6 DOF EKF SLAM in Underwater Environments

Markus Solbach

Final Masters Project
Universitat de les Illes Balears

September 24, 2014

Problem Statement

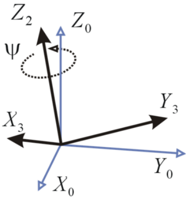
Problem Statement

- Accessibility of the sub-aquatic world is important for research and industry
- AUV[1] promising advantages compared to ROV[2]
  - Untethered, independent, self-powered, ...
- Question: **How to perform the localization** of AUVs
- Accurate localization is important for the mission success
  - Maintenance, Rescue Operations, Sampling, Inspections, ...

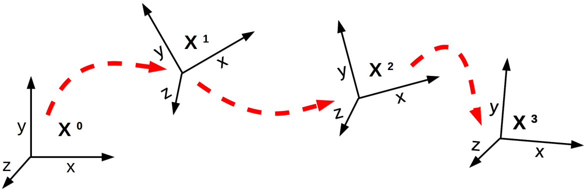
[1]Autonomous Underwater Vehicle
[2]Remotely Operated Vehicle

Vehicle Localization

- **pose** = Position and Orientation
- 6 Degrees of Freedom
  - 3 Translation
  - 3 Rotation

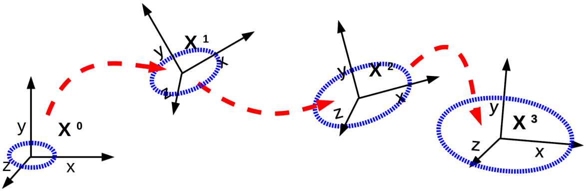


- Vehicle State X = pose (in this work)
- collection of poses = **State Vector** $\hookrightarrow$ **Trajectory**



Vehicle Localization

- Several possibilities
- Using:
  - IMU (velocity, orientation, and gravitational forces)
  - Odometry (Acoustic Sensors or Cameras)
  - Sensor Fusion
- Prone to Drift
- Visual Odometry, because Cameras
  - \+ Spatial and Temporal Resolution
  - \+ More Environmental Data
  - – Dependent on light and visibility



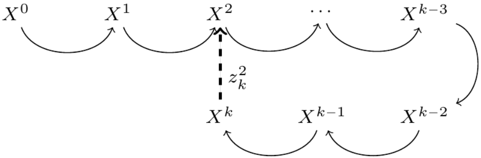
SLAM

- Visual Odometry
  - Displacement of two consecutive Images
  - **Estimation** of the Absolute Motion (Prone to drift)
- SLAM (Simultaneous Localization And Mapping)
  - Most successful approach
  - Computes pose
  - Refines pose of landmarks of environment
- Extended Kalman Filtering (EKF)

= **Visual EKF SLAM**

EKF (In a Nutshell)

- Three Stages
  1. Prediction Stage
     - Predicting vehicle's localization (visual odometry)
     - Prone to drift
     - Uncertainty is modelled with covariance matrix
  2. State Augmentation Stage
     - Prediction is added to the end of X
     - Uncertainty accumulates over time
  3. Update Stage
     - Detection of Loop Closings
     - Provide the system with more reliable Data
     - Update X



Related Work

Related Work

- Literature is scarce, but deals mainly with:
  - Correcting the odometry with the result of the Image Registration
  - Adding Landmarks to X
    + Continous Correction of pose and landmarks
    + Whole X is corrected
    - Increasing complexity over time (X gets big)
    - On-line usage no longer possible

Related Work

- [Schattschneider et al., 2011]
    - Underwater SLAM
    - Stereo Camera System used for ship hull inspection
    - 3D Landmarks used to detect Loop Closings
    - State = [poses , landmarks]

- [Eustice et al., 2008]
    - Underwater SLAM
    - State = [linear velocity, acceleration and angular rate]
    - Landmarks not saved in X
    - But: Image Registration used at every Iteration

Related Work

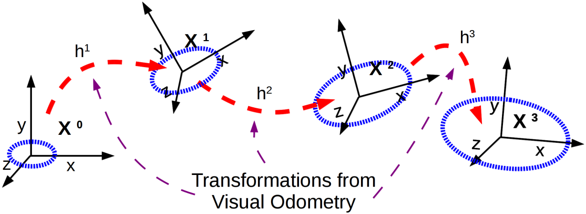
- [This study]
  - Underwater SLAM
  - Stereo Camera System
    - Obtains 3D Environmental Information
    - AUV is moving in 3D
  - $X =$ [poses]
  - Orientation is represented as a quaternion
  - Full 6 DOF Transformation
    - Different to [Burguera et al., 2014] (depth estimated by pressure sensor)
  - Jacobian Matrices of 3D Transformation
  - Application of EKF to correct the localisation

3D Transformation

3D Transformation

- Classical Transformation for 6 DOF
  - composition $\oplus$
  - inversion $\ominus$

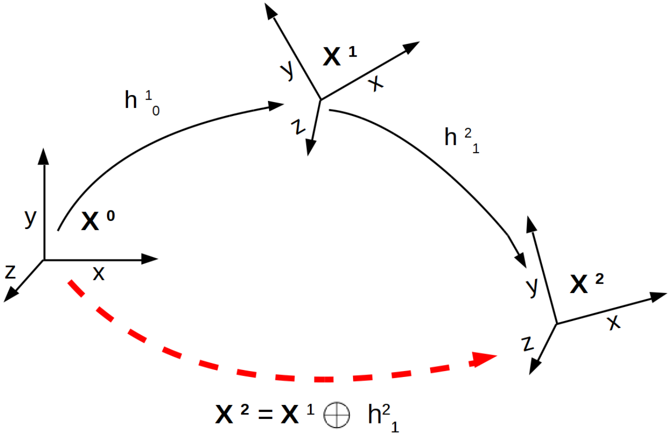
- Jacobian Matrices $J_{\oplus}$ and $J_{\ominus}$
  - Robot Transformation is non-linear
  - Direct Covariance computation in not possible
  - Approximation: Linearisation of transformation functions



Composition $\oplus$

Composition $\oplus$

- Adds a relative Transformation h to an absolute State X^x
- Result: New absolute pose $X_+ = \begin{bmatrix} X_+^t \\ X_+^r \end{bmatrix}$



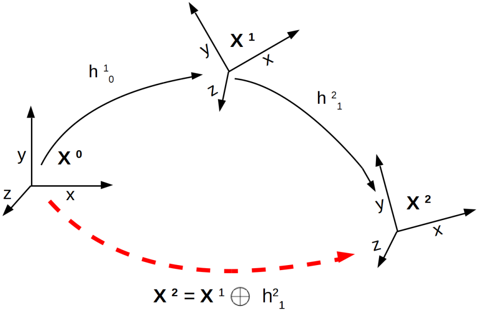
Composition $\oplus$

- Composition: $X_+ = \begin{bmatrix} X_+^t \\ X_+^r \end{bmatrix}$
- Quaternions (Orientation)
  - $q = \begin{bmatrix} q_w & q_1 & q_2 & q_3 \end{bmatrix}$
  - faster computation
  - no trigonometric functions
  - no gimbal lock
- Attention
  - Accumulation of Orientation = Multiplication of Quaternions
- $X_+^r = q^T \cdot q^P$
- Quaternion to rotation Matrix A

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

Composition $\oplus$

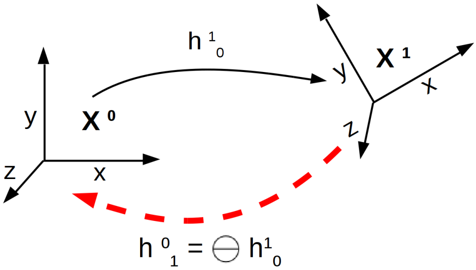
- Composition: $X_+ = \begin{bmatrix} X_+^t \\ X_+^r \end{bmatrix}$
- $X_+^t = \begin{bmatrix} x^P \\ y^P \\ z^P \\ 1 \end{bmatrix} + A^P \cdot \begin{bmatrix} x^T \\ y^T \\ z^T \\ 1 \end{bmatrix}$



Inversion $\ominus$

Inversion $\ominus$

- Inverts a Transformation h
- With $\oplus$ used to get relative Transformations from absolutes



Inversion $\ominus$

- Task: Invert $T = \left[\underbrace{x, y, z}_{t} \quad \underbrace{q_w, q_1, q_2, q_3}_{A}\right]$

$$
\begin{array}{cccc}
\vec{n} & \vec{o} & \vec{a} & \vec{p}
\end{array}
$$
$$
\begin{pmatrix} & A & & t \\ 0 & 0 & 0 & 1 \end{pmatrix}
$$

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

- $q^{-1} = \begin{bmatrix} q_w & -q_1 & -q_2 & -q_3 \end{bmatrix}$
- Result is

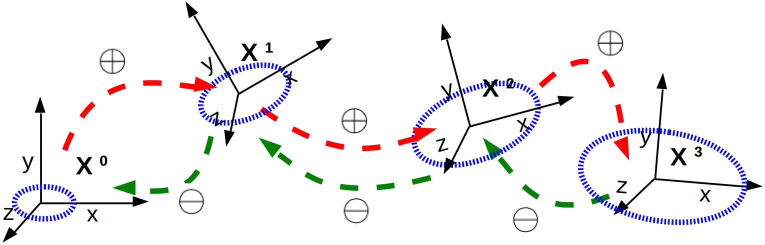
$$
\ominus X = \begin{bmatrix} -\vec{n} \circ \vec{p} \\ -\vec{o} \circ \vec{p} \\ -\vec{a} \circ \vec{p} \\ q^{-1\,T} \end{bmatrix}
$$

Jacobian Matrices $J_{1\oplus}$, $J_{2\oplus}$ and $J_{\ominus}$

- Necessary to compute the uncertainty
- Apply: Taylor Series of first order

$=$ **Covariance**: Uncertainty with zero mean random Gaussian noise

- Jacobian for each Transformation $\oplus$ and $\ominus$
- Jacobian Matrix in general
  - $\nabla f = \frac{\partial f}{\partial x}|_{\hat{x}}$



Jacobian Matrices $J_{1\oplus}$, $J_{2\oplus}$ and $J_{\ominus}$

- $J_{1\oplus}$ and $J_{2\oplus}$
  - Composition $\oplus$ has two parameters (T and P)
  - Each: Jacobian Matrix of $X_+ \hookrightarrow J_{1\oplus}$ and $J_{2\oplus}$

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

- Covariance of Composition $\oplus$:

$$C_+ = J_{1\oplus} \cdot C^1 \cdot J_{1\oplus}^T + J_{2\oplus} \cdot C^2 \cdot J_{2\oplus}^T$$

Jacobian Matrices $J_{1\oplus}$, $J_{2\oplus}$ and $J_{\ominus}$

- $J_{\ominus}$
  - Composition $\ominus$ has one parameter
  - Derivation will give us $J_{\ominus}$

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

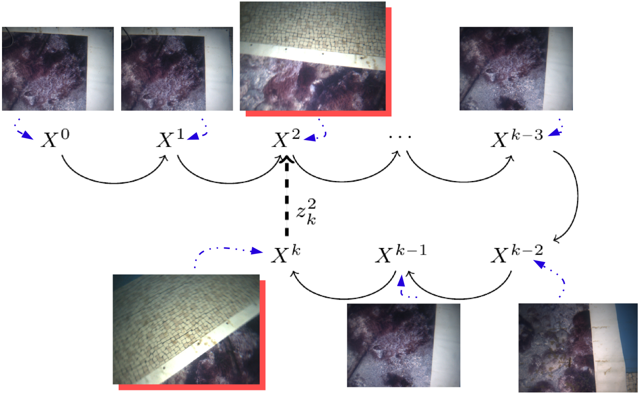
- Covariance of Inversion $\ominus$:

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

Image Registration

Image Registration

- Result: 3D camera Transformation z_k between two images
- Images have to be overlapped
- Detects Loop Closings: Update Stage (EKF)
- Without: Trajectory cannot be updated



Pseudocode

Algorithm 1: Image Registration

input : Current Stereo Image pair S_l, S_r and Recorded Images I_n

output: 3D Transformation $[R, t]$

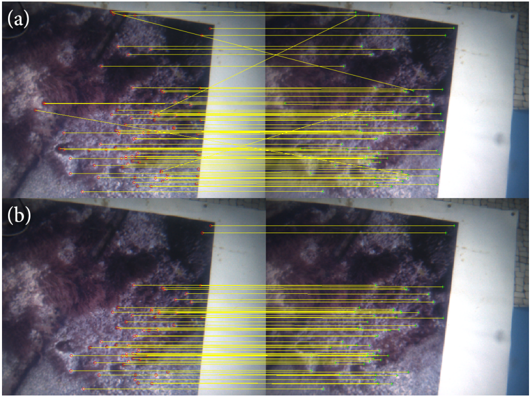
```
begin
    [F_l, F_r] ← stereoMatching (S_l, S_r);
    for I_i ∈ I_n do
        F_t ← findFeature (I_i);
        if match (F_l, F_t) == true then
            break;
        else
            continue;
    [F_l, F_r] ← updateFeature (F_l, F_r);
    P_3D ← calc3DPoints (F_l, F_r);
    [R, t] ← solvePnPRansac (F_t, P_3D)
end
```

findFeature(I_i)

- Important function
- Reliable Feature are very important
- SIFT Features
  - David G. Lowe (1999)
  - Scale invariant
  - Reliable
  - High reproducibility
  - Feature: 128 dimensional descriptor

stereoMatching(S_l, S_r)

- First: *findFeature*(I_i) with S_l, S_r
- Comparing the squared differences of each descriptor
- Differences reaches a certain treshold: Matched
- Additional: Usage of RANSAC



Pseudocode

Algorithm 1: Image Registration

input : Current Stereo Image pair S_l, S_r and Recorded Images I_n

output: 3D Transformation $[R, t]$

begin
- $[F_l, F_r] \leftarrow$ `stereoMatching` (S_l, S_r);
- **for** $I_i \in I_n$ **do**
  - $F_t \leftarrow$ `findFeature` (I_i);
  - **if** `match` $(F_l, F_t) ==$ *true* **then**
    - *break*;
  - **else**
    - *continue*;
- $[F_l, F_r] \leftarrow$ `updateFeature` (F_l, F_r);
- $P_{3D} \leftarrow$ `calc3DPoints` (F_l, F_r);
- $[R, t] \leftarrow$ `solvePnPRansac` (F_t, P_{3D})

end

$calc3DPoints(F_l, F_r)$

- Result: 3D Points
- Missing depth-value z can be calculated
  - Feature coordinates (x, y)
  - Reprojection Matrix Q

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

- C_x and C_y optical center
- f_x focal length
- $T_x = \text{baseline} \cdot f_x$
- Primed from left Camera, unprimed from right Camera

calc3DPoints(F_l, F_r)

- From 2D to 3D
- Applied for each stereo Matching

$$\begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = Q \cdot 1 \begin{bmatrix} x_l \\ y_l \\ d \\ 1 \end{bmatrix}$$

- $d = x_l - x_r$ (disparity)

$solvePnPRansac(F_t, P_{3D})$

- Solves the Perspective N-Point Problem (PnP)
- Estimates a pose transformation
- Minimizes the Reprojection Error between
  - 3D Feature
  - corresponding 2D Feature
- Result: 3D transformation $[R, t]$

$solvePnPRansac(F_t, P_{3D})$

- With respect to the Pseudocode
  - S_I is transformed into I_i (if overlap big enough)
  - Transformation is done in 3D

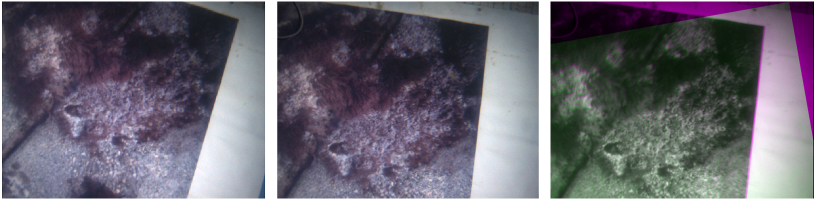


Figure: Left: S_I; middle: loop closing image I_i. On the right: the transformation of the image registration applied to S_I. The purple color indicates the error of the transformation.

EKF-SLAM

EKF-SLAM

- EKF: Bayesian filter
- State-Estimation of non-linear system
  - Using normally distributed Gaussian noise
- Three Stages
  1. Prediction Stage
  2. Stage Augmentation Stage
  3. Update Stage

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

Prediction Stage

getLastCovariance(C)

- Takes the last 7×7 Matrix of C

$$C = \begin{bmatrix} \begin{bmatrix} \sigma_{11}^1 & & & & & & \\ & \sigma_{22}^1 & & & & \alpha & \\ & & \sigma_{33}^1 & & & & \\ & & & \sigma_{44}^1 & & & \\ & & & & \sigma_{55}^1 & & \\ & \beta & & & & \sigma_{66}^1 & \\ & & & & & & \sigma_{77}^1 \end{bmatrix} & \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \ddots & \mathbf{D} \\ \mathbf{E} & \mathbf{F} & \begin{bmatrix} \sigma_{11}^n & & & & & & \\ & \sigma_{22}^n & & & & \delta & \\ & & \sigma_{33}^n & & & & \\ & & & \sigma_{44}^n & & & \\ & & & & \sigma_{55}^n & & \\ & \gamma & & & & \sigma_{66}^n & \\ & & & & & & \sigma_{77}^n \end{bmatrix} \end{bmatrix}$$

$composition(X_t, C_t, O, C_o)$

- Performs Composition $\oplus$
  - $X_+ = X_t \oplus O$
- Calculates Covariance Matrix
  - $C_+ = J_{1\oplus} \cdot C^t \cdot J_{1\oplus}^T + J_{2\oplus} \cdot C^o \cdot J_{2\oplus}^T$

Augmentation Stage

$addCovariance(C, C_t^+)$

- Not only adding (true for diagonal)

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

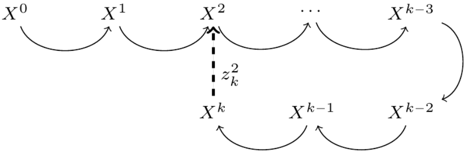
- Except for diagonal
- e.q. **B** and **E** are calculated
  - $\mathbf{B} = \mathbf{A} \cdot J_{1\oplus}^T$
  - $\mathbf{E} = J_{1\oplus} \cdot \mathbf{C}$

Update Stage

- Dependent on Image Registration
- Main part of EKF
- Executes Kalman Equations
- Corrects State Vector
- Key Features of this study:
  - Calculation of **Observation Function** h
  - Calculation of **Observation Matrix** H
  - Calculation of **Innovation** y

calcHkK(X^+, z)

- *observation function h*
  - Based on z relative motions from X are calculated
  - $h_k = \ominus X^k \oplus X^2$
  - Comparable h_k (State Vector) and z_k (Image Registration)
  - Multiple Loop Closings $h = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_k \end{bmatrix}$



$calcHkK(S_l, S_r, I_n)$

- *observation matrix H*
    - As many rows as Loop Closings (times 7)
    - As many columns as many states are stored in X^+ (times 7)
    - Stores Jacobian Matrices
    - Partially derivatives of h with respect to X^+

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

- Elements of H not referring to used states are 0

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

innovation(h, z)

- Innovation should be big if z and h are different
- Innovation should be 0 if z and h are similar
- In general: Difference between h and z
  - $y = z - h$
- Translation: subtraction
- Due to quaternions special treatment necessary

innovation(h, z)

- Different quaternions - similar orientation
- $q_z = [0.996, -0.010, 0.014, 0.083]$ $(1.55^\circ,\ -1.38^\circ,\ 9.50^\circ)$
- $q_h = [-0.996, -0.018, 0.001, -0.083]$ $(0.04^\circ,\ 2.09^\circ,\ 9.55^\circ)$
- $y_q = q_z - q_h = [1.992, 0.007, 0.013, 0.166]$
- Solution: Absolute values
- $y_q = |q_z| - |q_h|$
- $y_q = [0.0000, -0.0073, 0.0134, -0.0003]$

Pseudocode

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

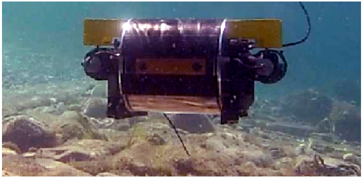
; `/* Prediction stage */`
$X_t \leftarrow$ `getLastState` (X) ;
$C_t \leftarrow$ `getLastCovariance` (C) ;
$[X_t^+, C_t^+] \leftarrow$ `composition` (X_t, C_t, O, C_o) ;
; `/* Augmentation stage */`
$X^+ \leftarrow$ `addState` (X, X_t^+);
$C^+ \leftarrow$ `addCovariance` (C, C_t^+);
; `/* Update stage */`
$z \leftarrow$ `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

Results

Results

- System Used
  - Laptop (Intel core i7 (2 2.9Ghz), 8GB RAM and SSD)
  - Ubuntu 12.04
  - MATLAB R2013a (single CPU core used)
  - The mission was recorded with ROS (Robot Operation System)
  - rosbag provided offline playback (recorded with Fugu-C)
- Set-Up
  - Fugu-C (Bumblebee 2 1032 $\times$ 776 pixel)
  - Watertank inside the UIB (7m $\times$ 4m $\times$ 1.5m)
  - Visual Odometry calculated with LIBVISO2
- Ground Truth: Seabed printed on a Poster



Results

- Test
  - 23.42m sweeping task
  - Different noise levels
  - System-Response to less accurate Odometry

Noise Level	*1*	*2*	*3*	*4*	*5*	*6*
Covariance	*0*	*3e-9*	*9e-9*	*3e-8*	*5e-7*	*3e-6*

- Error Definition:
  - Difference between Ground Truth
    - odometry
    - EKF-SLAM
  - Divided by the length of the Trajectory
  - Error units are meters per travelled meter

Results

- Quantitative Results

Noise Level *Covariance*	*1* *0*	*2* *3e-9*	*3* *9e-9*	*4* *3e-8*	*5* *5e-7*	*6* *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

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

Results

- Quantitative Results

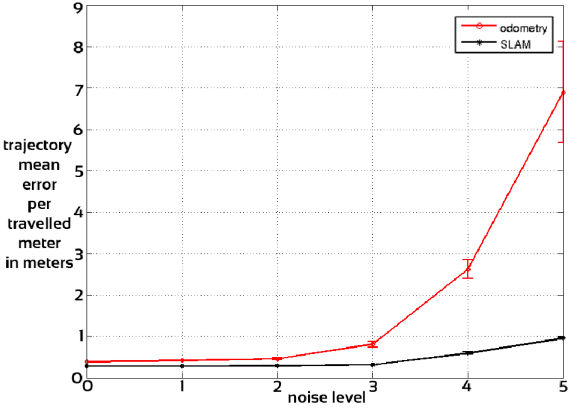


Figure: Comparison between state mean errors using raw odometry and EKF pose estimates. The standard deviation is set to 0.1σ.

Results

- Quantitative Results

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

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

- Separation of 4 already faster than Mission-Time

Results

- Qualitative Results Blue: Ground Truth, Black: Odometry, Red: EKF-SLAM

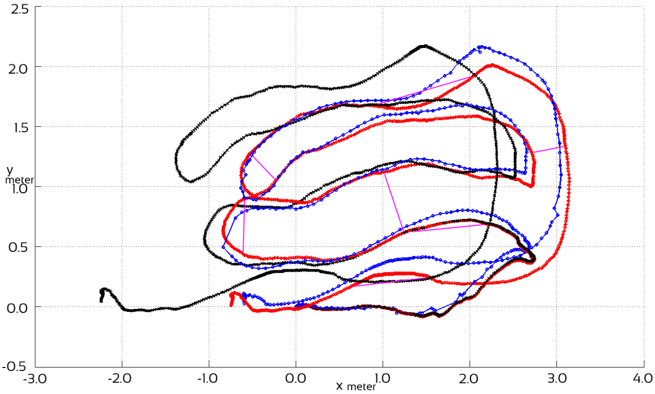


Figure: Example result with a noise level of two. Additionally the eight loop closings are plotted (magenta lines).

Results

- Qualitative Results Blue: Ground Truth, Black: Odometry, Red: EKF-SLAM

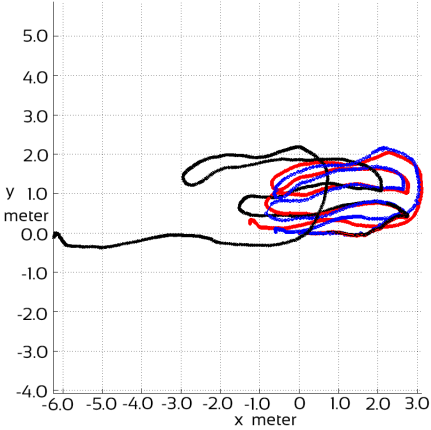


Figure: Example result with a noise level of four.

Results

- Qualitative Results Blue: Ground Truth, Black: Odometry, Red: EKF-SLAM

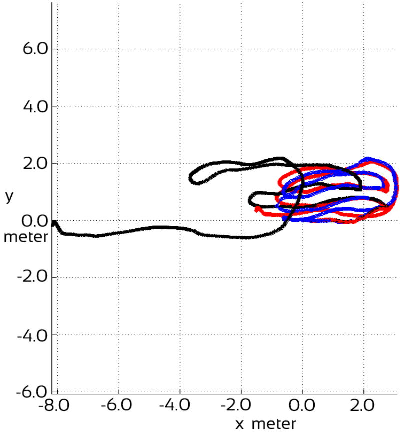


Figure: Example result with a noise level of six.

Results

- Video
  - Separation of 2
  - Noise Level of 3
  - Playback 4x

Conclusion

Conclusion

- Summary
  - Pose based visual EKF-SLAM approach
  - Underwater localization
  - Only Stereo Camera Data
  - Pure 6 Degrees of Freedom
  - Orientation is represented as quaternions
  - Improvement up to 86.1%
  - With Separation of 4 already Execution-Time under Mission-Time
- Future Work
  - Fine-Tuning: Measurement and Observation Covariance Matrices
  - Further test in the sea
  - ROS implementation

Literature I

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.

Eustice, R. M., Pizarro, O., and Singh, H. (2008).
Visually Augmented Navigation for Autonomous Underwater Vehicles.
Ieee Journal Oceanic Engineering, 33:103–122.

Schattschneider, R., Maurino, G., and Wang, W. (2011).
Towards stereo vision SLAM based pose estimation for ship hull inspection.
Oceans 2011, pages 1–8.

Autonomous Underwater Vehicles (AUVs)

- Remotely Operated Vehicles (ROVs)
  - Tethered
  - Support Vessels
  - Limited operative range
- Autonomous Underwater Vehicles
  - (Try to) Overcome this limitations
  - Highly repetitive, long or hazardous missions
  - Self-Powered
  - Independent (support ships and weather)
  - Reduction of
    - missions costs
    - human resources
    - execution time

Applications

- Maintenance
- Rescue Operations
- Surveying
- Infrastructure Inspections
- Sampling

getLastState(X)

- Takes the last 7 Elements of X

$$X = \Big[\underbrace{x^1 \;\; y^1 \;\; z^1 \;\; q_w^1 \;\; q_1^1 \;\; q_2^1 \;\; q_3^1}_{\text{vehicle pose at } 1^{\text{st}} \text{ iteration}} \;\cdots\; \underbrace{x^n \;\; y^n \;\; z^n \;\; q_w^n \;\; q_1^n \;\; q_2^n \;\; q_3^n}_{\text{vehicle pose at } n^{\text{th}} \text{ iteration}}\Big]^T$$

innovation(h, z)

- $y_q = q_z - q_h = [1.99274, 0.007344, 0.013427, 0.166257]$
- Pure subtraction: Big innovation (not right!)
- Solution: Absolute values
- $y_q = |q_z| - |q_h|$
- $y_q = [0.0000, -0.0073, 0.0134, -0.0003]$

innovationCov$(C^+, H\ C_m)$

- $S = H \cdot C \cdot H^T + R$
- *Measurement Matrix R*
- Size of R depends on number of detected Loop Closings

$$R = \begin{bmatrix} C_m & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & C_m & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & C_m \end{bmatrix} \tag{1}$$