RESEARCH ARTICLE

iSAM2 using CUR matrix decomposition for data compression and analysis

Wangseok Jang and Tae-wan Kim*

Department of Naval Architecture and Ocean Engineering, Research Institute of Marine Systems Engineering, Seoul National University, Seoul 08826, South Korea

*Corresponding author. E-mail: taewan@snu.ac.kr

Abstract

We introduce a factorization method to increase the calculation speed of incremental smoothing and mapping using Bayes tree (iSAM2), which is used in the back-end stage of simultaneous localization and mapping (SLAM), and to analyse the cause of the associated estimation error. iSAM2 is the method most commonly used to increase the accuracy of SLAM and shorten the calculation time required in real dense situations. In this paper, we describe the application of CUR matrix decomposition to iSAM2’s sparse linear system solver. CUR matrix decomposition is one of the low-rank matrix decomposition methods. It consists of matrices C and R, which are sets of columns and rows of the original matrix, and matrix U, which approximates the original matrix. Because of the characteristics of CUR matrix decomposition, it is possible to effectively approximate the sparse information matrix. Also, using principal component analysis, it is possible to identify the factors that increase or decrease the estimation error. We confirmed the feasibility of the proposed analysis method by applying it to real datasets and obtaining estimation errors similar to those obtained with iSAM2.

Keywords: simultaneous localization and mapping; CUR matrix decomposition; iSAM2

1. Introduction

1.1. Simultaneous localization and mapping

In robotics, simultaneous localization and mapping (SLAM) is a technology for simultaneously performing the localization of a robot and the mapping of the surrounding environment. Similar technologies that preceded SLAM solved the problem of locating a robot in a state in which the environment was configured (localization) or solved the problem of describing the environment when the robot’s location was accurately known (mapping). However, the operation of a robot, such as an autonomous vehicle or a small drone, in an unspecified environment necessitates simultaneously estimating the location of the robot and characterizing the surrounding environment.

SLAM performs localization and mapping by synthesizing the mechanical signals that the robot controls and sensors that detect the surrounding environment. This process requires a large amount of computation, and it is difficult to determine an exact solution. SLAM can be divided into two stages illustrated in Fig. 1. The first stage, called the front-end stage, involves abstracting the data received from the sensors in a form that can be used later for position estimation. In this stage, there is a feature extraction step in which the sensor data are simplified to improve the calculation speed and a data association step in which the relations between previously detected features and newly recognized features are calculated. The second stage, called the back-end stage, involves estimating the position of the robot based on the data abstracted in the front-end stage. The process of estimating the position of the robot involves identifying the location with the highest probability of being the robot's position, based on sensor data analysed through maximum a posteriori (MAP) estimation. The two methods most commonly used
in the SLAM back-end stage are EKF SLAM and FastSLAM. EKF SLAM utilizes an extended Kalman filter algorithm and treats the state of the robot and map as a Gaussian model. FastSLAM utilizes a particle filter algorithm, and unlike EKF SLAM, it does not assume a Gaussian model or perform model linearization, so it can be more readily applied to real-world cases.

1.2. iSAM2

In addition to the above algorithms, graph-based SLAM, which expresses real problems by constructing a graph model, has been used in recent research. When a graph model is used, the various types of raw data obtained with the various sensors of the robot are easily expressed as edges of the graph, and the positions of the robot are expressed as nodes of the graph. This method of representation has the advantage of making it simple to visualize real-world problems. Figure 2 represents the SLAM process in terms of the graph model.

When graph-based SLAM is used to derive a solution in the back-end stage, methods other than MAP estimation are used, e.g. pose graph optimization, factor graph optimization, smoothing, and mapping. The reason for this is that the classical SLAM methods have limitations. In some studies, the SLAM solution approach using EKF has resulted in insufficient solutions consistent with the nonlinear model. Changing the linearization method has not proven to be an efficient solution approach in the filter domain. Also, if the SLAM problem is complex and contains many states, as is often the case for real-world problems, the information matrix or covariance matrix, one of the key results of the filter, is densely expressed.

In contrast, the smoothing approach omits complicated covariance calculations in expressing the state of the robot, and by using the Jacobian of sensor measurements as the information matrix, the data structure remains sparse and the map representation becomes more accurate. iSAM2 in particular improves the calculation speed and estimation accuracy in the following ways. First, robot state optimization is performed using a simple linear system solver such as Cholesky or QR factorization to improve speed performance. Second, re-ordering methods are used to simplify the matrix structure, because there is no need to align the time sequence in constructing the matrix. Third, batch or incremental methods are applied, making the solution process more suitable for large-scale problems such as those encountered in real-world situations.
1.3. Related work

Lu and Milios (1997) presented the first smoothing approach to the SLAM problem. They formulated the estimation problem using constraints between graph nodes. At that time, solving linear systems by optimization processes was mostly done using matrix inversion methods (Gutmann & Nebel, 1997). However, matrix inversion is not a numerically stable algorithm. Alternative solution methods include relaxation (Duckett et al., 2002; Bosse et al., 2004; Thrun et al., 2005), gradient descent (Folkesson & Christensen, 2004, 2007), preconditioned conjugate gradient (Konolige, 2004; Dellaert et al., 2010), multilevel relaxation (Frese et al., 2005), and belief propagation (Ranganathan et al., 2007) methods.

Direct methods, such as QR and Cholesky matrix factorization, have the disadvantage of being computationally costly. According to Dellaert and Kaess (2006), who presented an efficient sparse factorization for SLAM, direct methods are widely used (Frese, 2006; Folkesson et al., 2007; Kaess et al., 2008; Mahon et al., 2008; Grisetti et al., 2010; Konolige et al., 2010; Strasdat et al., 2010). Square root SAM (Dellaert & Kaess, 2006) performs smoothing by Cholesky factorization of a complete, naturally sparse information matrix using the Levenberg–Marquardt algorithm.

Kaess et al. (2011) introduced an advanced algorithm named iSAM2 that offers several advantages. The iSAM2 algorithm achieves online bundle adjustment, at least for datasets of reasonable size. In incremental setting problems, iSAM2 has the advantage that both reordering and relinearization can be performed incrementally at every step. Note that iSAM2 does not simply apply existing methods, such as matrix factorization updates, but rather introduces a completely novel algorithm for solving sparse nonlinear least-squares problems that grow over time. In relative formulation problems, iSAM2 provides separation into sub-maps represented by different sub-trees; even though they are not completely separated, new measurements can change that topology at any time so that the complexity is not explicitly bounded. In this respect, iSAM2 is similar to Atlas (Bosse et al., 2004) and Tectonic SAM (Ni et al., 2007; Ni & Dellaert, 2010), which split the problem into sub-maps. Furthermore, iSAM2 focuses on making online recovery of globally metric maps more efficient using incremental update steps; however, it is more expensive than the batch update method for large loop closing cases. For this reason, some approximate methods, such as HOG-Man, are more efficient than iSAM2 for dense sequences. iSAM2 also represents problems using graphical models. However, iSAM2 solves the full SLAM problem and does not reject any information. Furthermore, the construction of the Bayes tree differs from the construction of a junction tree, which involves first forming a clique graph and then finding a spanning tree. The Bayes tree is based on a given variable ordering, similar to matrix factorization, although it gains some flexibility because the order of the sub-trees of a clique can be changed, in contrast to the fixed variable ordering of the square root information matrix. Also, iSAM2 formalizes this connection more comprehensively through the Bayes tree data structure.

Follow-up studies have improved the performance of the iSAM2 method. Hsiao and Kaess (2019) introduced multihypothesis iSAM2. In conventional iSAM2, one measurement (factor) represents a single hypothesis and is assumed to be a single Gaussian. However, in actual situations, expressing a measurement as a single Gaussian is a limitation. To overcome this shortcoming, iSAM2 was expanded by introducing a Gaussian mixture model rather than a single Gaussian. Shahir and Taghirad (2014) improved the optimization component of the iSAM2
algorithm, by proposing a double dogleg optimization that is an improved version of dogleg optimization and setting up Gould parameters suitable for the algorithm. Wang et al. (2018) modified the “bucket heap approximate minimum degree algorithm” for use in place of the “column approximate minimum degree algorithm,” a re-ordering method used in iSAM2 to solve the numerical error problem of Cholesky factorization. AprIiSAM is a proposed version of iSAM2 that uses this ordering algorithm.

Some studies have performed the registration of objects photographed using a method other than SLAM. Hasan and Ko (2016) and Lee et al. (2020) use a method of compressing and expressing an object through edge detection methods, and Lee (2020) was registered using the iterative closest point (ICP) method. This process is very similar to the whole SLAM process, and ICP can be replaced by the back-end SLAM, the subject of this study. In the study of Men and Pochiraju (2014), a mapping function similar to SLAM was performed. However, there is a disadvantage that an overlapping area is essential, and SLAM supplements this part with a location sensor other than the photographing sensor.

1.4. Contributions

The application of CUR matrix decomposition to iSAM2 offers the advantage of approximating the information matrix (or the measurement Jacobian matrix “A”) with a “low-dimensional model” rather than increasing the operation speed. This has the advantage of compressing and calculating the target matrix, in contrast to QR and Cholesky factorization, which utilize the existing information matrix as it is. This makes it possible to store an approximate information matrix at each time step that can be used later in the analysis of the iSAM2 results.

CUR matrix decomposition is one of the approximation methods that result in less error in compressing the original matrix. Therefore, it is considered to be a suitable method for approximating a sparse matrix as the current information matrix for iSAM2. Also, because CUR matrix decomposition involves sampling the columns or rows of the original matrix directly, it is easy to identify the factors that have major influences on the results. For these reasons, it is easy to compress data and analyze results by applying CUR matrix decomposition to iSAM2. Figure 4 illustrates details for applying CUR matrix decomposition to iSAM2. By substituting QR factorization to CUR matrix decomposition at the delta estimation step, we compress the original matrix and analyze results.

2. Main Algorithm of iSAM2

2.1. Add new measurements and new variables

iSAM2 is the second version of the iSAM algorithm, which incrementally estimates the position of a robot. The estimation process is repeated whenever a new measurement is added, and as a new measurement is added, the factor graph and the measurement Jacobian matrix “A” to be calculated are newly constructed. Figure 5 illustrates a variation of some of the most commonly used iSAM2 examples. As new measurements are added, a new pose \( x_3 \) and a new measured landmark \( l_2 \) are added to the factor graph as the nodes. At the same time, the measurements are added as the factors.

When a new measurement is added from the upper figure to the lower figure in Fig. 5, two rows and two columns are added to the measurement Jacobian matrix “A” because two factors and two nodes are added.

2.2. Linearize at the current estimate

The measurement Jacobian matrix “A” examined in Section 2.1 constitutes a linear system for estimating the value of \( \Theta \) to be obtained from iSAM2. \( \Theta \) is the variable of the SLAM problem; this paper contains robot poses and landmark positions. In this section, we explain the process of obtaining the elements of the corresponding matrix.
The basic formula for iSAM2 is expressed by MAP estimation as follows:

$$\theta^* = \arg \max_\theta F(\theta) = \arg \max_\theta \sum f_i(\theta_i).$$  \hfill (1)

where

- $F$: factors
- $\Theta$: variables
- $f_i$: one factor
- $\theta_i$: variables that correspond $f_i$.

In other words, the node value that minimizes the sum of the values of each factor is obtained. Assuming that the theoretical factor is a Gaussian measurement model, it is expressed as follows:

$$\theta^* = \arg \max_\theta P(X, L | Z) = \arg \min_\theta \{ - \log P(X, L | Z) \}. $$  \hfill (2)

where

- $X$: all robot poses
- $L$: all landmark positions
- $Z$: all measurements.

That is, it estimates poses of robots and landmarks with the highest probability estimated, based on measurements up to the present time. We transform the argmax problem into an argmin problem by taking the “−log” $P(X, L | Z)$ in the above optimization formula (2) is defined as a belief in the field of probabilistic robotics. We can formulate the belief using probabilities about measurements and landmarks.

$$P(X, L | Z) \propto P(X, L, Z) = P(x_0) \prod_{i=1}^M P(x_i | x_{i-1}, u_i) \times \prod_{k=1}^K P(z_k | x_k, l_k).$$  \hfill (3)

where

- $x_0$: initial robot poses
- $x_i$: robot poses
- $l_k$: landmark position
- $u_i$: control measurements
- $z_k$: actual observation measurements.

iSAM2, like most SLAMs, is assumed to be a Gaussian process, and the probability model is defined as a Gaussian distribution. Each probability model of belief can be summarized using the process model and measurement equation as follows:

$$P(x_i | x_{i-1}, u_i) \propto \exp \left( -\frac{1}{2} \| g_i (x_{i-1}, u_i) - x_i \|_{A_{i}}^2 \right)$$  \hfill (4)

$$P(z_k | x_k, l_k) \propto \exp \left( -\frac{1}{2} \| h_k (x_k, l_k) - z_k \|_{D_k}^2 \right)$$  \hfill (5)

$$\| \epsilon \|^2 \triangleq \epsilon^T \Sigma^{-1} \epsilon,$$  \hfill (6)

where

- $g_i$: control measurement model
- $h_k$: observation measurement model.

Because the coefficients in the exponential term of the PDF formula are all constants, they can be arranged in a proportional relationship. Also, because it is not a variable of the optimization problem, these terms can be ignored. The MAP estimation equation (2) summarized using equations (3)–(6) is as follows:

$$\theta^* = \arg \min_\theta \{ - \log P(X, L | Z) \}$$

$$= \arg \min_\theta \left\{ \sum_{i=1}^M \| g_i (x_{i-1}, u_i) - x_i \|_{A_{i}}^2 + \sum_{k=1}^K \| h_k (x_k, l_k) - z_k \|_{D_k}^2 \right\}.$$  \hfill (7)

In most real-world situations, the process model and measurement equation are nonlinear functions. In this case, the optimization uses a nonlinear optimization method such as the Gauss–Newton iteration method or the Levenberg–Marquardt algorithm. These methods derive a solution by linearizing the nonlinear function within a range within which the linear assumption is trusted until the solution converges. Function linearization is performed by utilizing up to the first term in the well-known Taylor series. After the linearization of two terms, they can be combined and expressed as a simple linear system, as shown below. The initial variable ($\theta^0$) is updated using delta (\Delta) of the linear system below ($\theta^1 = \theta^0 + \Delta$). This process is
repeated until the variables converge.

\[ \Delta^* = \arg\min_i \| \Lambda \Delta - b \|^2. \] (8)

where

\[ A : \text{measurement Jacobian matrix} \]
\[ \Delta : \text{updating value} \]
\[ b : \text{residual constants of linearization} \]

In the above equation for the example problem in Fig. 5, the measurement Jacobian matrix “A” can be expressed as follows. The size of the measurement Jacobian matrix “A” is (# of pose + # of measurement) × (# of pose + # of landmark). When constructing the actual matrix, the robot pose, landmark pose, and measurement dimension are also multiplied and expressed. The measurement Jacobian matrix “A” of the linear system arranged as above is not a square matrix, and of course, it is not a symmetric matrix. Also, for the example involving driving for a long time rather than the example illustrated in Fig. 5, you can see that the sparse matrix is constructed as shown in Fig. 6 below.

### 2.3. Eliminating a variable from the factor graph

The example in Table 1 illustrates the elimination of a variable. In the above algorithm, the factor in which the common variable exists in the factor graph is divided into a frontal variable (a common variable, \(x_i\)) and a separator (a remaining variable, \(s_j\)) and its conditional probability is expressed as \(P(\theta | s_j)\). For example, when \(l_1\), variable of Table 1 problem is eliminated, factors containing \(l_1\) are \(f_{l_1}(x_1, l_1)\) and \(f_{l_2}(x_2, l_1)\). In this case, \(l_1\) is a frontal variable and \(x_1, x_2\) are separators. In the Gaussian case, the corresponding process obtains the same result as the measurement Jacobian matrix obtained by QR factorization.

As a result of the eliminating algorithm, the formula to maximize this problem changes as follows:

\[ f(\theta) = \prod P(\theta | s_j) \] (9)

\[ f_{\text{joint}}(\Lambda_j, s_j) \propto \exp \left\{ -\frac{1}{2} (A_j \Lambda_j + A_s s_j - b)^2 \right\} \] (10)

\[ P(\Lambda_j | s_j) \propto \exp \left\{ -\frac{1}{2} (\Lambda_j + rs_j - d)^2 \right\} \] (11)

\[ f_{\text{new}}(s_j) \propto \exp \left\{ -\frac{1}{2} (s_j - b)^2 \right\}. \] (12)

where

\[ r \triangleq A_{\theta} A_{\theta}^t, \ d \triangleq A_{\theta} b, \ A_{\theta} \triangleq A_{\theta} - A_{l 1} r, \ b' \triangleq b - A_{l 1} d \]

That is, to maximize the productivity of all factors, \( \Delta_j + rs_j - d = 0 \) must be satisfied, so the delta to be obtained through the linearization process is as follows:

\[ \Delta_j = d - rs_j. \] (13)

Estimating \( \Delta_j \) through this process is performed faster than estimation using the factor graph. In Bayes net, a conditional is used to define the dependence between variables, and the \( \Delta_j \) is sequentially estimated. Table 2 shows an example of calculating the robot pose and landmark position using the factor graph and Bayes net of Table 1. The factor graph composes the entire measurement Jacobian matrix to obtain the \( \Delta_j \), so it takes a long time to calculate if the number of variables increases compared to Bayes net.

### 2.4. Creating the Bayes tree

iSAM2 uses a new data structure called the Bayes tree. The Bayes tree is created based on the Bayes net where elimination is performed, and when a new variable is added, the structure is updated. The Bayes tree has a structure similar to that of the clique tree, and in iSAM2, a clique is a unit of update calculation that performs delta calculation in the future. The clique is a bundle of the sub-Bayes net that variables depend on others. The clique corresponding to the variable that the new variable does not affect has no structural change even when a new variable is inserted, and delta update is not performed. The clique that the new variable affects is removed from the existing tree. The structure of the factor graph is updated by performing elimination and creating the Bayes tree again. Also, at this time, the delta is calculated to update the node.

For example, consider a problem such as that shown in the first progress of Fig. 7. In this problem, \( X_{22} \) is a newly added variable. In this case, only two cliques are affected by adding new variables, and they are updated. In some cases, as shown in the third progress of Fig. 7, there are cases in which a variable and an edge connecting the previous variable are added, and an extra edge is added. It is called the loop closure case. In this case, all variables that compose loop closure are updated, and the related cliques are reconstructed.
Table 1: Eliminating a variable from the factor graph.

| Factor graph | Graph structure | Factor | Conditional |
|--------------|----------------|--------|-------------|
| $l_1$        | $x_1$, $l_1$   | $f_1(x_1, l_1)$, $f_2(x_2, l_1)$, $f_3(x_3, l_2)$, $f_4(x_1)$, $f_5(x_1, x_2)$, $f_6(x_2, x_3)$ | - |
| $l_2$        | $x_1$, $x_2$, $x_3$ | $P(l_1 | x_1, x_2)$, $P(l_2 | x_1)$, $P(x_1 | x_2)$, $P(x_2 | x_3)$, $P(x_3)$ | |

Table 2: Comparison calculation procedure between factor graph and Bayes net.

| Procedure | Factor graph | Bayes net |
|-----------|-------------|-----------|
| $A = \begin{bmatrix} 1 & -1 \\ 1 & -1 \\ 1 & -1 \\ -1 & 1 \\ -1 & 1 \end{bmatrix}$ | $b = \begin{bmatrix} 150 \\ 50 \\ 50 \\ 0 \\ 100 \end{bmatrix}$ | $\Theta = \begin{bmatrix} 150 \end{bmatrix}$ |
| $x_1 = 0$ | $x_2 = 100$ | $x_3 = 200$ |

3. CUR Matrix Decomposition

CUR matrix decomposition is a decomposition method that is mainly used in big data analysis. It is a method known to yield good performance with sparse matrices in which the matrix elements are mostly zero. Compared to the most commonly known decomposition method, low-rank singular-value decomposition (SVD), the performance of CUR matrix decomposition is optimized for the worst sampling case. It has the advantage of finding a statistical meaning by constructing a matrix directly from the original data. CUR matrix decomposition approximates the original matrix as shown below.

$$A = CUR$$

Where $A \in \mathbb{R}^{m \times n}$, $C$ is the chosen set of $r$ columns of $A$ ($m \times k$), $R$ is the chosen set of $r$ rows of $A$ ($k \times n$), $U$ is a matrix that approximates the original matrix using matrices $C$ and $R$. As seen from the above formula, the results of CUR matrix decomposition may vary depending on the sampling rate. If the number of sampled columns/rows does not match the rank of the original matrix, an approximation is made instead of accurate decomposition. Therefore, the %-root-mean square distortion (PRD) and compression ratio (CR) have been proposed as indicators of the performance of CUR matrix decomposition. PRD is the degree of deformation between the original matrix and the approximate matrix, and the smaller its value is, the more closely it approximates the original matrix. CR is the ratio of the data size between the original matrix and the approximate matrix, and the larger the value is, the more the data are compressed.

$$\text{PRD} (\%) = \frac{||A - CUR||_F}{||A||_F}.$$  \hspace{1cm} (15)

Where $||A||_F = (\sum_{i,j} a_{ij}^2)^{1/2}$

$$\text{CR} = \frac{mn}{mc + cr + rn},$$  \hspace{1cm} (16)

Where $A \in \mathbb{R}^{m \times n}$, $C \in \mathbb{R}^{m \times c}$, $U \in \mathbb{R}^{c \times r}$, $R \in \mathbb{R}^{r \times n}$. 

Downloaded from https://academic.oup.com/jcde/article/8/3/855/6275213 by guest on 14 May 2021
CUR matrix decomposition has several approximations, depending on the column/row sampling method and the matrix $U$ composition method.

First, when column/row sampling is performed, the column/row leverage score is calculated, and sampling is performed based on the corresponding value. Either of two methods, the energy CUR matrix decomposition (E-CUR) or the SVD CUR matrix decomposition (SVD-CUR) method, is used to calculate the leverage score. E-CUR calculates the leverage score based on the norm of the column/row vector of the original matrix.

\[
p_i = \frac{\sum_{j=1}^{n} A(i, j)^2}{\sum_{i=1}^{m} \sum_{j=1}^{n} A(i, j)^2}
\]  

(17)

SVD-CUR performs SVD on the original matrix and calculates the leverage score based on the norm of the singular vector.

\[
p_j = \frac{1}{k} \sum_{\xi=1}^{k} (v_{j}^{\xi})^2.
\]  

(18)

where

\[v_{j}^{\xi}: j\text{th coordinate of the }\xi\text{th right singular vector}\]

\[A \approx \sum_{\xi=1}^{k} \langle \xi, \xi \rangle v_{j}^{\xi}.
\]

In general, SVD-CUR has the advantage of obtaining results similar to the original matrix with less sampling than E-CUR. Still, it has the disadvantage that it requires a more significant amount of computation because it performs SVD.

Four methods for constructing matrix $U$ have been identified in previous research. These methods are summarized in Table 3.

The results of testing these four algorithms on a matrix that is used in iSAM2 are as follows. Scaling is the main reason for
the differences in the results. The matrix used for iSAM2 is relatively small. For a large matrix, scaling is performed to improve the calculation speed. If scaling is applied when performing CUR matrix decomposition on a matrix of small size, the approximate accuracy is greatly reduced. In Table 4, four methods for constructing matrix U are tested in the iSAM2 delta estimation step and Aldroubi’s method (2019) is chosen.

### 4. CUR Matrix Decomposition in iSAM2 Method

The application of CUR matrix decomposition to iSAM2 offers the advantage of approximating the information matrix (or the measurement Jacobian matrix “A”) with a “low-dimensional model” rather than increasing the operation speed. This has the advantage of compressing and calculating the target matrix, in contrast to QR and Cholesky factorization, which utilize the existing information matrix as it is. This makes it possible to store an approximate information matrix at each time step that can be used later to analyse the iSAM2 results.

CUR matrix decomposition is one of the approximation methods that result in less error in compressing the original matrix, depending on the CR. Therefore, it is considered a suitable method for approximating a sparse matrix as the current information matrix for iSAM2. Also, because CUR matrix decomposition involves sampling the columns or rows of the original matrix directly, it is easy to identify the factors that have a significant influence on the results. For these reasons, it is easy to compress data and analyse results by applying CUR matrix decomposition to iSAM2.

As a result of the variable elimination algorithm in iSAM2, the delta obtained through the linearization process is as follows:

$$\Delta_{ij} = d - rs_{ij}.$$  \hspace{1cm} (19)

In iSAM2, the delta is calculated as above, and the pseudo-inverse matrix of the measurement Jacobian matrix is used. In other words, it is as follows:

$$Ax_{ij} = b - As_{ij}.$$  \hspace{1cm} (20)

In the above formula, \(A_s\) and \(s_j\) are the values given in the current step, and \(A_s s_j\) can be regarded as a constant vector. Therefore, the above equation can be expressed in a simple \(Ax = b\) form.

$$A_s \Delta_{ij} = c.$$  \hspace{1cm} (21)

where \(c = b - A_s s_j\).

Delta is calculated by applying CUR matrix decomposition to matrix \(A_s\). The delta calculation process using CUR matrix

| Author                  | Algorithm                                                                 |
|-------------------------|---------------------------------------------------------------------------|
| Drineas et al. (2006)   | 1. Construct matrix C by performing the “Column select” algorithm on matrix A. |
|                        | 2. Construct matrix R by performing the “Column select” algorithm on matrix \(A^T\). |
|                        | 3. Calculate matrix U by the following equation: \(U = C^T A R +\).           |
| Aldroubi et al. (2019)  | 1. Construct matrix C by performing the “Column select” algorithm on matrix A. |
|                        | 2. Construct matrix R by performing the “Column select” algorithm on matrix \(A^T\). |
|                        | 3. Construct matrix W using the intersection element of the sampled column/row. |
| Leskovec et al. (2014)  | 1. Select the column/row to sample using the Leverage score (using Frobenius norm) (sampling). |
|                        | 2. Divide each element of the sampled column/row by \(\sqrt{\# \text{ of sample} \times L.S.}\) (scaling). |
|                        | 3. Construct matrix W using the intersection element of the sampled column/row. At this time, the element is extracted from the original matrix A. |
|                        | 4. Matrix U is constructed using the results of the SVD of matrix W \(W_{\text{low}} = X \Sigma Y^T, U_{\text{low}} = Y(\Sigma_1)^2 X^T\). |
| Mahoney and Drineas (2009) | 1. Select the column/row to sample using the Leverage score (using Frobenius norm) (sampling). |
|                        | 2. Divide each element of the sampled column/row by \(\sqrt{\# \text{ of sample} \times L.S.}\) (scaling). |
|                        | 3. Perform SVD of \(C^T C\) using matrix C \(C^T C = Y \Sigma^2 Y^T\). Using this result, construct matrix \(\Phi\) \((\Phi = Y(\Sigma_1)^2 Y^T)\). |
|                        | 4. Intersection elements are extracted to construct matrix \(\Psi\). However, at this time, each element is created in matrix \(C\) in the same way as to how to construct matrix R from the original matrix A. |
|                        | 5. Matrix U is calculated as follows: \(U = \Phi \Psi^T\). |
decomposition is as follows:

\[ c = A_j \Delta_j \approx \text{CUR} \Delta_j \]

\[ \Delta_j \approx (\text{CUR})^\dagger c = R^\dagger U^\dagger C^\dagger c. \] (22)

5. Experimental Results

The results of applying CUR matrix decomposition in iSAM2 were compared with the original iSAM2 results obtained using QR factorization. The results of the SLAM application results were compared using the real-world datasets “mit-cscail.log” and “intel-lab.log,” obtained from https://github.com/tipaldi/friltlib/tree/master/data.

5.1. Accuracy

In the SLAM problem, the normalized \( \chi^2 \) error is used to evaluate the robot’s pose estimation accuracy. The original \( \chi^2 \) and its distribution are used as an index to compare the mean and variance of the assumed distribution with the mean and variance of the exact data. The \( \chi^2 \) value decreases as the hypothesized distribution is similar to that of the real data. In iSAM2, the predicted measurement (assumed average) from the robot’s pose is compared with the real measurement (average of actual data). The formula for calculating the normalized \( \chi^2 \) error by applying equation (23) to the iSAM2 problem is shown in equation (24) below.

\[
\text{Normalized } \chi^2 = \frac{1}{m-n} \sum_{i=1}^{m} \left( \frac{||h_i(\hat{\theta}_i) - z_i||^2}{\sigma_i^2} \right). \] (24)

\( m \): number of measurements
\( n \): number of degrees of freedom (\( n = 6 \) in 3D)

The reason why the pose estimation accuracy is evaluated using the \( \chi^2 \) error in the SLAM problem is that it is not possible to record the exact robot poses at every time step. Therefore, in most cases, pose estimation is performed by satisfying a factor that is a relative pose transformation between the robot poses generated by synthesizing a pose estimation sensor such as an IMU and a surrounding environment observation sensor such as LiDAR and camera. That is, equation (24) is used as an index that satisfies equation (1).

The first example is executing iSAM2 using mit-cscail.log as an input value applying CUR matrix decomposition and QR decomposition. The total variable vertices (the robot’s state) are 1051. Figure 8 is the result of calculating the normalized \( \chi^2 \) error at time intervals. In Fig. 8, the QR decomposition is symbolized by blue, the CUR matrix decomposition of 5/6 sampling is symbolized by red, 3/6 sampling is symbolized by green, and 1/6 sampling is symbolized by black lines. The error of all methods increased abnormally at time interval 168. It is because the wrong initial robot pose was estimated from the results of the front-end stage. Overall, when comparing the error of the QR decomposition and the 5/6 CUR matrix decomposition, the error of the 5/6 CUR matrix decomposition is relatively low because of the effect of ignoring noise for measurement. Also, the interval between
500 and 850 increases gradually and then decreases rapidly at a specific time step because the loop closure is formed at that time interval and all robot poses in the loop are re-estimated. Conversely, the error of the 3/6 CUR matrix decomposition and the 1/6 CUR matrix decomposition in the corresponding interval gradually increases and the width of the decrease is small because the loop closure is not formed due to failure to sample the relation factor forming the loop closure during the sampling process. On the other hand, the CUR matrix decomposition with a low sampling rate has an advantage for increased less error in the pose tracking section.

The second example is executing iSAM2 using intel-lab.log as an input value applying CUR matrix decomposition and QR decomposition. The total variable vertices (robot state) are 2762. In Fig. 9, the QR decomposition is symbolized by blue, the 5/6 sampling CUR matrix decomposition is symbolized by red, and the 3/6 sampling is symbolized by green lines. Since the intel-lab.log data are an example of a large number of small-scale loop closures, it was confirmed that the robot attitude estimation error rapidly increased by decomposing the CUR matrix with a small sampling. Since this dataset contains many loop closures, the results of the pose estimation using the 1/6 sampled CUR matrix decomposition example were not compared in this example. We discuss in the previous example that a low sampling rate cannot form the loop closure. It is verified by the 3/6 CUR matrix decomposition of this example.

Figure 10 shows the robot trajectory using x and y coordinates on a 2D plane among the robot’s poses in the two examples. As previously analysed through the normalized $\chi^2$ error, the path of the CUR matrix decomposition with a low sampling rate does not form loop closures so that the pose tracking is performed while maintaining the unreduced error.

5.2. Data compression and timing

The first data compression index is a viewpoint of matrix operation. In other words, it is the compression rate of the full matrix. The CR, an index of data compression rate proposed in previous studies using CUR matrix decomposition, was modified and used to compare with QR decomposition. The data compression rate in this way is as Table 5.

$$CR_{QR} = \frac{mn + nn}{mc + cr + rn}$$

where $A \in R^{m \times n}$, $C \in R^{m \times c}$, $U \in R^{c \times r}$, and $R \in R^{r \times n}$.

The second data compression index is a viewpoint of matrix store the delta estimation linear system collected in real time for data analysis. Since the data analysis presented in this paper uses the measurement Jacobian matrix, the data compression rate is compared for the measurement Jacobian matrix. In the case of compressed data, a general sparse matrix storage method is used. Since most of the matrix elements are 0, the sparse matrix stores nonzero elements and indexes of row and column of the elements. The data compression rate in this way is as Table 6. The CUR matrix decomposition is compressed about 0.01 times compared to the QR decomposition.

The computation time improvement applying CUR matrix decomposition is compared with the QR decomposition for two examples in Section 5.1. For all scenarios, the average operation time is measured by repeating 10 times under the
same conditions. The QR decomposition measures the elimination process and the computation time of the delta estimation process at each time interval. The CUR matrix decomposition measures the computation time of the CUR matrix decomposition and the delta estimation process at each time interval. The computation times for a mit-cacsil.log dataset compare with different methods are presented in figure 11 and quantitative values are presented in the table 7. These results for intel-lab.log dataset are presented in figure 12 and table 8.

The computation times are measured using cumulative time in the iSAM2 paper. However, when the calculation time is measured and when the sampling rate is low, the relation factor is ignored. Accordingly, loop closure is not formed, and thus the number of delta estimation calculations in the specific time interval decreases. Therefore, the normalized cumulative time is obtained by normalizing the cumulative time with the number of delta estimation calculations. The ratio of the cumulative calculation time to the QR decomposition is presented and compared.
As a result of the comparison, CUR matrix decomposition took up to about 0.29 times the computation time compared to QR decomposition. The reason that the operation time does not differ significantly compared to the CR values of the data is that the SVD algorithm used when constructing the W matrix of CUR matrix decomposition and solving the linear system. It takes longer time than the QR decomposition algorithm.

5.3. Data analysis

Another advantage of using CUR matrix decomposition in ISAM2 is analysing data about pose estimation accuracy. In this section, we describe how we analyse a result of ISAM2 applying CUR matrix decomposition in situations in which the normalized χ² error increases or decreases rapidly. Figure 13 shows the normalized χ² error at each step for the “mit-cscail.log” data when CUR matrix decomposition is applied with the 5/6 sampling rate. The results show that the error varies greatly in steps 860–862. We analyse data in steps 860–862 in this section.

In Section 2.2, each row of the measurement Jacobian matrix A is the result of linearizing each factor. So the measurement Jacobian matrix is the target of error analysis. Also, as an additional preprocessing step for the analysis, the normalization of the measurement Jacobian matrix A is performed in the form of \( A'x = 1 \) by dividing both sides by vector \( b \) of the linear system \( Ax = b \). Since the CUR matrix decomposition applies to solve the linear system, the vector \( b \) of the linear system is considered necessary.

Factors that are primarily affected are assessed by comparing the leverage scores of matrix elements corresponding to each factor. The leverage score is a value that assesses the importance of the column select algorithm. Therefore, we tried to assess the importance of each factor’s degree of freedom by using the leverage score. When the leverage score is calculated for the entire measurement Jacobian matrix for each degree of freedom, as shown in Table 9 below (sampling score, SS). These factors are summed and defined as the sum of the sampling score (SSS).

Table 10 is comparing the normalized \( \chi^2 \) error calculated for 907 factors in the 861st time step. The left is the sorted SSSs calculated in Table 9, and the right is the sorting the amount of change in the normalized \( \chi^2 \) error for each factor in the corresponding time step in order of size. Since there is no change in click at this time step, the change amount is 0 because the factor from 0 to 120 is not updated. The factors with the largest amount of error updates are the 903–906th factor (index 904–907). The analysis using the CUR matrix decomposition shows that the SSS of this factor is also high.

Table 7: Operation time comparison between QR factorization and CUR matrix decomposition for a mit-cscail.log dataset.

|               | QR          | 5/6 CUR     | 3/6 CUR     | 1/6 CUR     |
|---------------|-------------|-------------|-------------|-------------|
| Cumulative time (clock) | 99,718,608  | 80,069,318  | 9,284,757   | 1,408,121   |
| Norm. cumulative time (clock/cal. count) | 4,809,956   | 3,862,167   | 2,239,264   | 1,408,121   |
| Norm. cumulative time ratio | 1.00        | 0.80        | 0.47        | 0.29        |
Table 8: Operation time comparison between QR factorization and CUR matrix decomposition for an intel-lab.log dataset.

|                  | QR              | 5/6 CUR         | 3/6 CUR         |
|------------------|-----------------|-----------------|-----------------|
| Cumulative time (clock)    | 882,869,224     | 728,720,435     | 132,609,548     |
| Norm. cumulative time (clock/cal. count) | 322,244,655 | 265,980,803 | 132,609,548 |
| Norm. cumulative time ratio | 1.00        | 0.83           | 0.41           |

Figure 13: Normalized $\chi^2$ error for a mit-cacasil.log using 5/6 sampling CUR matrix decomposition.

Table 9: Summation of SS for each factor.

| Factor index | SS for each D.O.F. | SSS for each factor |
|--------------|--------------------|--------------------|
| 1            | 66                 | 666                |
|              | 294                |                    |
|              | 306                |                    |
| 2            | 36                 | 342                |
|              | 162                |                    |
|              | 144                |                    |
| 3            | 167                | 514                |
|              | 150                |                    |
|              | 197                |                    |
| ...          | ...                | ...                |

Table 10: Normalized $\chi^2$ errors compared with CUR matrix decomposition results.

| Factor index | SSS for each factor | Factor index | NCE difference |
|--------------|--------------------|--------------|----------------|
| 142          | 13 751             | 899          | 1.3893E-12     |
| 120          | 13 766             | 900          | 1.6862E-12     |
| 125          | 13 781             | 901          | 1.9156E-12     |
| 295          | 13 786             | 902          | 1.981E-12      |
| 140          | 13 895             | 903          | 1.9236E-12     |
| 904          | 13 906             | 904          | 1.4015E-10     |
| 905          | 13 928             | 905          | 2.5334E-07     |
| 906          | 14 747             | 906          | 4.3052E-07     |
| 907          | 15 974             | 907          | 4.3052E-07     |

6. Conclusion

We have introduced a CUR matrix decomposition method to solve a sparse linear system in iSAM2. This method helps to analyse the estimation error and to increase the calculation speed for the back-end stage in the SLAM process. Through testing with real datasets, we confirmed that there is no significant difference in accuracy for estimation in the appropriate sampling rate.

Our new algorithm employing CUR matrix decomposition is an approximation method, so it differs from the existing factorization method. Therefore, the sampling rate, a parameter of approximation, should be adjusted according to the purpose of the situation. If there are many loop closures, it is appropriate to apply a high sampling rate. If the user focuses on improving the computation time and data compression, it is appropriate to apply a low sampling rate.

In the future, we have a plan to investigate the more advanced approximation to apply in iSAM2. A new method focuses on reducing the difference in accuracy from the factorization method and improving the computation time. Also, we are
planning to apply an algorithm for determining hyperparameters such as CR.

Acknowledgments

This research is supported by research funding from three organizations: (i) the National Research Foundation of Korea funds through a project titled “A study on reinforced deep learning for multichannel monitoring data based total fault prediction of manufacturing process”; (ii) the Korean Ministry of Oceans and Fisheries funds through a project titled “Development of Autonomous Navigation System with Intelligent Route Planning Function”; and (iii) the Research Institute of Marine Systems Engineering funds.

We acknowledge and appreciate the support provided for our research by these organizations.

We are grateful for the support on SLAM from prof. John J. Leonard at MIT and on CUR matrix decomposition from prof. Gilbert Strang at MIT.

Conflict of interest statement

None declared.

References

Aldroubi, A., Hamm, K., Koku, A. B., & Sekmen, A. (2019). CUR decompositions, similarity matrices, and subspace clustering. Frontiers in Applied Mathematics and Statistics, 4, 65–81.

Bassi, F., Chao, L., Iwaza, L., & Kieffer, M. (2012). Compressivelin.

Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Duckett, T., Marsland, S., & Shapiro, J. (2002). Fast, on-line learning of globally consistent maps. Autonomous Robots, 21, 103–122.

Grissetti, G., Kuemmerle, R., Stachniss, C., Frese, U., & Hertzberg, C. (2010). Hierarchical optimization on manifolds for online 2D and 3D mapping. In IEEE International Conference on Robotics and Automation (ICRA)(pp. 273–278).

Gutmann, J. S., & Nebel, B. (1997). Navigation mobilier Roboter mit Laserscans. In Autonome Mobile Systeme 1997. Informatik aktuell. Springer.

Hasan, S., & Ko, K. (2016). Depth edge detection by image-based smoothing and morphological operations. Journal of Computational Design and Engineering, 3(3), 191–197.

Hsiao, M., & Kaess, M. (2019). MH-iSAM2: Multi-hypothesis ISAM using Bayes tree and hypo-tree. In International Conference on Robotics and Automation (ICRA)(pp. 1274–1280).

Kaess, M., Ranganathan, A., & Dellaert, F. (2008). iSAM: Incremental smoothing and mapping. IEEE Transactions on Robotics, 24, 1365–1378.

Kaess, M., Johannsson, H., Roberts, R., Ilia, V., Leonard, J. J., & Dellaert, F. (2011). iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering. In IEEE International Conference on Robotics and Automation(pp. 3281–3288).

Konolige, K. (2004). Large-scale map-making. In Proceedings of the 21st AAAI National Conference on AI(pp. 457–463).

Konolige, K., Grisetti, G., Kuemmerle, R., Burgard, W., Benson, L., & Vincent, R. (2010). Efficient sparse pose adjustment for 2D mapping. In 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(pp. 22–29).

Leskovec, J., Rajaraman, A., & Ullman, J. (2014). Mining of massive datasets. Cambridge University Press.

Lee, W., Lee, K., Lee, J., & Nam, B. (2020). Registration method for maintenance-work support based on augmented-reality-model generation from drawing data. Journal of Computational Design and Engineering, 7(6), 775–787.

Lu, F., & Milios, E. (1997). Globally consistent range scan alignment for environment mapping. Autonomous Robots, 4, 333–349.

Mahon, I., Williams, S., Pizarro, O., & Johnson-Roberson, M. (2008). Efficient view-based SLAM using visual loop closures. IEEE Transactions on Robotics, 24, 1002–1014.

Mahoney, M. W., & Drineas, P. (2009). CUR matrix decompositions for improved data analysis. Proceedings of the National Academy of Sciences, 106(3), 697–702.

Men, H., & Pochiraju, K. (2014). Hue-assisted automatic registration of color point clouds. Journal of Computational Design and Engineering, 1(4), 223–232.

Ni, K., Steedly, D., & Dellaert, F. (2007). Tectonic SAM: Exact; out-of-core, submap-based SLAM. In IEEE International Conference on Robotics and Automation (ICRA)(pp. 1678–1685).

Ni, K., & Dellaert, F. (2010). Multi-level submap based SLAM using nested dissection. In 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(pp. 2558–2565).

Shahir, R. T., & Taghirad, H. D. (2014). An improved optimization method for iSAM2. In 2014 Second RSI/ISM International Conference on Robotics and Mechatronics (ICRoM)(pp. 582–587).

Folkesson, J., Leonard, J., Leederkerken, J., & Williams, R. (2007). Feature tracking for underwater navigation using sonar. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(pp. 3678–3684).

Frese, U., Larsson, P., & Duckett, T. (2005). A multilevel relaxation algorithm for simultaneous localisation and mapping. IEEE Transactions on Robotics, 21, 196–207.
Ranganathan, A., Kaess, M., & Dellaert, F. (2007). Loopy SAM. In International Joint Conference on AI (IJCAI) (pp. 2191–2196).

Strasdat, H., Montiel, J., & Davison, A. (2010). Real-time monocular SLAM: Why filter? In 2010 IEEE International Conference on Robotics and Automation (ICRA) (pp. 2657–2664).

Taniguchi, A., WanPeng, L., Taniguchi, T., Takano, T., Hagiwara, Y., & Yano, S. (2017). Simultaneous localization, mapping and self-body shape estimation by a mobile robot. Advances in Intelligent Systems and Computing, 531, 53–68.

Thrun, S., Burgard, W., & Fox, D. (2005). Probabilistic robotics. The MIT Press.

Wang, X., Marcotte, R., Ferrer, G., & Olson, E. (2018). ApríISAM: Real-time smoothing and mapping. In 2018 IEEE International Conference on Robotics and Automation (ICRA) (pp. 2486–2493).