calculation of the pseudoinverse of the compression matrix $\mathbf{H}$. Many common software implementations of this calculation, for example, Matlab's pinv( ) function [26], makes use of the Singular Value Decomposition (SVD) of the matrix. The process of calculating the SVD of a matrix is worse from a performance perspective than that of matrix-matrix multiplication which is $\mathcal{O}(mn^2)$ [27] for $m \times n$ matrix.

The second step of the algorithm is a matrix-vector multiplication operation which in a naive implementation has a performance of $\mathcal{O}(n^2)$. We would like to improve the performance of the reconstruction algorithm to get rid of these computationally intensive steps. By making use of the compression matrix developed in the previous section that just selects m coefficients with the largest magnitudes, the performance of the reconstruction process can be improved. Each of the rows in the compression matrix $\mathbf{H}$ is orthogonal; that is, for two rows in the selection matrices $\mathbf{H}(i,:)$ and $\mathbf{H}(j,:)$ the discrete inner product is zero; $\langle \mathbf{H}(i,:) \mid \mathbf{H}(j,:) \rangle = 0$. A second property is that each row of $\mathbf{H}$ is normalized; $\sum_{j=1}^{n} \mathbf{H}(i,j) = 1$; thus $\mathbf{H}$ is orthonormal. The pseudoinverse

```
(1) procedure FastLTWOReconstruction(c, m)
(2)     for i = 1, n do
(3)         c(i) = 0
(4)     end for
(5)     for i = 1, m do
(6)         c(i(i)) = y(i)
(7)     end for
(8) end procedure
```

Algorithm 8: Updated form of l_2 reconstruction algorithm that is performed without needing to calculate the pseudoinverse of the compression matrix or perform the matrix-vector multiplication.

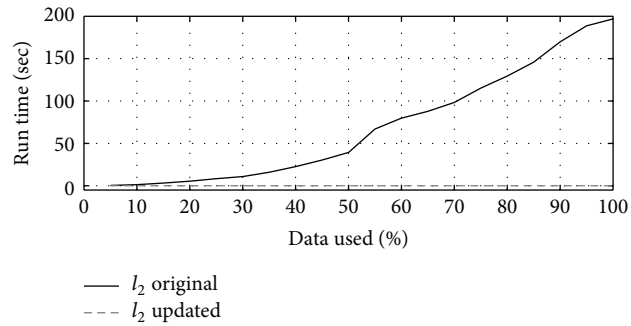


Figure 11: The comparison in run time for the original l_2 reconstruction along with the updated form of the reconstruction algorithm.

of a orthonormal matrix can be easily computed without the computationally intensive SVD process according to

$$\mathbf{H}^{\dagger} = \mathbf{H}^{*}, \tag{68}$$

where $\mathbf{H}^{*}$ is the conjugate transpose of $\mathbf{H}$ and is found by taking the transpose of $\mathbf{H}$ followed by the complex conjugate of each element in $\mathbf{H}$. Since the compression matrix is composed of elements containing just 0 or 1 the pseudoinverse is just the transpose $\mathbf{H}^{\dagger} = \mathbf{H}^{\mathrm{T}}$.

Finally, in (67) we can replace the pseudoinverse of $\mathbf{H}$ with the transpose; thus the reconstruction process becomes

$$\widehat{\mathbf{c}} = \mathbf{H}^{\mathrm{T}}\mathbf{y}. \tag{69}$$

We can improve the performance even more if we assume that the compression is performed as discussed above, that is, where m coefficients with the largest magnitudes are selected. In that case then $\mathbf{H}$ is full of zeros except for elements that correspond to the coefficients with largest magnitudes. Using this information we simplify l_2 reconstruction algorithm to just generate a vector of zeros and then copy the stored coefficients into their proper locations in the reconstructed vector. This replaces the matrix-vector multiplication with m value copies. The updated form of l_2 reconstruction algorithm that does not require the potentially expensive calculation of the matrix pseudoinverse or matrix-vector multiplication is presented in Algorithm 8.

To examine the performance benefit of the new l_2 reconstruction algorithm a compressed form of the occupancy grid seen in Figure 11 is reconstructed using the original algorithm explained in Section 4.1 and the new algorithm described in Algorithm 8. The algorithm is run for x varying compressed data sizes in $[0\%, 100\%]$ and the timing results are seen in Figure 11. As seen from the results the run time for the new l_2 reconstruction remains small as the size of the compressed signal grows while the original l_2 reconstruction approach slows down as the size of the compressed signal grows.

6. Experimental Results

We would now like to examine how using compressed occupancy grids affects the FastSLAM OG algorithm. Based on the timing results presented in Section 4 only two of the uncompression methods, l_2 (Section 4.1) and CS (Section 4.2), were selected to use in the FastSLAM COG algorithm. Both of the algorithms are compared using data captured by a small unmanned ground vehicle (UGV). The UGV is equipped with Mecanum wheels [28] which allow the vehicle to move in any direction without the constraints that are typical with many ground vehicles. The use of Mecanum wheels makes it difficult to use standard wheel encoders for localization purposes; to overcome this a downward facing camera is attached to the UGV which provides visual odometry data for localization using the approach described in [29]. Other sensors attached to the UGV include a digital compass to provide the global heading of the vehicle and a Hokuyo UTM-30LX LiDAR sensor, with reported accuracy of 0.1 to 10 m ± 30 mm and 10 to 30 m ± 50 mm, to provide range and bearing measurements to objects in the environment. In order to provide a "ground truth" trajectory for our test, a Hagisonic StarGazer indoor localization system was used. This sensor uses an upward facing camera and static landmarks placed on the ceiling to triangulate a robotic vehicle in an environment. This sensor has a precision between 1 mm and 10 mm based on how many landmarks the sensor can see at a given time. This accuracy was much higher than the visual odometry algorithm that provided the odometry input to the SLAM algorithm; thus this trajectory was used to compare the effect of using compressed occupancy grids in our FastSLAM algorithm as opposed to the full occupancy grid.

For comparison purposes, the standard FastSLAM OG algorithm described in Algorithm 1 was used to generate a baseline vehicle path and occupancy grid map. Once the baseline was generated the FastSLAM COG algorithm described in Algorithm 3 was run using the same data used by the standard approach. For each of the two reconstruction algorithms the compression method used is the optimized approach described in Section 5.1 and l_2 approach is implemented as described in Section 5.2. For the CS reconstruction algorithm the convex optimization problem is solved using the NESTA library.

The UGV was remotely driven around a small enclosed 10 ft × 12 ft environment while all of the sensor data was logged to memory on a small attached computer. The data was then postprocessed to produce a baseline vehicle path and an occupancy grid representing the testing environment. The baseline vehicle path, along with the true vehicle path, can be

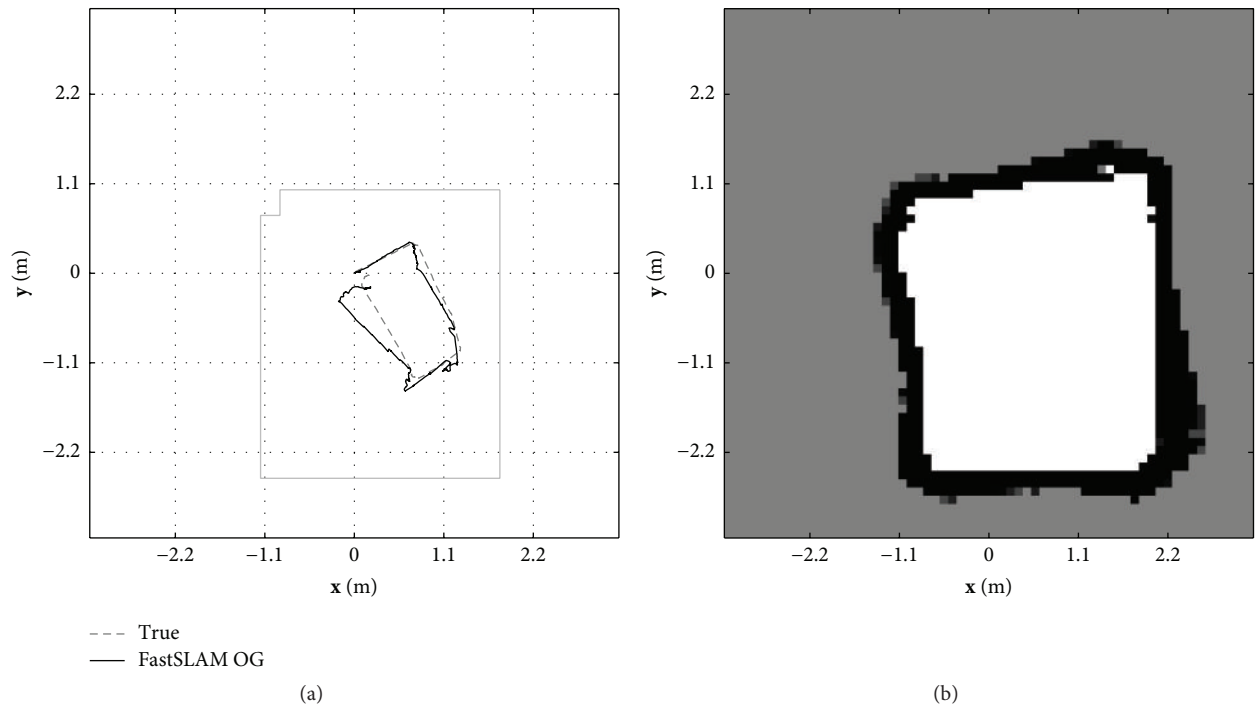


Figure 12: The baseline path generated by the FastSLAM algorithm with occupancy grids algorithm along with the true path of the vehicle (a) and the occupancy grid generated by the algorithm that represents the environment in which the vehicle is operating (b).

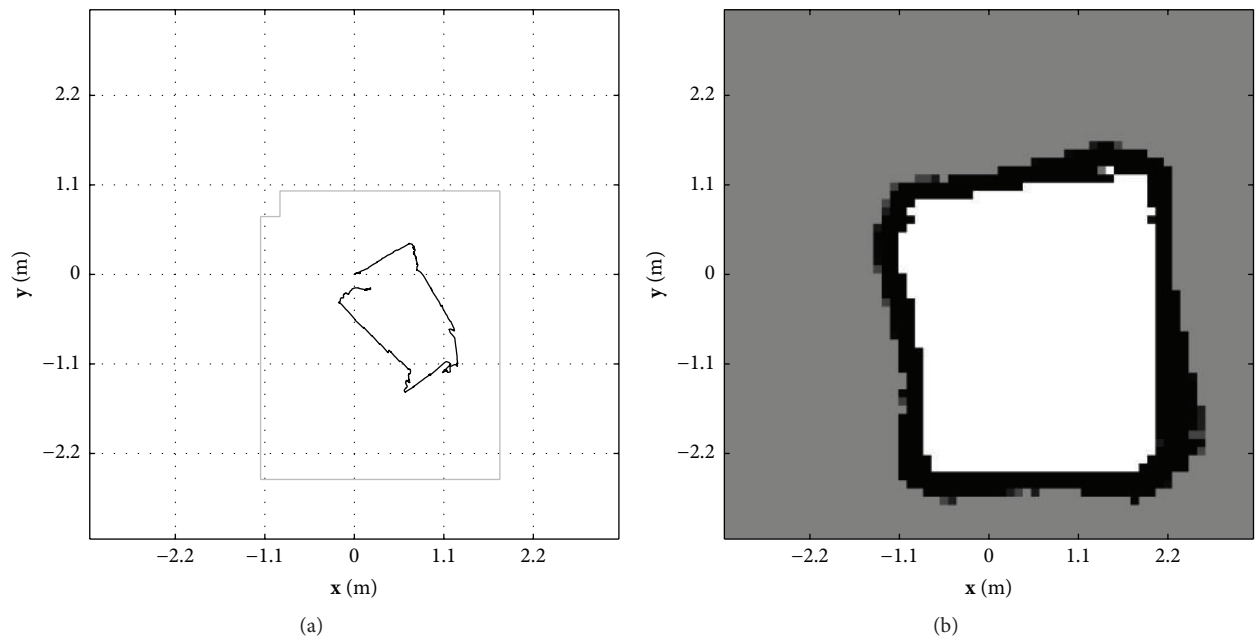


Figure 13: The path generated by the FastSLAM COG algorithm along with l_2 reconstruction method using 40% of the original data (a) and the reconstructed occupancy grid generated by the algorithm (b).

seen in Figure 12(a) and the occupancy grid with a 0.1 m cell size is shown in Figure 12(b). The environment used to generate the baseline shown in Figures 12(a) and 12(b) is a simple rectangular indoor environment. Our choice of this simple environment, as opposed to a more complex environment, was made for two reasons. First, in order to properly compare how the use of compressed occupancy grids affected the localization and mapping an accurate baseline was required for comparison purposes. Based on the sensors that we had at our disposal, an indoor testing was more ideally suited for this task. Secondly, the goal of this research was not to examine how well an occupancy grid based SLAM algorithm

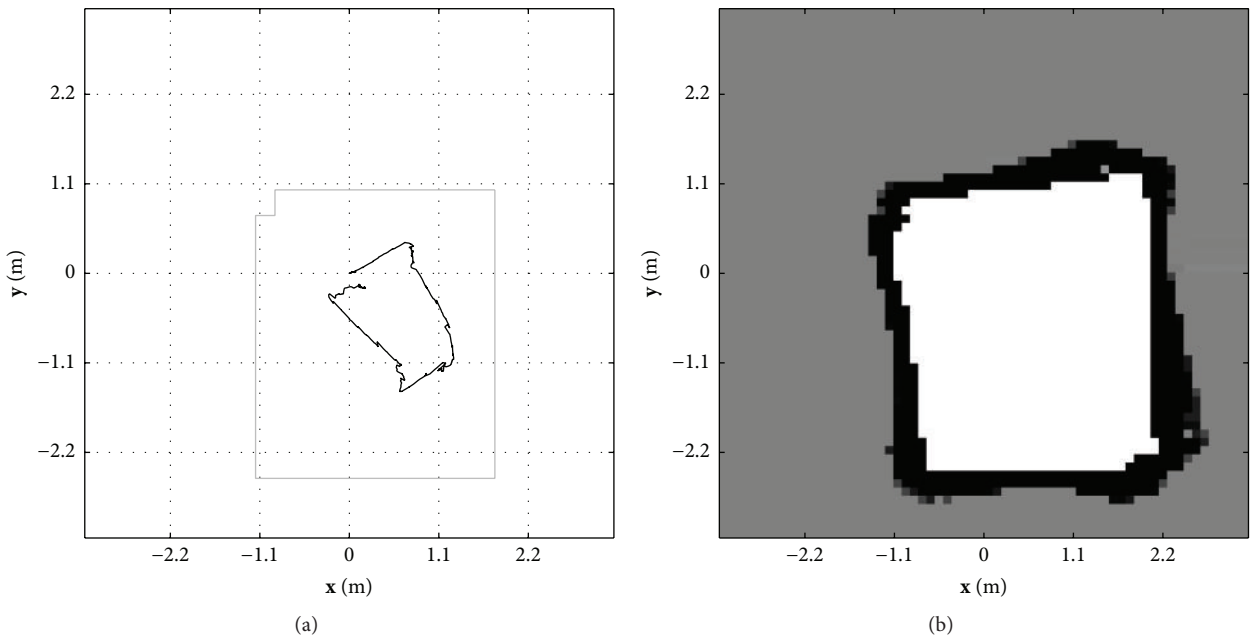


Figure 14: The path generated by the FastSLAM COG algorithm using 40% of the original data (a) and the reconstructed occupancy grid generated by the algorithm (b).

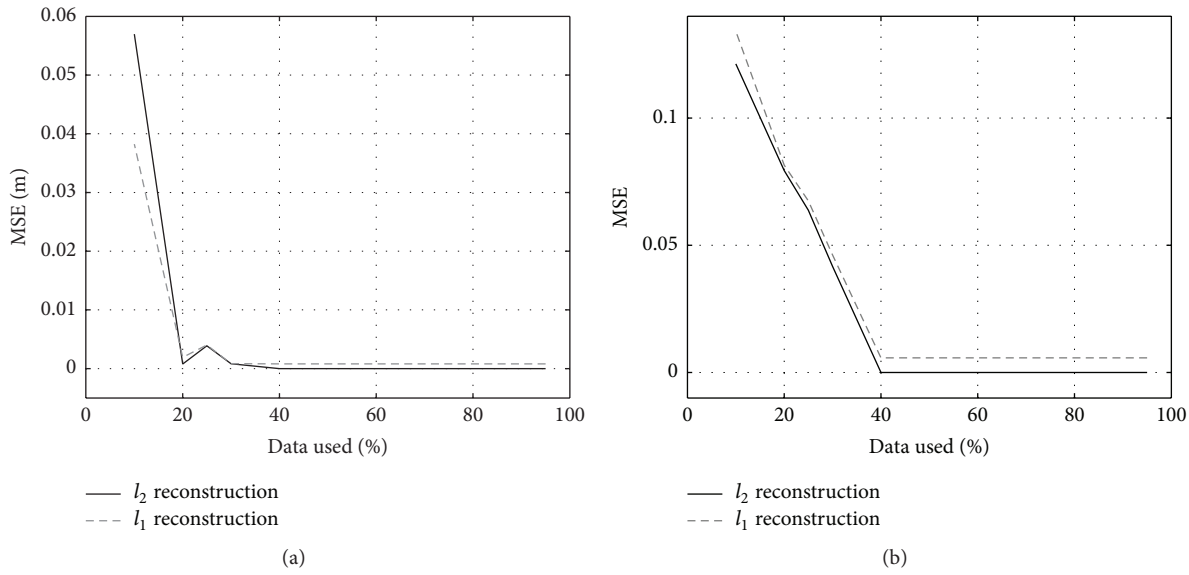


Figure 15: The MSE between the path generated by the original algorithm and that generated by the compressed version as a function of the amount of data stored (a) and the MSE between the occupancy grid generated by the original algorithm and that generated using the compressed form as a function of the amount of data stored (b).

performs in complex environment that was beyond our scope and addressed by others; rather our goal was to examine how the performance of the approach was affected by compressing the occupancy grid used to represent the environment, thus testing in complex environment was not attempted for this initial research.

The vehicle path estimate generated by FastSLAM COG algorithm using l_2 reconstruction can be seen in Figure 13(a) and the reconstructed occupancy grid is shown in Figure 13(b). The generated path estimate using compressed sensing reconstruction approach is shown in Figure 14(a) and the reconstructed occupancy grid is shown in Figure 14(b). For each of the algorithms 40% of the amount of data used to represent the full occupancy grid is used. Figure 15 shows the mean square error (MSE) in the path estimate and reconstructed occupancy grid between the uncompressed form of the algorithm and the compressed form of the algorithm. As Figure 15 shows using more than 40% of

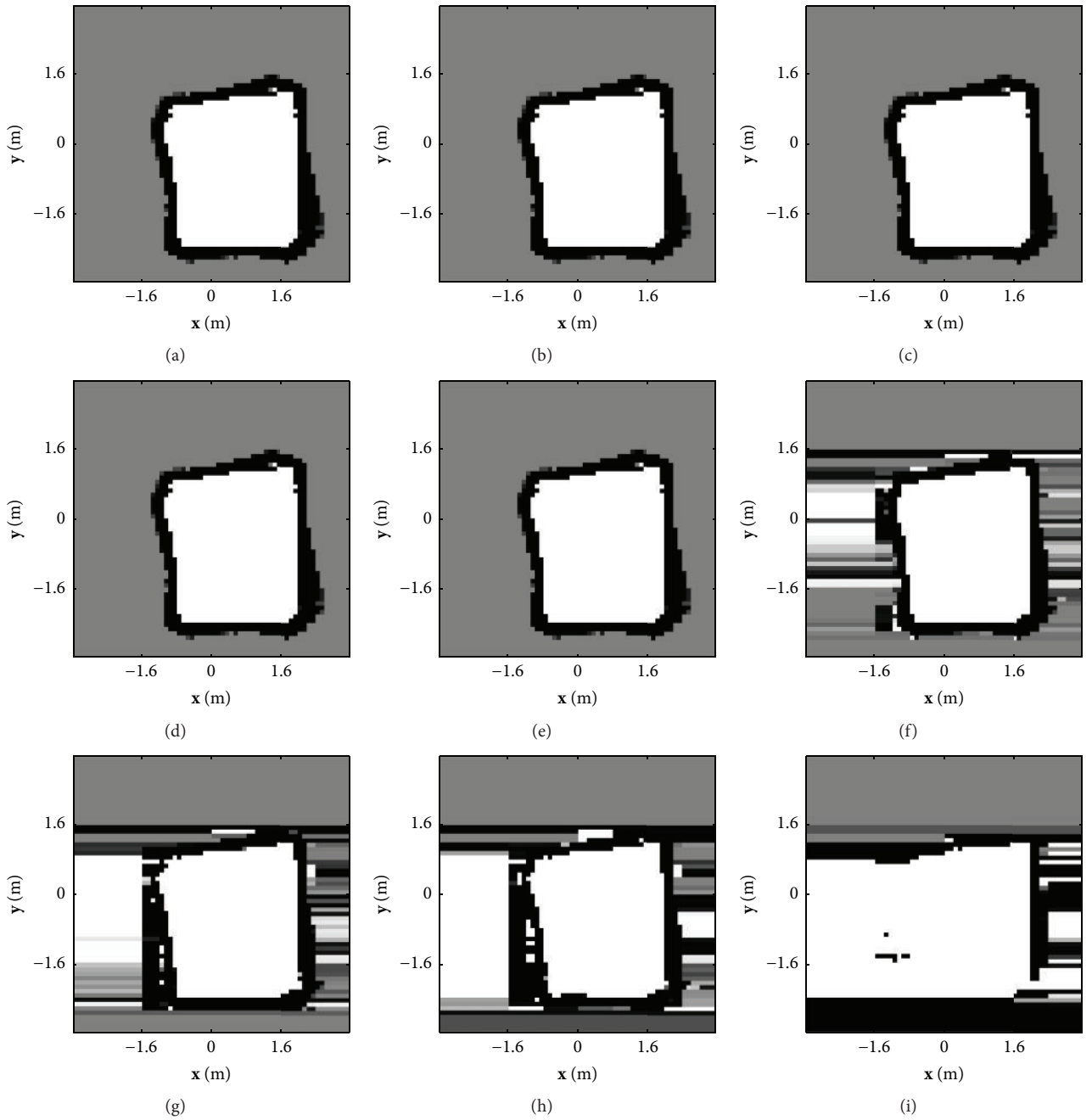


Figure 16: A series of final occupancy grids generated by the FastSLAM COG algorithm using l_2 reconstruction approach using 75% (a), 70% (b), 60% (c), 50% (d), 40% (e), 30% (f), 25% (g), 20% (h), and 10% (i) of the full amount of data.

the data does not decrease the MSE any more than the error that exists when using 40% of the data.

The plots provided in Figure 16 illustrate how a decrease in the amount of stored data decreases the quality of the reconstructed occupancy grid. This set of plots shows a series of occupancy grids that were generated as the final map of by the FastSLAM COG algorithm using l_2 reconstruction approach for decreasing amounts of data. As can be seen from the first several grids, up to 40% of the data show no decrease in quality; however as the amount of data used continues to decrease the amount of error in the final occupancy grid increases until the grid becomes unusable. A similar set of plots are provided in Figure 17 for the FastSLAM COG algorithm using the compressed sensing reconstruction approach and similar results can be seen.

By examining Figure 15 it can be seen that there is a constant error that appears in both the path estimate and the occupancy grid reconstruction from using the compressed sensing approach. This error comes from the parameters that are chosen for the NESTA library. Of most significance

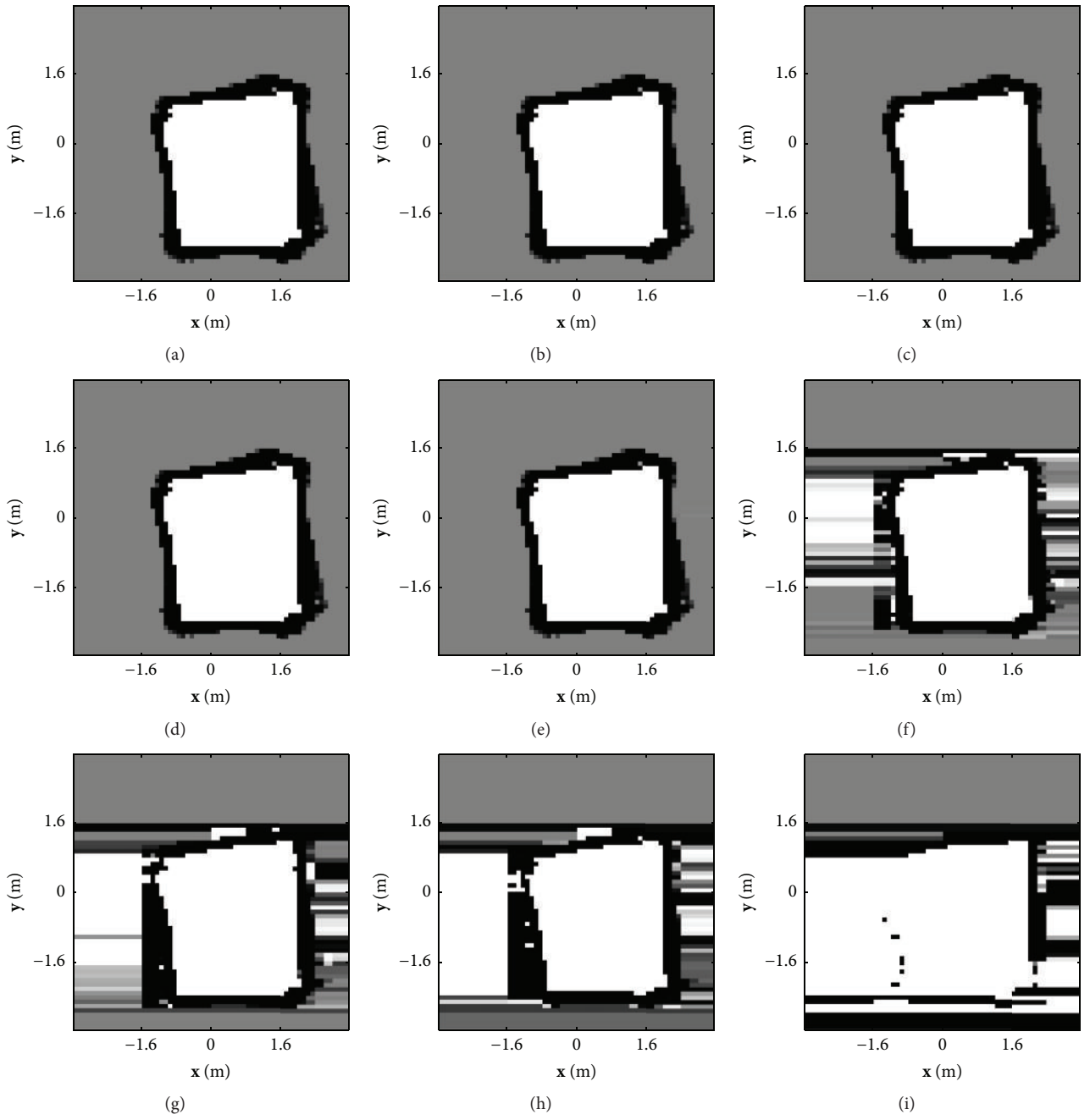


Figure 17: A series of final occupancy grids generated by the FastSLAM COG algorithm using the compressed sensing reconstruction approach using 75% (a), 70% (b), 60% (c), 50% (d), 40% (e), 30% (f), 25% (g), 20% (h), and 10% (i) of the full amount of data.

is the value of ε from (36) which is the amount of error that is allowed between the coefficients that represent the compressed form of the occupancy grid and the value of those coefficients in the reconstructed occupancy grid. For the presented results the value of ε was chosen to be 10^{-10} which is quite small; however it does allow for small errors to occur in the reconstructed occupancy grid. These errors can be reduced or removed completely, from the estimate by choosing a smaller value of ε and adjusting the parameters used by the NESTA library. However, as previously discussed, the accuracy of the grid reconstruction is not the only property of importance to us when deciding which approach should be used for storing and reconstructing the compressed occupancy grid. The other significant property of each algorithm that must be investigated is the time it takes to execute the algorithm. A comparison of the average time it takes to run a single iteration of the FastSLAM COG algorithm using l_2 approach and the compressed sensing approach for varying amounts of data is shown in Figure 18. As seen in these results the average run time of the FastSLAM

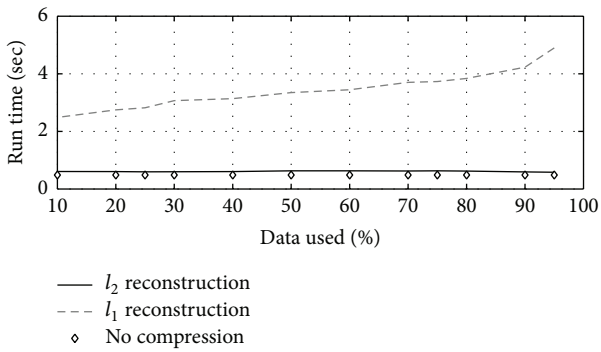


Figure 18: The average time to complete a single iteration of the FastSLAM COG algorithm using l_2 reconstruction approach and the compressed sensing reconstruction approach for varying amounts of data stored.

COG algorithm using the compressed sensing approach is significantly longer at each compression level than the algorithm using l_2 reconstruction approach.

6.1. How Much Compression Can Be Achieved? There are two key components that affect how much an occupancy grid can be compressed; first is the complexity of the environment in which the UMV is operating. If the environment is simple, such as a rectangular room, then a relatively small amount of data is needed to represent it with respect to a more complex environment. The second component that affects how much an occupancy grid can be compressed is the cell size used. If a large cell size is selected then there are fewer cells to represent the environment compared to when a smaller cell size is selected. It follows that when a larger cell size is selected a larger percentage of the cells must be used to store the information about the environment, even for simple environments so the grid cannot be compressed as much. To illustrate these two components, three separate environments of increasing complexity where generated and a compressed occupancy grid was generated for each environment using four separate cell sizes (1.0 m, 0.5 m, 0.2 m, and 0.1 m) and varying amounts of compression. The first environment is a simple rectangular room and an occupancy grid for this environment generated using the full set of data with the smallest cell size is shown in Figure 19. The MSE as function of data stored for this environment is shown in Figure 20 for each of the four cell sizes. As expected the environment with the largest cell size, when fewer coefficients are needed to store all of the environment information, has the largest error when a small amount of data is used to represent the environment. It can also can be seen that, for each of the four cell sizes, as more information is stored, the error in the reconstructed form of the occupancy grid decreases.

A second environment with more complexity is shown in Figure 21 which is a rectangular room with complex corners. The MSE between the true occupancy grid and the reconstructed grid as a function of the amount of data stored is shown in Figure 22 for each of the cell sizes. The overall behavior of the error is the same as in the simple case with the largest cell size having the largest error when small amounts

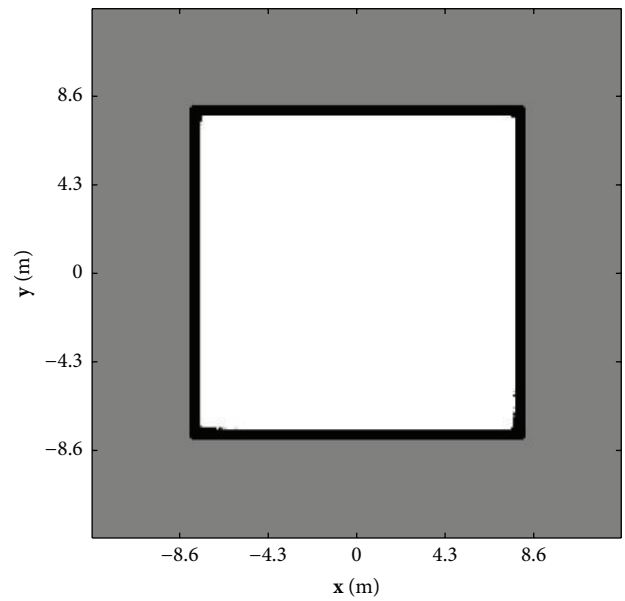


Figure 19: Occupancy grid of simple rectangular environment using 100% of the data and a cell size of 0.1 m.

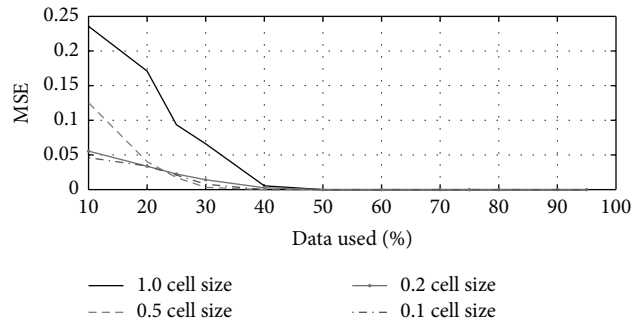


Figure 20: MSE between the true occupancy grid of the simple environment and the reconstructed form of the grid as a function of percent of data stored for each of the four cell sizes.

of data are used. However, because of the added complexity in this environment, the errors at the lower data percentages are larger than the simple case which is what we expected because more information is required to store the more complex environment.

Finally, a complex environment of a rectangular room with complex corners along with rectangular objects located throughout the rectangular room is shown in Figure 23. As with the previous two environments the MSE between the true occupancy grid and the reconstructed occupancy as a function of data is shown in Figure 24 for each cell size. As expected the use of larger cell sizes causes more errors in the reconstructed form of the grid when small amounts of data are used. Also, the additional complexity causes the errors at lower data percent to be larger than in the more simple environments. It can be seen from the presented results that there appears to be a "magic" compression level of 40% at which, no matter the complexity, the compressed occupancy can be reconstructed with significant accuracy.

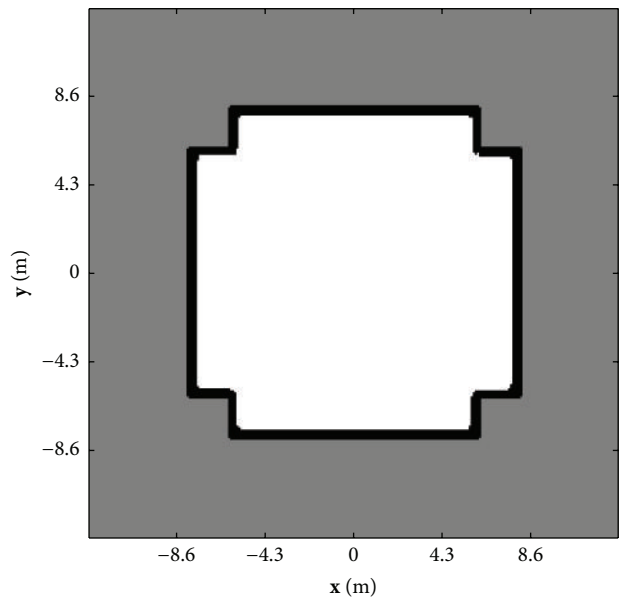


Figure 21: Occupancy grid of rectangular environment with complex corners using 100% of the data and a cell size of 0.1 m.

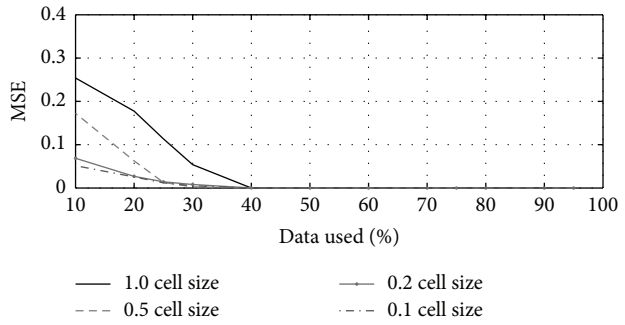


Figure 22: MSE between the true occupancy grid of the environment with medium complexity and the reconstructed form of the grid as a function of percent of data stored for each of the four cell sizes.

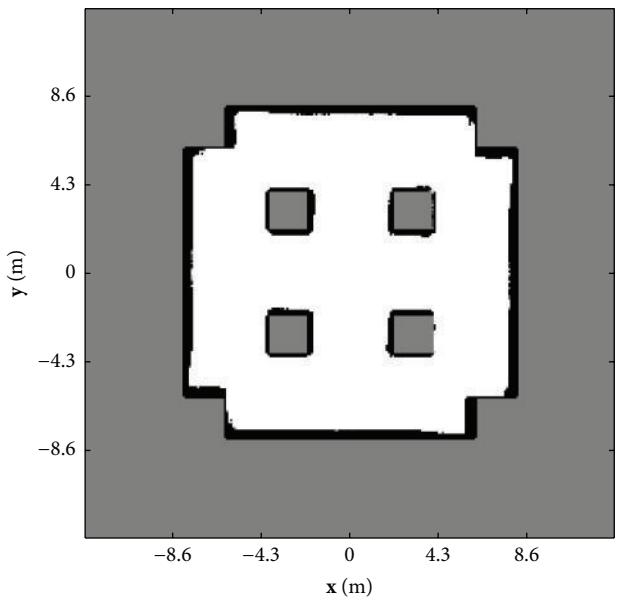


Figure 23: Occupancy grid of the complex environment using 100% of the data and a cell size of 0.1 m.

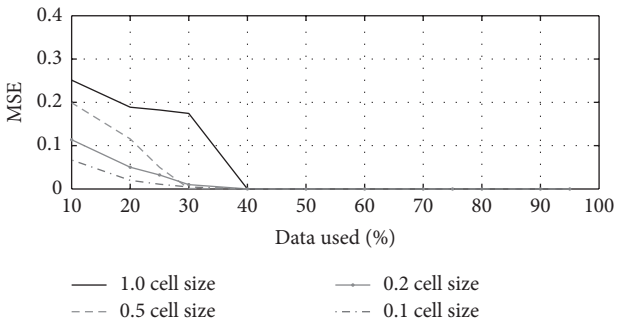


Figure 24: MSE between the true occupancy grid of the complex environment and the reconstructed form of the grid as a function of percent of data stored for each of the four cell sizes.

7. Conclusions

In this paper an approach for solving the SLAM problem was presented that makes use of compressed occupancy grids. The approach is based on an extension to the FastSLAM algorithm that represents an unknown environment by using occupancy grids. A modified form of the SLAM algorithm, FastSLAM COG, was presented in Section 3 which makes use of a generic compression and reconstruction method to solve the SLAM problem. Four specific reconstruction methods were examined in Section 4 in order to see how well each of the approaches could reconstruct a compressed signal along with an analysis of the time needed for each algorithm to complete. Based on the reconstruction time, two of the four algorithms were used to replace the generic reconstruction method in the FastSLAM COG algorithm and their performance was examined in Section 6 using experimental data. From the presented experimental results it was concluded that the FastSLAM COG algorithm using l_2 reconstruction approach could solve the SLAM problem with almost no errors compared to the FastSLAM OG algorithm while storing only 40% of the data required to store the complete occupancy grid. The experimental results presented in this paper were achieved by exploring how compressing the occupancy grid affects the performance of the algorithm in simple environments. In order to further examine our approach, a next step would be to examine how well this approach performs in more complex and unstructured environments. Also, further investigation into the effect of using more complex compression and reconstruction approaches as well as the possibility of dynamically resizing the cell size could lead to higher data compression rates without inducing errors into the SLAM solution.

Competing Interests

The authors declare that there are no competing interests regarding the publication of this paper.

Acknowledgments

This material is based upon work supported by (while serving at) the National Science Foundation of the United States.

References

[1] A. Elfes, *Occupancy grids: a probabilistic framework for robot perception and navigation [Ph.D. thesis]*, Carnegie Mellon University, Department of Electrical and Computer Engineering, Pittsburgh, Pa, USA, 1989.

[2] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: a factored solution to the simultaneous localization and mapping problem," in *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI '02) and the 14th Innovative Applications of Artificial Intelligence Conference (IAAI '02)*, pp. 593–598, Edmonton, Canada, August 2002.

[3] C. Radhakrishna Rao, "Information and accuracy attainable in the estimation of statistical parameters," *Bulletin of the Calcutta Mathematical Society*, vol. 37, no. 3, pp. 81–91, 1945.

[4] D. Blackwell, "Conditional expectation and unbiased sequential estimation," *The Annals of Mathematical Statistics*, vol. 18, pp. 105–110, 1947.

[5] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel approach to nonlinear/non-gaussian Bayesian state estimation," *IEE Proceedings Part F: Radar and Signal Processing*, vol. 140, no. 2, pp. 107–113, 1993.

[6] A. F. Smith and A. E. Gelfand, "Bayesian statistics without tears: a sampling-resampling perspective," *The American Statistician*, vol. 46, no. 2, pp. 84–88, 1992.

[7] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with Rao-Blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[8] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, MIT press, Cambridge, Mass, USA, 2006.

[9] R. Baraniuk, M. A. Davenport, M. F. Duarte, and C. Hegde, "An introduction to compressive sensing," *Connexions E-Textbook*, 2011.

[10] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Transactions on Computers*, vol. 23, no. 1, pp. 90–93, 1974.

[11] E. Moore, "On the reciprocal of the general algebraic matrix," *Bulletin of the American Mathematical Society*, no. 26, pp. 294–295, 1920.

[12] R. Penrose, "A generalized inverse for matrices," in *Mathematical proceedings of the Cambridge philosophical society*, vol. 51, no. 3, pp. 406–413, Cambridge University Press, 1955.

[13] E. J. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 489–509, 2006.

[14] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society, Series B: Methodological*, vol. 58, no. 1, pp. 267–288, 1996.

[15] S. Becker, J. Bobin, and E. J. Candès, "Nesta: a fast and accurate first-order method for sparse recovery," *SIAM Journal on Imaging Sciences*, vol. 4, no. 1, pp. 1–39, 2011.

[16] E. Candes and J. Romberg, "ℓ_1*-Magic : Recovery of Sparse Signals via Convex Programming*," vol. 4, 2005, http://users.ece.gatech.edu/justin/l1magic/downloads/l1magic.pdf.

[17] A. Carmi, P. Gurfil, and D. Kanevsky, "Methods for sparse signal recovery using Kalman filtering with embedded pseudo-measurement norms and quasi-norms," *IEEE Transactions on Signal Processing*, vol. 58, no. 4, pp. 2405–2409, 2010.

[18] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[19] S. J. Julier and J. LaViola Jr., "On Kalman filtering with nonlinear equality constraints," *IEEE Transactions on Signal Processing*, vol. 55, no. 6, pp. 2774–2784, 2007.

[20] R. Chartrand, "Exact reconstruction of sparse signals via nonconvex minimization," *IEEE Signal Processing Letters*, vol. 14, no. 10, pp. 707–710, 2007.

[21] R. G. Brown and P. Hwang, *Introduction to Random Signals and Applied Kalman Filtering: With MATLAB Exercises and Solutions*, John Wiley & Sons, New York, NY, USA, 3rd edition, 1997.

[22] T. H. Cormen, *Introduction to Algorithms*, McGraw-Hill Higher Education, 2001.

[23] G. Arfken, *Mathematical Methods for Physicists*, Academic Press, Orlando, Fla, USA, 3rd edition, 1985.

[24] M. H. Lee and M. Kaveh, "Fast hadamard transform based on a simple matrix factorization," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, no. 6, pp. 1666–1667, 1986.

[25] R. Coifman, F. Geshwind, and Y. Meyer, "Noiselets," *Applied and Computational Harmonic Analysis*, vol. 10, no. 1, pp. 27–44, 2001.

[26] MathWorks, "Moore-penrose pseudoinverse of matrix," 2014, http://www.mathworks.com/help/matlab/ref/pinv.html.

[27] L. N. Trefethen and D. Bau III, *Numerical Linear Algebra*, vol. 50, Society for Industrial and Applied Mathematics, 1997.

[28] O. Diegel, A. Badve, G. Bright, J. Potgieter, and S. Tlale, "Improved mecanum wheel design for omni-directional robots," in *Proceedings of the 2002 Australasian Conference on Robotics and Automation*, pp. 117–121, Auckland, New Zealand, 2002.

[29] C. H. Cain and A. Leonessa, "Testing vision-based sensors for enclosed underwater environments when applied to ekf slam," in *Proceedings of the 5th Annual Dynamic Systems (ASME '12) and Control Conference Joint with the 11th Motion and Vibration Conference (JSME '12)*, pp. 213–220, American Society of Mechanical Engineers, 2012.