6 DOF EKF SLAM in Underwater Environments

MARKUS SOLBACH
Universitat de les Illes Balears

Abstract. The increasing number of industrial or scientific applications of Autonomous Underwater Vehicles (AUV) raises the challenging question on how to derive the vehicle's localization accurate enough for the mission success.

This paper details an approach to accurate localization based on EKF (Extended Kalman Filtering) SLAM (Simultanously Localization and Mapping) with pure 3D stereo data, which consists of three major stages.

Stage one is, in terms of EKF, the so called *prediction stage*. During this stage the algorithm predicts the vehicle's localization using the visual odometry, which is known to be noisy and to provide drift in position and orientation (pose). The uncertainty of the odometry data is modeled with the covariance matrix.

Stage two is the *state augmentation step*. In this phase, the current odometry estimation is added at the end of the state vector of the EKF. The uncertainty accumulated over time makes the resulting predicted state non reliable.

The last Stage (*update*) tries to reduce this error by finding visual *Loop Closings*. Loop Closings are areas of the environment which the robot already observed in the past. Loop Closings are important because they provide the system with new and often more reliable information, what is a second transformation of an already observed one. With the difference of these two transformations the approach is able to update the whole state vector to a one with less error by using *Extended Kalman Filtering* equations.

During the three steps of the filter, all the data concerning the robot pose (odometry and filter estimation) are expressed as (x, y, z) for translation and a quaternion (q_w, q_1, q_2, q_3) for orientation.

In this work, a pure *stereo system* is used to compute the visual odometry in 3D and to find the visual loop closings. A Kalman update is performed if the algorithm is able to find a Loop Closing between an image associated to a position stored in the state vector with the current image, that is, if both images present a certain level of overlapping.

To calculate the motion of the camera between two positions which are suspected to be a loop closing, first SIFT [Lowe 2004] features or SURF [Bay et al. 2008] features of both images are computed. Then, applying the principle of the stereoscopy, the 3D points corresponding to the image features matched between the current stereo pair are calculated. Afterwards, the *Perspective N-Point* (PNP) is solved between the 3D points computed from the current stereo pair and the 2D features of the candidate image to close a loop. The transformation between both views is computed by minimizing the error of reprojecting all the 3d points onto the 2d features of the candidate image. The difference between this transformation and the transformation obtained by composing the successive positions between both views stored in the state vector is fundamental to correct the whole state vector.

Thanks to the robust Loop Closing detection, as shown in the experiments, the presented approach provides an improvement of the vehicle's localization.

Additional Key Words and Phrases: Image Registration, Visual Navigation, Underwater Robots, EKF, Visual SLAM, 6 DOF

1. INTRODUCTION

1.1 Problem Statement

In the last years, technological advances made easier the accessibility of the sub-aquatic world for research, exploration and industry exploitation. Nowadays *Remotely Operated Vehicles* (ROVs) are used in a wide range of applications, such as maintenance, rescue operations, surveying, infrastructure inspection and sampling. Some of ROVs limitations, such as limited operative range and the need of support vessels, are overcome by *Autonomous Underwater Vehicles* (AUVs). These kinds of vehicles are used in highly repetitive, long or hazardous missions. Moreover, since they are untethered and self-powered, they are also significantly independent from support ships and weather conditions. This, in comparison to ROVs, can reduce considerably the missions costs, human resources and execution time.

One of the most challenging points in research associated to underwater vehicles is the one of localization. There are several possibilities to estimate the vehicle's pose, for instance, using inertial sensors, or by computing the odometry with acoustic sensors or cameras. Another possibility is sensor fusion, which means combining inertial sensors and odometers, in *extended Kalman filtering* (EKF) or *particle filters*, to correct errors within the trajectory [Lee et al. 2004], [Kinsey et al. 2006].

The most successful approach to perform a reliable localization is called *SLAM* [Durrant-Whyte and Bailey 2006]. SLAM (*Simultaneous Localization And Mapping*) computes the position of the vehicle simultaneously to the calculation and refinement of the position of landmarks of the environment.

In the past, underwater SLAM was mainly developed by using acoustic sensors. These sensors provide good underwater properties, such as large sensing ranges[Ribas et al. 2007]. The problem with acoustic sensors is the spatial and temporal resolution, which is lower than using cameras. This higher resolution permits cameras to provide more environmental data than the data provided by a sonar.

But using cameras also has some disadvantages, which are briefly stated next. A video camera system, as mounted on an AUV, is dependent on light and visibility. Poor illumination conditions or turbid water (particles in the water) make the information obtained by the camera corrupt. Only the application of certain filtering techniques can solve this problem.

However, the higher spatial and temporal resolution of cameras, the usage of quaternions to avoid singularities and the existence of few solutions of pose-based ekf stereo slam [Eustice et al. 2008], are the main reasons to justify this visual approach. Another reason is to have the possibility to compare 3D EKF SLAM with Graph SLAM, [Thrun 2006] or [Olson et al. 2006].

Accordingly, this paper proposes a vision based approach with a stereo camera system to perform underwater pose based visual EKF-SLAM. The first approximation to estimate the pose of an underwater vehicle, using a camera system, is using the visual odometry. However, odometric methods are prone to drift and it is necessary to periodically correct this estimation. Extended Kalman Filtering is a technique extendedly used to perform such a correction.

Future states of the vehicle are predicted by modeling properly the vehicle motion, and predictions are refined and corrected by using detected visual loop closings as a set of environmental measurements. As shown in the approach of [Burguera et al. 2014] EKF is a good choice to perform visual SLAM.

1.2 Related Work

In natural sub-aquatic scenarios visual SLAM has to deal with difficulties not present onshore. For instance low light conditions, flickering, scattering, particles, and the difficult task to find reliable and robust features in a non-man made environment.
The key to perform underwater visual EKF-SLAM is to detect reliable loop closings. They have to be detected robustly under different influences, such as changing light conditions and variation of the viewpoint. In the context of visual SLAM, this procedure is called *Image Registration* and the task is to recognize scenes the robot already observed in the past, by finding images that have certain overlapping, and moreover to calculate the relative camera displacement between both viewpoints.
Literature about stereo SLAM solutions for underwater robotic systems is scarce. The available literature deals mainly with EKF-SLAM [Matsebe et al. 2008] by correcting the odometry with the results of the image registration. Such systems include newly observed landmarks into the state vector. The advantage of such systems is the continuous correction of the robot pose and landmarks in the whole state vector. The disadvantage is the increasing complexity as the state vector gets bigger. After a certain amount of iterations an on-line usage is no longer possible.
In [Schattschneider et al. 2011] the system set-up was quite similar. In this work, a stereo camera system was used for ship hull inspection. 3D landmarks obtained by the stereo camera were used to detect loop closings, besides the state vector contains the vehicle poses and the landmarks, [Matsebe et al. 2008].
Another approach for underwater SLAM is presented in [Salvi et al. 2008]. The authors build the state vector by using the pose and the velocity of the vehicle using a *Doppler Velocity Log* (DVL), and the 3D pose is computed by the installed stereo image system. Furthermore, the filter update is performed by using image registration and comparing all 3D landmarks stored in the state vector with new ones.
Providing the current vehicle pose, its linear velocity, acceleration and the angular rate to the state vector is proposed in [Eustice et al. 2008]. The landmarks are not saved in the state vector, what decreases the computational resources in comparison to other approaches. But the usage of image registration at every iteration to update the state vector still has a high computational cost and takes the major running time.
A different point of view to solve the localization problem is to use graph-optimization or bundle adjustment. In these approaches each vehicle pose and sometimes the position of the landmarks are added as a node to a graph. Subsequent nodes are linked by edges, which usually represent the distance between consecutive poses. Loop closings generate additional nodes and edges, apart from those obtained by the visual odometry. After a new loop closing is added to the graph, global optimization of the whole graph is computed by applying Levenberg-Marquardt algorithm, to improve the distance between all nodes by minimizing the quadratic error [Beall et al. 2011]. SLAM using graph-optimization is also known as *Graph-SLAM*.
A benefit of Graph-SLAM is the lack of linearization errors, as given by EKF-SLAM. But on the other hand, graphs grow hugely with the trajectory what effects the computational resources dramatically.

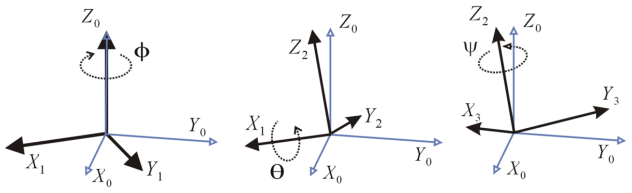

Fig. 1: 3D Transformation. Source: "EulerG" by DF Malan - Own work. Licensed under Public domain via Wikimedia Commons (21.07.2014)

This study presents a vision based approach to stereo EKF pose-based slam for AUVs, with the vehicle orientation represented as a quaternion. Explained in more detail in chapter 4, the EKF estimates continuously the pose of the vehicle by applying three steps. The first two steps are the prediction and the state augmentation steps. The prediction step predicts the vehicles future pose by composing the current vehicle pose with the current odometry. The state vector, which stores all vehicle poses, is augmented by the current prediction in the state augmentation step. In the third step all predictions are corrected in the update phase of the Kalman filter using as observations the detected loop closings [Schattschneider et al. 2011].
The prediction of the vehicle motion is obtained from a stereo visual odometer and predictions are updated using the transformations obtained from a set of visual loop closings computed using an approximation of the PNP problem from 3D to 2D.
The paper is structured as follows: the next Section presents the necessary mathematical background of 3D transformations to estimate robot movements in 3D; Section 3 explains the image registration to detect loop closings; Section 4 explains the design and the structure of Extended Kalman Filtering to perform SLAM; Section 5 discusses the results obtained by a real underwater dataset recorded in a pool located in the University of the Balearic Islands (UIB); Section 6 concludes the paper and gives some outlines of the forthcoming work.

2. 3D TRANSFORMATIONS

One of the key targets of this work is to model the classical transformations, *composition* ($\oplus$) and *inversion* ($\ominus$), for robots with 6 DOF and derive the Jacobian matrices of each transformation.
Both operations define a transformation in translation and rotation. The $\oplus$ permits to add a transformation to a current pose , and $\ominus$ permits to invert a current pose transformation. Defining both operations in pure 3D will permit us to predict the vehicle motion model and to run the Kalman updates [Matsebe et al. 2008], [Smith et al. 1988].
This approach is close to [Burguera et al. 2014], but with a major difference. Since [Burguera et al. 2014] uses 2.5D, what means 2D given by the dead-reckoning for x, y and z is given by an altimeter, in this paper the proposal will be a full 3D Transformation model, which takes all its information from the stereo camera system. The basic 3D Transformation assumes 3 degrees of freedom (DOF) for translation, x, y, z and another 3 DOF for rotation ϕ, θ, ψ (roll, pitch, yaw). The rotation is illustrated in Figure 1 and it can be seen easily, that it is not commutative.

2.1 Composition

Adding a relative displacement, for instance given by odometry (observation), to an absolute pose can be done using the *Transformation Composition*. Where the relative displacement describes the relative motion from the previous state to the current, whereas the absolute pose state describes the absolute motion from the origin to the current pose.
The presented approach corresponds to a full three dimensional composition and it is different than [Burguera et al. 2014], where the depth was calculated externally.
In more detail: Let X be the absolute pose state and Y the newly observed, relative transformation, both defined as shown in Equation 1.

$$X = \begin{bmatrix} x^X \\ y^X \\ z^X \\ q_w^X \\ q_1^X \\ q_2^X \\ q_3^X \end{bmatrix}, \qquad Y = \begin{bmatrix} x^Y \\ y^Y \\ z^Y \\ q_w^Y \\ q_1^Y \\ q_2^Y \\ q_3^Y \end{bmatrix} \tag{1}$$

A pose state, for instance X, consists of seven variables. The first three to express the position in x^X, y^X, z^X the last four express the orientation in 3D in form as a unit-quaternion with $\hat{q} = [q_w^X, q_1^X, q_2^X, q_3^X]$.
Quaternions were conceived by Sir William Rowan Hamilton in 1843 and they provide a strong algebra for rotation [Vince 2011]. Advantages of using quaternions instead of Euler angles are the lack of gimbal lock and the ability of good interpolation [Vince 2011]. Furthermore, from the implementation point of view they are more space efficient: a quaternion can be stored in four float values, whereas a classical rotation matrix needs at least nine. For further reading on quaternions refer to: [Opower 2002] and [Kuipers 2002].
To cover the necessary algebra of quaternions needed to understand this work, I will give a short overview of the used operations. First the multiplication of quaternions will be described. This operation is important when angles of two quaternions are accumulated. Although the angles are summed up, the quaternions have to be multiplied. Let $r = [r_w, r_1, r_2, r_3]^T$ be the first and $s = [s_w, s_1, s_2, s_3]^T$ the second quaternion to be multiplied. Equation 2 shows Multiplication of r with s, with its result $t = [t_w, t_1, t_2, t_3]^T$.

$$t = \begin{bmatrix} t_w \\ t_1 \\ t_2 \\ t_3 \end{bmatrix} = \begin{bmatrix} r_w \cdot s_w - r_1 \cdot s_1 - r_2 \cdot s_2 - r_3 \cdot s_3 \\ r_w \cdot s_1 + r_1 \cdot s_0 - r_2 \cdot s_2 + r_3 \cdot s_2 \\ r_w \cdot s_2 + r_1 \cdot s_3 + r_2 \cdot s_2 - r_3 \cdot s_1 \\ r_w \cdot s_3 - r_1 \cdot s_2 + r_2 \cdot s_2 + r_3 \cdot s_0 \end{bmatrix} \tag{2}$$

Secondly, the quaternions need to be normalized if they are used to express orientations. An orientation is represented by a unit-quaternion, which is a quaternion with a magnitude equal to one [Vince 2011]. Equation 3 shows the calculation of the quaternion magnitude.

$$|q| = \sqrt{q_w^2 \cdot q_1^2 \cdot q_2^2 \cdot q_3^2} \tag{3}$$

The normalization is done by dividing each element of the quaternion by $|q|$, as given in Equation 4. The resulting quaternion is illustrated as $\hat{q}$.

$$\hat{q} = \frac{q}{|q|} \tag{4}$$

In order to simplify the treatment of the robot orientation and facilitate the operations with successive orientations, quaternions were converted into rotation matrices. The advantage of using rotation matrices are: faster computation, because of the absence of trigonometric functions, which are normally used to express rotation matrices [Vince 2011]. The final 3D rotation matrix derived from a quaternion q is shown in Equation 17.
The inverse of a quaternion is calculated as shown in Equation 5. This will be used to calculate the inverse angle during the *Transformation Inversion*.

$$q^{-1} = [q_w, -q_1, -q_2, -q_3] \tag{5}$$

This notation illustrates that all elements of the quaternion except the scalar q_w, will be multiplied by -1.

The Transformation Composition is commonly represented by means of the operator $\oplus$, as shown in Equation 6.

$$X_+ = X \oplus Y \tag{6}$$

The composition of pose X with pose Y reflects their accumulation taking into account position and orientation. The resulting pose is X_+.
To perform such a computation, first, the A-matrix (Equation 17) is calculated with the quaternion of the absolute state X and is multiplied by the translation component of Y. The result is afterwards added to the translation of X, as shown in Equation 7. So far only the translation of the transformation composition is handled, what is indicated by the sub-index t.

$$X_+^t = X \oplus_t Y = \begin{bmatrix} x^X \\ y^X \\ z^X \\ 1 \end{bmatrix} + A^X \cdot \begin{bmatrix} x^Y \\ y^Y \\ z^Y \\ 1 \end{bmatrix} \tag{7}$$

Second, accumulation of both rotations by multiplying the quaternions of X and Y ($\hat{q}^Y$), to express the rotation of the transformation composition. This part of the overall result is indexed by r, as shown in Equation 8.

$$X_+^r = X \oplus_r Y = \hat{q}^X \cdot \hat{q}^Y \tag{8}$$

Where $\hat{q}^X$ and $\hat{q}^Y$ represent the normalized quaternions of X and Y. After the multiplication it is not necessary to normalize again, because the multiplication is not influencing the length of the resulting quaternion.
The final Transformation Composition, including translation and rotation, can be written as shown in Equation 9.

$$X_+ = X \oplus Y = \begin{bmatrix} X_+^t \\ X_+^r \end{bmatrix} \tag{9}$$

Although the quaternion is transformed into a rotation matrix, the composed state has the same structure as before, with x, y, z for position and a pure unit-quaternion $\hat{q}$ to express the orientation.
Recapitulating Section 1.1, in this scenario also the state itself provides some uncertainties, which are given by the linealized structure of the EKF.
The robot transformation as used in this work, is non-linear, so it is not possible to compute the covariance directly [Burguera 2009]. However, it is common to approximate the covariance by linearizing the non-linear transformation function X_+ using the Jacobian ∇X_+ and the Taylor Series of 1st order.
The Jacobian is defined in general as follows, Equation 10.

$$\nabla f = \frac{\partial f}{\partial x}|_{\hat{x}} \tag{10}$$

In the context of this work, the Transformation Composition can be seen as the function f, then it would have two arguments $f(x, y)$, respectively the absolute current pose and the isometric displacement. In this context, two Jacobian matrices are derived, one with respect to the first and the other with respect to the second argument. The two Jacobian matrices for the Transformation Composition $J_{1\oplus}$ and $J_{2\oplus}$ are presented in Equation 16 and Equation 18. For further details about Jacobian matrices, refer to [Hildebrandt 2007].
Short Summary: After the Transformation Composition has been calculated and its Jacobian matrices are derived, now it is possible to compute the corresponding Covariance C of X_+.
The covariance of the composition is calculated using Equation 11, as presented for instance in [Siciliano and Oussama 2008] and [Choset et al. 2005].

$$C_+ = J_{1\oplus} \cdot C^X \cdot J_{1\oplus}^T + J_{2\oplus} \cdot C^Y \cdot J_{2\oplus}^T \tag{11}$$

Where C^X and C^Y are the corresponding covariances to the pose states X and Y.

2.2 Inversion

The Transformation Composition together with the *Transformation Inversion* allows to calculate the relative motion between two absolute poses of the robot. This will be used during the update stage of the Extended Kalman Filter. The Transformation Inversion gives the inverse of a given pose. This operation is commonly represented by means of the operator $\ominus$.
In general, a pose transformation can be represented as a matrix, where the upper left part is occupied by the rotation matrix A described in Equation 17 and the last column contains the translation in x, y, z. See Equations 12 and 13.
It is necessary to take care of the different areas of the transformation matrix A and t, as shown in 12.

$$\begin{matrix} \vec{n} & \vec{o} & \vec{a} & \vec{p} \end{matrix} \\ \begin{pmatrix} & A & & t \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{12}$$

Due to A is a rotation matrix, inverting or building the transpose has the same result. But to invert the translation t, it is necessary to calculate the dot product between t and each column of A. The result of this procedures is shown in Equation 13.

$$\begin{pmatrix} & A & & t \\ 0 & 0 & 0 & 1 \end{pmatrix}^{-1} = \begin{pmatrix} & A^T & & -\vec{n} \circ \vec{p} \\ & & & -\vec{o} \circ \vec{p} \\ & & & -\vec{a} \circ \vec{p} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{13}$$

Where $\circ$ describes the dot product operator.
The final $\ominus$ operation looks like:

$$\ominus X = \begin{bmatrix} -\vec{n} \circ \vec{p} \\ -\vec{o} \circ \vec{p} \\ -\vec{a} \circ \vec{p} \\ q^{-1} \end{bmatrix} \tag{14}$$

Where q^-1 is the inverse quaternion of q^X, what expresses basically the same orientation as A^T.
To derive the Covariance of the Transformation Inversion, it is necessary to compute the Jacobian matrix as also done for the Transformation Composition before. This time the function f has only one argument, this means one Jacobian matrix $J_\ominus$ is obtained. This Jacobian matrix looks as shown in Equation 19. With the Jacobian matrix the Covariance C_- is computed as shown in Equation 15.

$$C_- = J_\ominus \cdot C^X \cdot J_\ominus^T \tag{15}$$

3. IMAGE REGISTRATION

After the mathematical background of pose Transformation with six degrees of freedom was described in Section 2, this Section describes the *image registration* Algorithm used specifically for the stereo SLAM approach presented in this work. The result of the image registration procedure indicates the camera motion between the two poses at which the two images that overlap were taken. This result is of vital importance to update the motion predictions based on the odometry. Without this, the obtained trajectory will be exactly the same as the odometry, including the drift. And the task is to eliminate the drift.

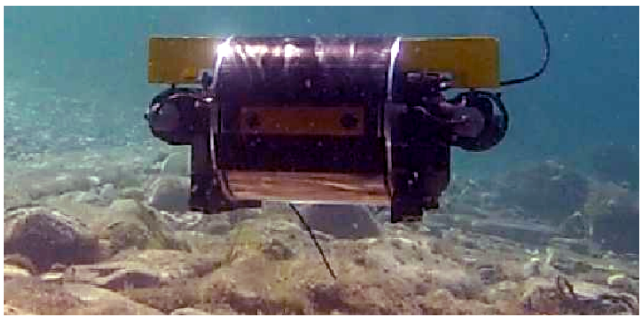

Fig. 2: Fugu-C Robot as used in this study

One iteration of the EKF receives a relative motion, given by the odometry, and a stereo image pair, given by the stereo camera system.
The Pseudocode shown in Algorithm 1 describes the main steps of the image registration process.

Algorithm 1: Image Registration

input : Current Stereo Image pair S_l, S_r and Recorded Images I_n
output: 3D Transformation $[R, t]$
begin
1 $[F_l, F_r] \leftarrow$ `stereoMatching` (S_l, S_r);
2 **for** $I_i \in I_n$ **do**
3 $F_t \leftarrow$ `findFeature` (I_i);
4 **if** `match` (F_l, F_t) == *true* **then**
5 *break*;
else
6 *continue*;
7 $[F_l, F_r] \leftarrow$ `updateFeature` (F_l, F_r);
8 $P_{3D} \leftarrow$ `calc3DPoints` (F_l, F_r);
9 $[R, t] \leftarrow$ `solvePnPRansac` (F_t, P_{3D})
end

The input of the Algorithm is the current stereo image pair S_l, S_r and all already recorded left images of the stereo sequence I_n. In this Pseudocode the Algorithm only registers one image at the same time, in the final implementation the Algorithm is able to register user-defined number of images, if possible. An example is shown in Figure 3.
Line 1 performs a stereo matching between S_l and S_r, the result is

$$J_{1\oplus} = \begin{bmatrix} 1 & 0 & 0 & 2\cdot q_2^X \cdot z^Y - 2\cdot q_3^X \cdot y^Y & 2\cdot q_2^X \cdot y^Y + 2\cdot q_3^X \cdot z^Y & 2\cdot q_1^X \cdot y^Y - 4\cdot q_2^X \cdot x^Y + 2\cdot q_w^X \cdot z^Y & 2\cdot q_1^X \cdot z^Y - 2\cdot q_w^X \cdot y^Y - 4\cdot q_3^X \cdot x^Y \\ 0 & 1 & 0 & 2\cdot q_3^X \cdot x^Y - 2\cdot q_1^X \cdot z^Y & 2\cdot q_2^X \cdot x^Y - 4\cdot q_1^X \cdot y^Y - 2\cdot q_w^X \cdot z^Y & 2\cdot q_1^X \cdot x^Y + 2\cdot q_3^X \cdot z^Y & 2\cdot q_w^X \cdot x^Y - 4\cdot q_3^X \cdot y^Y + 2\cdot q_2^X \cdot z^Y \\ 0 & 0 & 1 & 2\cdot q_1^X \cdot y^Y - 2\cdot q_2^X \cdot x^Y & 2\cdot q_3^X \cdot x^Y + 2\cdot q_w^X \cdot y^Y - 4\cdot q_1^X \cdot z^Y & 2\cdot q_3^X \cdot y^Y - 2\cdot q_w^X \cdot x^Y - 4\cdot q_2^X \cdot z^Y & 2\cdot q_1^X \cdot x^Y + 2\cdot q_2^X \cdot y^Y \\ 0 & 0 & 0 & q_w^Y & -q_1^Y & -q_2^Y & -q_3^Y \\ 0 & 0 & 0 & q_1^Y & q_w^Y & q_3^Y & -q_2^Y \\ 0 & 0 & 0 & q_2^Y & -q_3^Y & q_w^Y & q_1^Y \\ 0 & 0 & 0 & q_3^Y & q_2^Y & -q_1^Y & q_w^Y \end{bmatrix} \tag{16}$$

$$A = \begin{bmatrix} -2\cdot q_2^2 - 2\cdot q_3^2 + 1 & 2\cdot q_1\cdot q_2 - 2\cdot q_3\cdot q_w & 2\cdot q_1\cdot q_3 + 2\cdot q_2\cdot q_w & 0 \\ 2\cdot q_1\cdot q_2 + 2\cdot q_3\cdot q_w & -2\cdot q_1^2 - 2\cdot q_3^2 + 1 & 2\cdot q_2\cdot q_3 - 2\cdot q_1\cdot q_w & 0 \\ 2\cdot q_1\cdot q_3 - 2\cdot q_2\cdot q_w & 2\cdot q_2\cdot q_3 + 2\cdot q_1\cdot q_w & -2\cdot q_1^2 - 2\cdot q_2^2 + 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{17}$$

$$J_{2\oplus} = \begin{bmatrix} -2\cdot q_2^X q_2^X - 2\cdot q_3^X q_3^X + 1 & 2\cdot q_1^X \cdot q_2^X - 2\cdot q_3^X \cdot q_w^X & 2\cdot q_1^X \cdot q_3^X + 2\cdot q_2^X \cdot q_w^X & 0 & 0 & 0 & 0 \\ 2\cdot q_1^X \cdot q_2^X + 2\cdot q_3^X \cdot q_w^X & -2\cdot q_1^X \cdot q_1^X - 2\cdot q_3^X \cdot q_3^X + 1 & 2\cdot q_2^X \cdot q_3^X - 2\cdot q_1^X \cdot q_w^X & 0 & 0 & 0 & 0 \\ 2\cdot q_1^X \cdot q_3^X - 2\cdot q_2^X \cdot q_w^X & 2\cdot q_2^X \cdot q_3^X + 2\cdot q_1^X \cdot q_w^X & -2\cdot q_1^X \cdot q_1^X - 2\cdot q_2^X \cdot q_2^X + 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & q_w^X & -q_1^X & -q_2^X & -q_3^X \\ 0 & 0 & 0 & q_1^X & q_w^X & -q_3^X & q_2^X \\ 0 & 0 & 0 & q_2^X & q_3^X & q_w^X & -q_1^X \\ 0 & 0 & 0 & q_3^X & -q_2^X & q_1^X & q_w^X \end{bmatrix} \tag{18}$$

$$J_{\ominus} = \begin{bmatrix} 2\cdot q_2^X \cdot q_2^X + 2\cdot q_3^X \cdot q_3^X - 1 & -2\cdot q_1^X \cdot q_2^X - 2\cdot q_3^X \cdot q_w^X & 2\cdot q_2^X \cdot q_w^X - 2\cdot q_1^X \cdot q_3^X & 2\cdot z^X \cdot q_2^X - 2\cdot y^X \cdot q_3^X & -2\cdot y^X \cdot q_2^X - 2\cdot z^X \cdot q_3^X & 4\cdot x^X \cdot q_2^X - 2\cdot y^X \cdot q_1^X + 2\cdot z^X \cdot q_w^X & 4\cdot x^X \cdot q_3^X - 2\cdot y^X \cdot q_w^X - 2\cdot z^X \cdot q_1^X \\ 2\cdot q_3^X \cdot q_w^X - 2\cdot q_1^X \cdot q_2^X & 2\cdot q_1^X \cdot q_1^X + 2\cdot q_3^X \cdot q_3^X - 1 & -2\cdot q_2^X \cdot q_3^X - 2\cdot q_1^X \cdot q_w^X & 2\cdot x^X \cdot q_2^X - 2\cdot z^X \cdot q_1^X & 4\cdot y^X \cdot q_1^X - 2\cdot x^X \cdot q_2^X - 2\cdot z^X \cdot q_w^X & -2\cdot x^X \cdot q_1^X - 2\cdot z^X \cdot q_3^X & 2\cdot x^X \cdot q_3^X + 4\cdot y^X \cdot q_2^X - 2\cdot z^X \cdot q_w^X \\ -2\cdot q_1^X \cdot q_3^X - 2\cdot q_2^X \cdot q_w^X & 2\cdot q_1^X \cdot q_w^X - 2\cdot q_2^X \cdot q_3^X & 2\cdot q_1^X \cdot q_1^X + 2\cdot q_2^X \cdot q_2^X - 1 & 2\cdot y^X \cdot q_1^X - 2\cdot x^X \cdot q_2^X & 2\cdot y^X \cdot q_w^X - 2\cdot x^X \cdot q_3^X + 4\cdot z^X \cdot q_1^X & 4\cdot z^X \cdot q_2^X - 2\cdot y^X \cdot q_3^X - 2\cdot x^X \cdot q_w^X & -2\cdot x^X \cdot q_1^X - 2\cdot y^X \cdot q_2^X \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{19}$$

shown in Figure 3. This function consists of three steps.
(1) First it extracts SIFT features in both images, as presented in [Lowe 2004]. The results are two sets of SIFT features $sift_1$ and $sift_2$. (2) Second, the feature sets are matched. This is done by comparing the squared differences of each 128 dimensional features descriptor of $sift_1$ with $sift_2$. If the similarity reaches a certain threshold two features are matched. From this process two new sets are obtained, in particular $siftCo_1$ and $siftCo_2$. The extension Co indicates that these two sets are corresponding to each other now.
The result up to this point can be seen in Figure 3 (a). It is easy to see that, although the matching process is quite accurate, still some wrong matches remain. Such wrong matches are called outliers. A common approach to get rid of outliers is to run a *RANSAC*-Algorithm as presented in [Fischler and Bolles 1981]. Such an Algorithm filters the outliers and is performed as the last and (3) third step of the stereo matching.
Line 2 starts the loop over all images recorded from the beginning of the trajectory to the current pose. I_n. It will be stopped after I_n does not contain any more images i or if the following steps are successful.
Line 3 is similar to (1) the first step of *Line 1*, it basically detects SIFT features in the given image I_i and stores this features set into F_t.
Line 4 performs a matching between F_l and F_t. Again, this is also used during the *stereoMatching* function in steps (2) and (3) of line 1. Due to the comparison of the SIFT descriptors similar/corresponding features in both sets are able to be found. The result is again filtered by RANSAC to discard outliers. If a certain

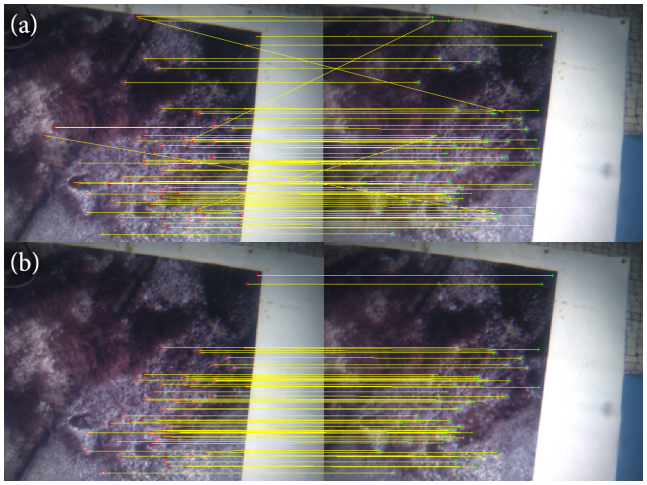

Fig. 3: Stereo matching without (a) and with RANSAC (b). The outliers are discarded during the RANSAC process providing a reliable matching result.

amount of inliers is reached, the Algorithm will exit the loop (*Line 5*) and will go to *Line 7*, otherwise the next loop-iteration with the next Element of I_n is executed (*Line 6*). The threshold to evaluate a matching as successful is basically given by the used function *solvePnPRansac*. It has been seen that this function provides

reliable results with a minimum of 17 matches.
Line 7 can been seen as a small function with a big effect. It simply updates the feature set F_r to be in line with F_l again. In the previous loop it is highly feasible that the matching between F_l and F_t (*Line 4*) will discard some features from F_l. This has the consequence that features from F_l and F_r do not match pairwise anymore, what is important for the ongoing steps of the image registration Algorithm. Without this update *solvePnPRansac* is not reliable. More information about *solvePnPRansac* will be given within the explanation of *Line 9*.
Line 8 calculates 3D points from the matching features between F_l and F_r. As given in [Siciliano and Oussama 2008] using the principle of stereoscopy the missing depth dimension (Z) of the feature can be calculated from the feature coordinates x, y using the *reprojection matrix* Q, as shown in Equation 20.

$$Q = \begin{bmatrix} 1 & 0 & 0 & -C_x \\ 0 & 1 & 0 & -C_y \\ 0 & 0 & 0 & f_x \\ 0 & 0 & -\frac{1}{T_x} & \frac{(C_x - C_{x'})}{T_x} \end{bmatrix} \tag{20}$$

Where C_x and C_y describe the optical center, f_x is the focal length and T_x is the relative translation of one camera to the other. T_x is computed as the *baseline* times f_x for the right camera and for the left camera T_x is 0. The primed parameters are taken from the intrinsic camera parameters of the left camera, the unprimed from the right.

(x_1, y_1) from F_l and (x_2, y_2) from F_r are the coordinates of two matching features, one on the left image and the other on the right image, with the disparity $d = x_1 - x_2$. With this information the 3D coordinates can be calculated as follows (Equation 21):

$$\begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = Q \cdot 1 \begin{bmatrix} x \\ y \\ d \\ 1 \end{bmatrix} \tag{21}$$

This procedure is applied for each pair of matching features in F_r and F_l, so n 3D points are obtained, if n features were stored in F_r and F_l, respectively. It is important that F_r and F_l have the same order and number of elements. The result is stored in P_{3D}.
Line 9 solves the *Perspective N-Point* problem (PNP) with additional use of RANSAC to make the function resistant to outliers. The function estimates with specification of the PNP problem a pose transformation that minimizes the reprojection error of a given set of 3D features onto a set of corresponding 2D features. Beside the 3D and 2D features, the intrinsic camera matrix K is necessary (Equation 22). K can be obtained during a calibration process as given in [Opower 2002].

$$K = \begin{bmatrix} f_x & s & C_x \\ 0 & f_y & C_y \\ 0 & 0 & 1 \end{bmatrix} \tag{22}$$

Where f_x, f_y are the focal length again, s the skew-parameter and C_x and C_y describe the optical center. The result of *Line 9* is a pose transformation that minimizes the error of reprojecting the 3D points obtained from the current stereo pair and calculated in line 8 onto the 2D features of the image I candidate to close a loop with the current view calculated in line 3. The PNP-problem is widely discussed and can be found in literature formulated in multiple solutions. This technique is applied in a wide range of applications such as *object recognition*, *structure from motion* (*sfm*) and others [Bujnak et al. 2011] [Mei 2012]. The result is a 3D transformation $[R, t]$ that transforms S_l, by using rotation and translation, into the loop closing image I_i, if both images have a certain overlap. Elsewhere, if the overlap is not big enough the image registration process fails.
The *solvePnPRansac* function in the evaluation implementation was taken from the computer vision library *OpenCV*.
A sample of the final result of the whole image registration process can be seen in Figure 4. The image on the left shows the left frame of a stereo pair S_l and the image in the middle shows the loop closing candidate I_i recorded during one experiment in a pool of the University of the Balearic Islands. On the right, the result of applying the transformation given by the image registration on S_r can be seen. The purple color indicates the error, which are areas in which both images do not perfectly fit.

Fig. 4: Left: S_l; middle: loop closing image I_i. On the right: the transformation of the image registration applied to S_l. The purple color indicates the error of the transformation.

4. VISUAL EKF-SLAM

This work performs *Simultanous Localization and Mapping* (SLAM) based on *Extended Kalman Filtering* (EKF). EKF belongs to the family of Bayesian filters which estimate the state of a non-linear system with normally distributed gaussian noise [Welch and Bishop 1995]. In principle, the EKF estimates continuously the pose of the vehicle by applying three steps. The prediction and the state augmentation steps where the current vector state is augmented with the current odometry measurement and the vehicle future pose is predicted by composing the current vehicle pose with the current odometry. Afterwards, predictions are corrected in the update phase of the Kalman filter using as observations the detected loop closings [Schattschneider et al. 2011].
The Pseudocode shown in Algorithm 2 describes the main steps of the pose based visual EKF-SLAM process.

Algorithm 2: Visual EKF-SLAM

input : $X, C, O, C_o, S_l, S_r, C_m\ I_n$
output: Updated state vector X_u, covariance C_u and recorded Images I_u
begin
; /* Prediction stage */
$X_t \leftarrow$ `getLastState` (X) ;
$C_t \leftarrow$ `getLastCovariance` (C) ;
$[X_t^+, C_t^+] \leftarrow$ `composition` (X_t, C_t, O, C_o) ;
; /* Augmentation stage */
$X^+ \leftarrow$ `addState` (X, X_t^+);
$C^+ \leftarrow$ `addCovariance` (C, C_t^+);
; /* Update stage */
$z \leftarrow$ `imageRegistration` (S_l, S_r, I_n);
if `imageRegistration` == *false* **then**
return;
else
$[h, H] \leftarrow$ `calcHkK` (X^+, z) ;
$y \leftarrow$ `innovation` (h, z) ;
$S \leftarrow$ `innovationCov` (C^+, H, C_m) ;
$K \leftarrow C^+ \cdot H^T \cdot S^{-1}$;
$X_u \leftarrow X^+ + K \cdot y_k$;
$C_u \leftarrow (1 - K \cdot H) \cdot C^+$;
$I_u \leftarrow I_n \bigcup S_l$;
end

The Algorithm has as input the current state vector X. X is formed by the vehicle pose state $(x, y, z, q_w, q_1, q_2, q_3)$ estimated at each iteration. It is initially set to $[0, 0, 0, 1, 0, 0, 0]$, what represents the robot pose at position $(0, 0, 0)$ with an orientation of 0 degrees at every axis. The second parameter C is the global covariance of the EKF, which is initially also set to a 7×7 zero-matrix. The next argument is for predicting the robot motion and is the odometry O, which is the displacement of the robot between two given successive points. Associated to O a covariance C_o is received, the uncertainty of the odometry. S_l and S_r are already known to be a stereo image pair, which will be used for the update stage, as well as the set of recorded images I_u. To express the uncertainty of a measurement, in this case the image registration, another covariance C_m is introduced.
The output of the Algorithm is, if one or several loop closings are found, an updated (corrected) state vector X_u, as well as an updated covariance matrix C_u and the set of images I_n is now extended by S_l and is named I_u. X_u is used in the next iteration as the input X.
Every time the Algorithm is executed, the size of X and C grows. How this is done will be briefly explained now. X is the so called state vector, which is storing robot pose states as shown in Equation 1. After $n-$ executions X looks like as shown in Equation 23.

$$X = \left[\underbrace{x^1\ y^1\ z^1\ q_w^1\ q_1^1\ q_2^1\ q_3^1}_{\text{vehicle pose at 1st iteration}} \cdots \underbrace{x^n\ y^n\ z^n\ q_w^n\ q_1^n\ q_2^n\ q_3^n}_{\text{vehicle pose at nth iteration}} \right]^T \quad (23)$$

It is easy to see that as new states are just added at the end of the vector, and after n-executions the length of the vector is $n \cdot 7$.
C is build quite similarly. At every iteration, the new obtained covariance C_t^+ of the predicted motion is added on the diagonal of C, this will cause after n-iteration a covariance dimension of $7{\cdot}n \times 7{\cdot}n$, as can be seen in Equation 24.

$$C = \begin{bmatrix} \begin{bmatrix} \sigma_{11}^1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_{22}^1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{33}^1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{44}^1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{55}^1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{66}^1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \sigma_{77}^1 \end{bmatrix} & \mathbf{A} & & \mathbf{B} & \\ \mathbf{C} & & \ddots & & \mathbf{D} & \\ \mathbf{E} & & \mathbf{F} & \begin{bmatrix} \sigma_{11}^n & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_{22}^n & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{33}^n & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{44}^n & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{55}^n & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{66}^n & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \sigma_{77}^n \end{bmatrix} \end{bmatrix} \quad (24)$$

Where the matrix in the upper left corner (σ_{11}^1, σ_{22}^1, ...) is the covariance matrix of the first iteration and the matrix in the bottom right corner (σ_{11}^n, σ_{22}^n, ...) is C_t^+.
Besides this $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{E}, \mathbf{F}$ are computed in another way. An element of the last row of C, which has also the size 7×7, is the result of the multiplication of the current Jacobian $J_{1\oplus}$ by its antecessor. Analogically for the last column, whereas the antecessor is multiplied by the transposed $J_{1\oplus}$. This means in a mathematical notation, $\mathbf{E}$ is derived as shown in Equation 25.

$$\mathbf{E} = J_{1\oplus} \cdot \mathbf{C} \quad (25)$$

An element of the last column, for instance $\mathbf{B}$ is calculated as shown in 26.

$$\mathbf{B} = \mathbf{A} \cdot J_{1\oplus}^T \quad (26)$$

Where $\mathbf{A}$ and $\mathbf{C}$ are covariances from the previous iteration calculated with the same scheme as as shown in Equation 25 and 26.
This procedure can be found in numerous EKF-literature as in [Welch and Bishop 1995], [Thrun et al. 2005], [Schattschneider et al. 2011]. It is quite important to introduce the last row/column computations, otherwise EKF would only update certain states, which belong to loop closings, and not the whole vector.

4.1 Prediction stage

The first stage of an EKF is the *Prediction stage*. In this stage the filter predicts the state by a given motion estimate of the odometry O and the associated covariance C_o. This process uses the *Transformation Composition* explained in chapter 2. The whole stage goes from *Line 2* to *Line 4*.
Line 2 extracts the last absolute pose of the state vector, since the state vector is always storing absolute poses in this EKF implementation, and stores it in X_t. The last element can be obtained by just taking the last 7 elements of X.
Line 3 is similar to *Line 2*, instead of extracting the last state, it extracts the last covariance from C and stores it in C_t. The last element is taken from the bottom right 7×7 matrix, which is for instance in Equation 24 the illustrated element with σ_{11}^n, σ_{22}^n.
Line 4 performs the composition $\oplus$ between the last pose of the vector state and the last odometry sample as explained in chapter 2. Provided to this function are the absolute state X_t, the relative motion O and to each the associated covariances. In Section 2 X_t was introduced as X and O as Y in Equation 1. The result of the composition is a new absolute state X_t^+, and a new corresponding covariance C_t^+.

4.2 Augmentation stage

The second stage in this EKF implementation is called the *Augmentation stage*. Every time a new image of the stereo camera system is

available, the state is augmented by the current X_t^+. As illustrated by Equation 23 and 24 X and C will grow, this is done in this stage, by augmenting them with the results of the *Prediction stage*. The augmentation stage includes *Line 6* and *7*.
Line 6 augments the state vector X by putting the predicted pose state X_t^+ at the end. The result is a new vector X^+, with 7 new elements.
Line 7 augments C with C_t^+ at the bottom right. The resulting C^+ has now the size of $7 \cdot (n+1) \times 7 \cdot (n+1)$.

4.3 Update stage

The last main stage of the EKF is the *Update stage*. In the context of this work the update depends on the success of the image registration. If the Algorithm does not find any loop closing, the process will be interrupted and the system will wait for the next iteration. This stage is the main part of EKF. If the image registration is successful, the Kalman Equations are executed and X^+ gets corrected. The loop closing process is already described in Section 3. The following Figure 5 shows the interpretation of the result z, which is a relative motion between the current state and the loop closing candidate.

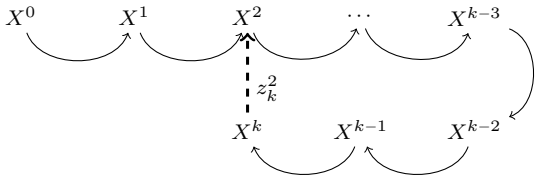

Fig. 5: Illustration of a loop closing (dashed arrow) and the current state vector (black arrows).

X^0, X^1, ... are elements of the state vector which represent the successive absolute poses of the vehicle along its trajectory, that is why this approach is posed based EKF-SLAM. After k iterations the image registration was able to detect a loop closing by registering the current image with the stored image of state X^2. The result is a relative motion from X^k to X^2 and is called z_k^2. With z_k^2 it is possible to run the remaining Kalman Equations and update the robot-localization, what is described now in more detail. The update stage ranges from *Line 9* to *Line 18*.
Line 9 performs the image registration with all recorded images I_n and the current stereo image pair S_l and S_r. If the registration was successful, what means that, at least, one loop closing was found, *Line 10* will be evaluated to be false. Otherwise the Algorithm will go to *Line 11* and return without updating the state vector.
Line 12 calculates based on z the corresponding relative motions of the state vector. Remember z was computed without any influences of the state, it is a pure product of the image registration process. For each image registered with the current one giving a transformation called measurement (z_k), it is possible to associate an observation calculated with the corresponding elements of the state vector. For the example of Figure 5, the calculation of the Kalman observation function h is shown in the following Equation 27.

$$h = \ominus X^k \oplus X^2 \tag{27}$$

In the sense of EKF, h is known as the *observation function* and it grows with the number of detected loop closings. If the image registration has found four loop closings, h will have the length of 28, which can be seen in Equation 28 as a set of loop closings. h denotes how the loop closings are expected to be according to the state vector and z denotes how they actual are. The EKF corrects the state vector by trying to change the state vector according to the actual loop closings.

$$h = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_n \end{bmatrix} \tag{28}$$

Where n in this example would be 4. Besides the observation function, the *observation matrix* H is also calculated. This matrix has as many rows as loop closings have been found (times 7) and columns as many states are stored in X^+ (times 7). H stores basically the result of the following Jacobian matrix. What are actually the partial derivatives of the observation function h with respect to the state vector X^+, which is defined as follows in Equation 29.

$$H = \left.\frac{\partial h}{\partial X^+}\right|_{\hat{X}^+} \tag{29}$$

All elements of H, which are not referring to the states used to calculate the relative motions, as given in 27, will be zero. For instance, if one loop closing between X^k and X^2 is found, it means that H will have the size of $7 \cdot 1 \times 7 \cdot k$ and the partial derivatives will be only non-zero at the positions which correspond to the state 2 and k. The resulting H is shown in Equation 30.

$$H = \begin{bmatrix} \mathbf{0} & \frac{\partial h^1}{\partial X^2} & \mathbf{0} & \dots & \mathbf{0} & \frac{\partial h^1}{\partial X^k} \end{bmatrix} \tag{30}$$

If more loop closings have been found, for instance with state X^3, H would consists of one more row and the structure will be as shown in Equation 31.

$$H = \begin{bmatrix} \mathbf{0} & \frac{\partial h^1}{\partial X^2} & \mathbf{0} & \dots & \mathbf{0} & \frac{\partial h^1}{\partial X^k} \\ \mathbf{0} & \mathbf{0} & \frac{\partial h^2}{\partial X^3} & \dots & \mathbf{0} & \frac{\partial h^2}{\partial X^k} \end{bmatrix} \tag{31}$$

Line 13 calculates the *innovation* y of the EKF. It describes the discrepancy between the observation function h and the measurement function z, thus so how good or bad the observation is in comparison to the measurement. In case the estimation of h is bad, the innovation gets big, is the estimation good, y gets small. Thus the update will be big or small and can be seen proportional to the size of y. Taking all this into account the discrepancy is often described by a pure subtraction of $y = z - h$. The innovation of the position holds these criteria by using a pure subtraction. For example, if the stored translation from z and h are quite different, the resulting innovation of the position is big. But since quaternions are used to describe the orientation of a state, a pure subtraction is not providing a correct innovation of the orientation. Different looking quaternions can express a similar orientation. One example is given in Equation 32.

$$q_z = [0.9964, -0.0109, 0.0145, 0.0830]$$

$$q_h = [-0.9964, -0.0183, 0.0011, -0.0833]1 \tag{32}$$

This example shows the orientation for q_z with pitch $= 1.55°$, roll $= -1.38°$ and yaw $= 9.5046°$. For q_h with pitch $= 0.04°$, roll $= 2.09°$ and yaw $= 9.5543°$. A pure subtraction would provide a big innovation, as can been seen in 33.

$$y_q = q_z - q_h = [1.99274, 0.007344, 0.013427, 0.166257] \tag{33}$$

A big innovation in this case is not right, because the angles are quite similar. One way is to interpret the quaternions as Euler angles, perform the subtraction with them and go back to a quaternion. But this leads to the next problem. As shown in this example the resulting rotations in euler angles are small, but expressed as a quaternion y_q, the magnitudes are big. This is due to the definition of quaternions. Does it express big angles, for example $180°$, they are mapped to a value around 0. Small angles around $0°$ are mapped to a value near to 1. So exactly the opposite how the EKF-innovation is defined.
In this study, the innovation of the orientation is performed by taking the absolute values of the quaternion before subtracting, as presented in Equation 34.

$$y_q = |q_z| - |q_h| \tag{34}$$

The given example from before would provide the result as shown in Equation 35.

$$y_q = [0.0000, -0.0073, 0.0134, -0.0003] \tag{35}$$

It is easy to see, that the values of the innovation of the orientation are small. This means, that the update is small, what is correct, because the observation is similar to the measurement.
The innovation of the position is done by a pure subtraction, as already mentioned, and is expressed as given in Equation 36.

$$y_p = \begin{bmatrix} x^X \\ y^X \\ z^X \end{bmatrix} - \begin{bmatrix} x^Y \\ y^Y \\ z^Y \end{bmatrix} \tag{36}$$

If everything is put together, the final innovation of the update stage, as used in this work, is formulated as follows (Equation 37):

$$y = \begin{bmatrix} y_p \\ y_q \end{bmatrix} \tag{37}$$

Line 14 computes the *innovation covariance* S. This matrix is computed straightforward as already presented in other EKF-Literatur, such like [Matsebe et al. 2008] [Thrun et al. 2005] [Siciliano and Oussama 2008] and [Welch and Bishop 1995]. The Equation is given in 38.

$$S = H \cdot C \cdot H^T + R \tag{38}$$

The big covariance matrix C and the observation matrix H have been detailed in previous sections. Now, it is necessary to introduce the *measurement covariance* R, which is used to compute S. This matrix is build with respect to the number of loop closings, which can be obtained by the number of rows of H divided by 7, and the uncertainty of the image registration C_m. For each loop closing R keeps one entry of C_m on its diagonal. The structure of the matrix, if three loop closings were found, can be seen in Equation 39.

$$S = \begin{bmatrix} C_m & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & C_m & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & C_m \end{bmatrix} \tag{39}$$

Line 15 calculates the *Kalman Gain*, what represents the strength of the update or how much the state vector will be changed. One slight change, due to computation improvement, has been done by dealing with the inverse of S. In many implementations K is calculated with the inverse of S. What is, especially after some iterations, a huge computational effort and can lead to less accuracy. Instead of multiplying by the inverse, in this implementation, K is computed by a matrix right division (Equation 40).

$$K = (C^+ \cdot H^T)/S \tag{40}$$

Line 16 finally updates the state vector by taking all information together. In this implementation this is done like in common EKF-Algorithms and the Equation is given in 41.

$$X_u = X^+ + K \cdot y \tag{41}$$

Line 17 updates the covariance matrix C. The Equation is given in 42.

$$C_u = (1 - K \cdot H) \cdot C^+ \tag{42}$$

Where 1 represents an identity matrix of the size of the product $K \cdot H$.
Line 18 stores the new image S_l to I_u to use this image during upcoming iterations as a loop closing candidate. This process is performed every time a new state is added to the state vector, this is due to the fact, that to each state the corresponding image is needed. Another important note is, that the state vector X is not storing the images contrarily other implementations.

5. RESULTS

After the visual EKF-SLAM approach has been explained, in this chapter the results will be presented.
The system used for the software evaluation is a laptop with an Intel core i7 ($2 \times 2.9Ghz$), 8GB RAM and a Solid State Drive, running MATLAB R2013a on Ubuntu 12.04 with a single CPU core. The robots mission was recorded using *ROS* (Robot Operation System), a widely used software framework for robot software development. Thanks to the rosbag technology provided by ROS, the mission can be reproduced offline viewing the images recorded online. With this, it is possible to run the localization task offline on a laptop using the data collected online.
The experiments were conducted with a robot called Fugu-C, as illustrated in Figure 2. Fugu-C is developed by the *University of the Balearic Islands* and is a low-cost mini-AUV. The robot provides two stereo rigs, one down looking, the other one looking forward, a MEMS-based Inertial Measurement Unit and a pressure sensor. For these experiments the down looking camera system was used only. The Camera used to perform the image registration is a *Point Grey Bumblebee 2* with a resolution of 1032×776 pixel and a baseline of 12 cm. The video sequences were recorded in a water tank located in the UIB. The tank is 7 meters long, 4 meters wide and 1.5 meters deep. The bottom is covered with a repeating image pattern of a real seabed. To obtain the ground truth, each image recorded online was registered to the whole printed image, which is known.
The odometry to feed this SLAM approach was obtained with *LIBVISO2* (Library for Visual Odometry 2) [Geiger et al. 2011]. This is a fast, reliable and cross-platform library to compute the motion of a moving mono/stereo camera and was already used in several other publications of this working group.
The mission of this experiment is a sweeping task. During the mission, the robot is gathering images from the down looking stereo image system and the other sensors are not used. From successive stereo pairs of the rig, LIBVISO2 calculates the odometry, which is used later to run the algorithm off-line to enhance the localization.
In order to assess the performance of the SLAM approach with different levels of error and drift in the visual odometry, the results of the LIBVISO2 were corrupted with different levels of noise. In total six noise levels were tested 20 times to obtain significant statistical results. The noise used is additive zero mean Gaussian and the noise covariance ranges from $[\Sigma_x, \Sigma_y, \Sigma_z, \Sigma_{qw}, \Sigma_{q1}, \Sigma_{q2}, \Sigma_{q3}] = [0, 0, 0, 0, 0, 0, 0]$ (noise level

Noise Level	*1*	*2*	*3*	*4*	*5*	*6*
Covariance	*0*	*3e-9*	*9e-9*	*3e-8*	*5e-7*	*3e-6*
Odom. error ∅	0.038	0.417	0.494	0.806	2.614	6.898
EKF error ∅	0.027	0.282	0.285	0.309	0.590	0.953
Improv. (%)	28.9	32.3	42.3	61.6	77.4	86.1

Table I. : Comparison between visual odometry and EKF-SLAM trajectory mean error $(\varnothing)$ with respect to the ground truth. Error is measured in meters per traveled meter.

1) to $[\Sigma_x, \Sigma_y, \Sigma_z, \Sigma_{qw}, \Sigma_{q1}, \Sigma_{q2}, \Sigma_{q3}] = [3e-6, 3e-6, 3e-6, 3e-6, 3e-6, 3e-6, 3e-6]$ (noise level 6). The error function used is as follows. In order to have a quantitative measure of the quality of the SLAM estimates, we defined the trajectory error as the difference between the ground truth and the corresponding estimate given by the odometry and by the EKF, divided by the length of the trajectory. Calculated like this, the obtained error units are meters per traveled meter. The advantage of this technique is a comparable result of the different experiments and future missions.

5.1 Quantitative Results

Table I shows that the presented EKF-SLAM approach improves the robot pose estimates compared with the odometry estimates, since the mean of the trajectory errors with respect to the ground truth are clearly smaller. In the first example, where actually no noise is used, the improvement is 28.1%. Although the odometry is quite good, with an mean error of $0.038m$, EKF-SLAM was able to improve it to $0.027m$. When the noise level added to the odometry increases, the correction given by the EKF-SLAM is more evidently reflected in the percentage of improvement. The odometry gets more corrupted and provides a mean error of $0.806m$ with a noise level of 4. In this scenario it is significant that the error of EKF-SLAM is increasing only slightly, for example from level two to four just about $0.027m$. This observation can also be seen in the next higher noise levels. Even in the highest level, where the odometry provides an error of $6.898m$ and can be considered useless for localization, the EKF-SLAM approach is able to update the trajectory successfully. This is off course reflected in the amount of improvement, which is 86%.
It can be seen in Figure 6 that the odometry trajectory mean error raises very fast by applying a noise level from three and higher, but the trajectory mean error by EKF-SLAM increases only slowly.
The computational effort of this algorithm implemented in MATLAB is quite big without any further improvement. If the image registration is performed at the half of the frame rate, the runtime is in average $8.4min$, in comparison to $4.3min$, what the total mission time of the $23.42m$ long sweeping task is. But if the key-frame separation is increased the algorithm runs much faster, as can be seen in Table II. This is a possible technique to improve the run-time, because the update Equations are executed at lower frequency. Especially the calculation of C_u (Pseudocode 2 Line 17) takes a long time because with each iteration C_u grows.
But increasing the key-frame separation has an evident disadvantage: the update Kalman Equations are executed with a lower frequency, which means that the localization system will rely on the odometry during longer periods of time. For example, if the update Equations are run every eight frames, the correction of the motion estimated by the odometry will be run at a frequency equal to the frame rate divided by eight.
This is influencing the error as well. Due to having less loop closings the error goes up. But even with a separation of four images,

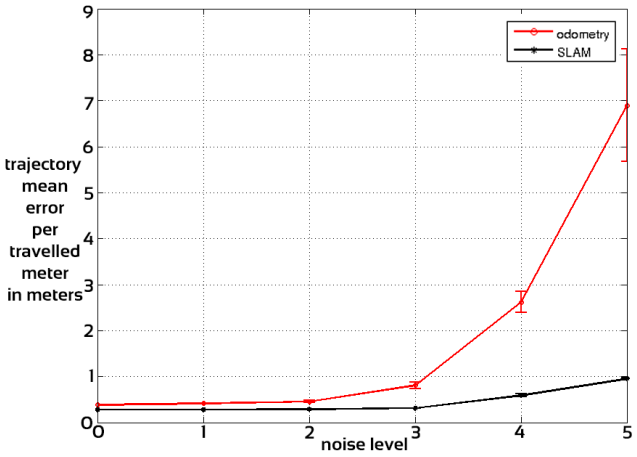

Fig. 6: Comparison between state mean errors using raw odometry and EKF pose estimates. $y-$axis shows the error per travelled meter in meters. $x-$axis represents the different noise levels. The standard deviation is set to 0.1σ to provide a clear representation.

Separation between frames	*2*	*4*	*8*
Run-Time (min)	8.4	4.3	2.3
error (m)	0.28	0.32	0.39

Table II. : Comparison run time of different key-frame separations and error. Used noise level 2.

the system obtains good results and terminates within only $4.3min$. This results show, that it is already possible to run the code on-line in real-time. If the separation level is set to eight, the run time can be decreased to $2.3min$. In this case the error is increasing another $0.07m$ to $0.39m$. Table II shows how by increasing the frame separation at which the update step is executed, the running time decreases, but also the mean trajectory error increases.

5.2 Qualitative Results

Besides the quantitative results, the qualitative results are also important. These results now show what the trajectory looks like, in contrast to the number of error per meter.
As can be seen in Figure 7, the EKF-SLAM approach is able to correct even huge errors given by the odometry. The ground truth is plot in blue, the odometry in black and the EKF estimates in red. All units in all of trajectories are expressed in meters.
Even if the noise level is very high and the odometry is far away from the ground truth, EKF-SLAM corrects the trajectory satisfactorily. The visual error as seen in the plots is increasing only slightly. This leads to the conclusion that the given approach is robust against corrupted odometry with even a huge drift (noise level six).
In Figure 8 a trajectory with a noise level of two is shown, with its eight loop closings in magenta, which are used by EKF-SLAM to update the localization. All experiments were using the same loop closings, to make the results more comparable. Normally an EKF-SLAM approach would use more loop closings (30 and more), but even with only a few, the algorithm presented in this paper is able to perform good results.
As already mentioned during the quantitative evaluation, it is pos-

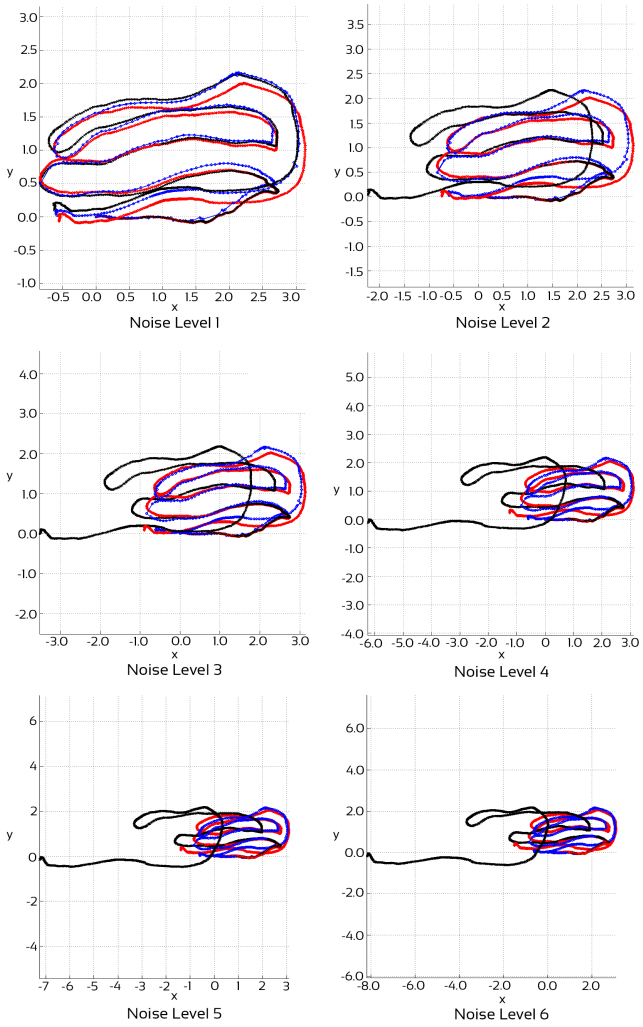

Fig. 7: Example results of the experiments. Starting from noise level one and goes up to noise level six. The blue trajectory is the ground truth, red is EKF and black is the pure visual odometry, identically in all following Figures and units are expressed in meters.

sible to decrease the execution time significantly by decreasing the frequency at which the update stage is executed. By doing so, the error of EKF-SLAM goes slightly up, but still the obtained localization is much better than the pure odometry. In Figure 9 the result of using an image separation of four is shown. It is easy to see that, the result is almost as good as in Figure 8, although three loop closings were used instead of seven. The improvement of the execution time is in average 48.8%.
The execution time can be improved even more by increasing the frame separation to eight, as illustrated in Figure 10. With an improvement in average of 72.6% $(2.3m)$.

6. CONCLUSION

This paper presents a pose based visual EKF-SLAM approach to perform underwater localization. This work is mainly focused on performing a pure stereo localization approach using variables

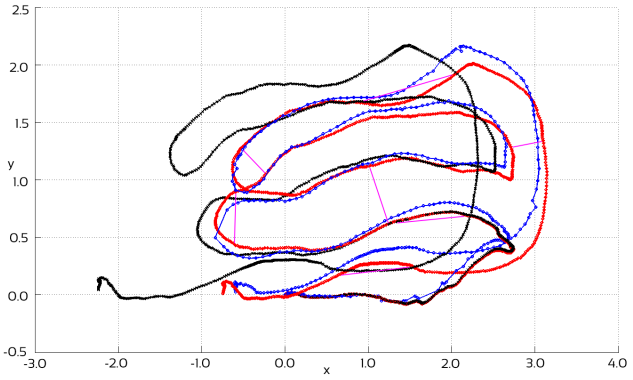

Fig. 8: Example result with a noise level of two. Additionally the eight loop closings are plotted (magenta lines).

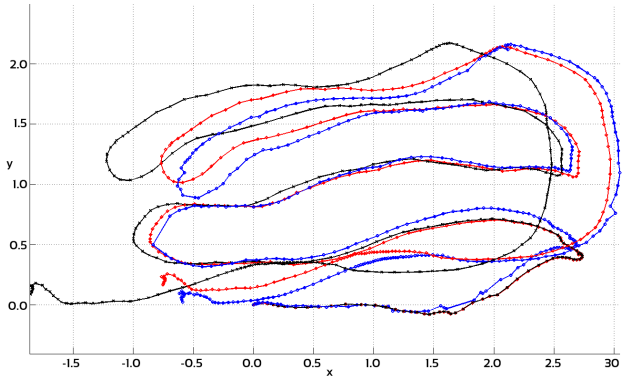

Fig. 9: Example results of the experiments. Using noise level two and a image separation of two.

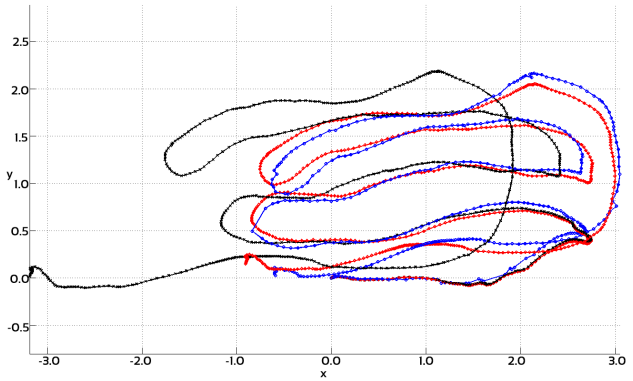

Fig. 10: Example results of the experiments. Using noise level three and a image separation of eight.

with 6 DOF $(x, y, z, roll, pitch, yaw)$ during the whole filtering process, where the orientation is represented by quaternions. Besides, the image registration is a crucial factor, providing the Algorithm with the necessary robust measurements to perform

the update stage. As seen in the results, the registration works precise and reliable. Thanks to that, as seen in chapter 5, the given approach improves satisfactorily the 3D underwater visual odometry in a controlled underwater scenario. Furthermore, the system is robust against highly corrupted odometry (improvement of 86.1%). This improvement gets smaller the better the odometry is, but still an odometry with no additional noise and a drift of just $0.038m$ was improved by 28.9% to an error of $0.027m$.
The Algorithm can be further enhanced by adapting the measurement and observation covariance matrices. This means to provide the Algorithm with more correct uncertainties of the measurement and the prediction. Only with this the Algorithm is able to work on its optimum. For example, if the covariance of the measurement is too big, the Algorithm will rely less on the measurement and more on the prediction, even if the measurement is more reliable than the prediction. These two matrices are the main parameters to tune the Algorithm, which are highly dependent on the stereo camera system used. In the future, the matrices will be improved to obtain the best results of this system.
Further on, more tests should be done, especially to test the system in the sea. The tests made in the pool, with a poster as a relief, showed good results, off course a pool is not the actual use case, but the environment gets quite close to the real one. The ROS implementation will be done as a next step and will be published on the working-group repository to provide it to the whole scientific world. The current MATLAB implementation with some test data is already published: `https://github.com/srv/6dof_stereo_ekf_slam`.

Under the following links some videos of the results of the sweeping experiments can be seen:

(1) `http://youtu.be/zR4TKjrbG3M`

(2) `http://youtu.be/xV3_DtVneL8`

(3) `http://youtu.be/CmEuQ0dTUhU`

The first link (1) shows an experiment with a key frame separation of 2. This led to an execution time of $8.4m$ and an improvement of 86.1%. This improvement was due to seven found loop closings in an odometry corrupted by noise level four. The second link (2) shows the same experiment but with a less corrupted odometry (noise level three). Due to doubling the key frame separation to 4, the execution time was $4.3m$. The improvement was 79.5% with three loop closings detected. The third link (3) leads to the sweeping task with a key frame separation of eight, what was decreasing the execution time to $2.3m$. The improvement with two loop closings was 72.8%.
To conclude, from the author's point of view, the next steps have to be (1) improvement of the covariance matrices and (2) tests in an un-controlled underwater scenario.

ACKNOWLEDGMENTS

I want to thank following people, who always took the time to answer my questions and listen to my ideas: Francisco Jesus Bonin Font, Antoni Burguera Burguera and Pep Lluis Negre. Furthermore, I am grateful to the SRV-group (Systems, Robotics & Vision Group) for hosting me for six months. Last but not least I want wholeheartedly thank my parents.

REFERENCES

BAY, H., ESS, A., TUYTELAARS, T., AND GOOL, L. V. 2008. Speeded-up robust features (SURF). *Comput. Vis. Image Underst. 110,* September, 346–359.

BEALL, C., DELLAERT, F., MAHON, I., AND WILLIAMS, S. B. 2011. Bundle adjustment in large-scale 3D reconstructions based on underwater robotic surveys. *OCEANS 2011 IEEE - Spain*, 0–5.

BUJNAK, M., KUKELOVA, Z., AND PAJDLA, T. 2011. New efficient solution to the absolute pose problem for camera with unknown focal length and radial distortion. *Computer VisionACCV 2010*.

BURGUERA, A. 2009. A contribution to mobile robot localization using sonar sensors. Ph.D. thesis, Universitat de les Illes Balears.

BURGUERA, A., BONIN-FONT, F., AND OLIVER, G. 2014. Towards Robust Image Registration for Underwater Visual SLAM. In *International Conference on Computer Vision, Theory and Applications (VISSAP)*. Lisboa.

CHOSET, H., LYNCH, K. M., HUTCHINSON, S., KANTOR, G. A., BURGARD, W., KAVRAKI, L. E., AND THRUN, S. 2005. *Principles of Robot Motion*. MIT Press, Boston.

DURRANT-WHYTE, H. AND BAILEY, T. 2006. Simultaneous localization and mapping: part I. *IEEE Robotics & Automation Magazine 13*.

EUSTICE, R. M., PIZARRO, O., AND SINGH, H. 2008. Visually Augmented Navigation for Autonomous Underwater Vehicles. *Ieee Journal Oceanic Engineering 33*, 103–122.

FISCHLER, M. A. AND BOLLES, R. C. 1981. Random Sample Consensus: A Paradigm for Model Fitting with. *Communications of the ACM 24*, 381–395.

GEIGER, A., ZIEGLER, J., AND STILLER, C. 2011. StereoScan: Dense 3d reconstruction in real-time. *IEEE Intelligent Vehicles Symposium, Proceedings* Iv, 963–968.

HILDEBRANDT, S. 2007. *Analysis 2* -, 1. Aufl. ed. Springer Berlin Heidelberg, Wiesbaden.

KINSEY, J. C., EUSTICE, R. M., AND WHITCOMB, L. L. 2006. Navigation : Recent Advances and. *IFAC Conference on Maneuvring and Control of Marine Craft*.

KUIPERS, J. B. 2002. *Quaternions and Rotation Sequences - A Primer with Applications to Orbits, Aerospace, and Virtual Reality*, Reprint ed. Princeton University Press, Kassel.

LEE, P.-M., KIM, S.-M., JEON, B.-H., CHOI, H. T., AND LEE, C.-M. 2004. Improvement on an inertial-Doppler navigation system of underwater vehicles using a complementary range sonar. *Proceedings of the 2004 International Symposium on Underwater Technology (IEEE Cat. No.04EX869)*.

LOWE, D. G. 2004. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision 60,* 2 (Nov.), 91–110.

MATSEBE, O., NAMOSHE, M., AND TLALE, N. 2008. Basic Extended Kalman Filter Simultaneous Localisation and Mapping. In *Structure*. Number 1. InTech, Chapter three, 39–58.

MEI, C. 2012. Robust and accurate pose estimation for vision-based localisation. *IEEE International Conference on Intelligent Robots and Systems*, 3165–3170.

OLSON, E., LEONARD, J., AND TELLER, S. 2006. Fast iterative alignment of pose graphs with poor initial estimates. In *International Conference on Robotics and Automation (ICRA)*. Number May. Orlando FL, 2262–2269.

OPOWER, H. 2002. *Multiple view geometry in computer vision*, Second ed. Vol. 37. Cambridge University Press.

RIBAS, D., RIDAO, P., TARDOS, J. D., AND NEIRA, J. 2007. Underwater SLAM in a marina environment. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*. 1455–1460.

SALVI, J., PETILLOT, Y., AND BATLLE, E. 2008. Visual SLAM for 3D large-scale seabed acquisition employing underwater vehicles. *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, 1011–1016.

SCHATTSCHNEIDER, R., MAURINO, G., AND WANG, W. 2011. Towards stereo vision SLAM based pose estimation for ship hull inspection. *Oceans 2011*, 1–8.

SICILIANO, B. AND OUSSAMA, K. 2008. *Handbook of Robotics*. Springer, Berlin, Heidelberg.

SMITH, R., SELF, M., AND CHEESEMAN, P. 1988. A stochastic map for uncertain spatial relationships. *Proceedings of the 4th international symposium on Robotics Research*, 467–474.

THRUN, S. 2006. The Graph SLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures. *The International Journal of Robotics Research 25*, 403–429.

THRUN, S., BURGARD, W., AND FOX, D. 2005. *Probabilistic Robotics*. The MIT Press.

VINCE, J. 2011. *Rotation Transforms for Computer Graphics -Vince, J. (2011). Rotation Transforms for Computer Graphics - (1. Aufl.). Berlin Heidelberg: Springer Science & Business Media.*, 1. Aufl. ed. Springer Science & Business Media, Berlin Heidelberg.

WELCH, G. AND BISHOP, G. 1995. An introduction to the Kalman filter. Tech. rep., University of North Carolina at Chapel Hill, Chapel Hill, NC, USA.