Maritime radar odometry inspired by visual odometry

Henrik D. Flemmen *, Rudolf Mester †, Annette Stahl *, Torleiv H. Bryne *, Edmund Førland Brekke *
* Department of Engineering Cybernetics
† Department of Informatics and computer science
Norwegian University of Science and Technology

***Abstract*—Future autonomous ships will need several redundant positioning systems to navigate reliably. Global Navigation Satellite Systems are highly accurate but they are susceptible to disruptions and intentional jamming. Maritime radars have long range and are robust against bad weather and darkness, but the use for ownship motion estimation has received relatively little attention in the research field. In this work, we present a radar odometry estimation method inspired by advances in visual odometry and simultaneous localization and mapping. The method works on raw radar data in a coastal environment and combines the Kanade-Lucas-Tomashi tracker with a factor graph back-end. We test it on data from a large ship with a maritime radar with a range of 19 km. We find that it is robust with only a small drift and no erroneous jumps in the estimate.**

***Index Terms*—radar, odometry, maritime, navigation, positioning**

I. Introduction

Emerging prototypes for autonomous ships rely heavily on Global Navigation Satellite Systems (GNSS) for positioning. GNSS provides highly accurate positioning and has high integrity but may, in rare cases, be unavailable. Due to the weak signal power from orbit, GNSS systems are highly susceptible to both intentional jamming and unintentional disruptions [1]. To enable safe autonomous ships, we need a backup positioning system.

Several positioning systems can serve as potential redundancy for GNSS, including Long range navigation (E.g. E-Loran/Loran-C), dead reckoning, and Simultaneous Localization And Mapping (SLAM). E-Loran/Loran-C systems are less susceptible to jamming than GNSS, but the ship still needs to rely on external cooperative infrastructure. Traditional dead reckoning based on compass and log is an established fallback in manned shipping. Similar to Inertial Navigation Systems (INS), they provide a robust and reliable position estimate, but the accuracy will deteriorate with time. Without a prior map, SLAM and SLAM-like odometry will be a dead reckoning type of position estimate. It is, therefore, also subject to drift over time, but it may be smaller than for the two aforementioned dead reckoning methods.

This work is supported by the Norwegian Research Council through the project Center for Research-based Innovation Autoship, project number 309230. The data is provided by Kongsberg Maritime. Corresponding author: henrik.d.flemmen@ntnu.no

A. Maritime radar SLAM/odometry

SLAM and SLAM-like odometry have received great attention and interest from the robotics community, particularly using cameras and lidar [2, 3]. Ships typically have a maritime radar, which compared to camera/lidar offers a longer range and better robustness against fog, rain, and darkness. This motivates us to choose maritime radar as our sensor for odometry estimation.

Early work on radar SLAM based on techniques from computer vision include [4], which developed a SLAM framework based on the Scale Invariant Feature Transform (SIFT). They use the tracks from the SIFT features in an "Exactly Sparse Delayed-state Filter" which reduces the distortion of the radar scans by dividing them into smaller sectors with individual timestamps.

In parallel, [5] developed a SLAM framework based on point features. They use the point features in a filter based on Random Finite Sets-theory, which probabilistically associates the detected points to their previous observations.

Using the coastline as a stable feature, [6] and [1] developed first a localization scheme in prebuilt maps and later a SLAM framework. They fuse the coastline tracks with measurements from an Inertial Measurement Unit (IMU) in an Extended Kalman Filter (EKF).

Line features are extracted with traditional computer vision techniques in [7] and found again in consecutive radar images using the Hough transform.

Recently, [8] implemented a radar odometry system based on Oriented fast and Rotated Brief (ORB) features. For estimation, they use an estimation scheme similar to a sliding window least squares estimator. The difference is that they only optimize over the youngest pose in the window and keep the others fixed.

Except for [8], all the methods mentioned above are made for relatively small watercraft, either a patrol craft[4], Unmanned Surface Vehicle (USV)([5, 6, 1]) or Rigid Inflatable Boat (RIB)([7]). The work of [8] describes a ferry where the radar is mounted approximately 10 m above the water line. [5, 6, 1] use auxiliary measurements from other sensors to supplement the radar in their algorithm. Most of them either employ a filter-type estimator or are not explicit about their estimation step, with the notable single exception of [8].

B. Progress in other fields

Recent years have seen a great interest in visual- and visual-inertial navigation. Several recent methods in visual (inertial) SLAM/visual odometry ([9, 10, 11, 12]) uses the established Kanade-Lucas-Tomashi (KLT) feature tracker, which boasts highly accurate feature tracks.

Furthermore, factor graph techniques such as incremental Smoothing And Mapping 2 (ISAM2)[13] are often used in state-of-the-art visual-inertial odometry [10] and lidar inertial odometry as [14].

C. Contribution and outline

Inspired by the progress in SLAM/odometry in the robotics community, we ask ourselves whether the combination of KLT and factor graph optimization can be used to solve maritime radar SLAM. As a first step towards this, we develop and odometry pipeline that combines KLT and ISAM2. The implementation is tested on data collected from a larger ship than earlier works with a radar height of over 20 m. For the benefit of the community, we release the source code as open source. [1]

In the following, section II describes the sensor model and assumptions we use and introduces the notation. Section III describes our proposed system's feature tracking and estimation. We then outline the data we tested on and compare the results in section IV. Finally, we briefly discuss the findings section V before we conclude in section VI.

II. Sensor model

A. Notation and convention

A 2D point is represented by

$$\mathbf{p} \in \mathbb{R}^2 = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix}. \tag{1}$$

Here we define 2D transformations with both rotation ($\mathbf{R}_b^a \in SO(2)$) and a new origo as an abstract object where transforming a point is an action of the object on the point. They can also be composed to form the cascaded transformation:

$$\mathbf{p}^a = \mathbf{R}_b^a \mathbf{p}^b \tag{2a}$$
$$\mathbf{T}_b^a \in SE(2) = [\mathbf{R}_b^a \in SO(2), \mathbf{p}_{ab}^a \in \mathbb{R}^2] \tag{2b}$$
$$\mathbf{p}^a = \mathbf{T}_b^a \mathbf{p}^b = \mathbf{R}_b^a \mathbf{p}^b + \mathbf{p}_{ab}^a \tag{2c}$$
$$\mathbf{T}_c^a = \mathbf{T}_b^a \mathbf{T}_c^b, \tag{2d}$$

where $SO(2)$ is the special orthogonal group of 2D rotations, and $SE(2)$ is the special eucledian group of 2D rotations and translation.

[1] https://github.com/hflemmen/radar_odometry

1) Coordinate frames: The convention for coordinate frames in this document is as given below.

$\mathbf{q}^i = \begin{bmatrix} r & \theta \end{bmatrix}^T$**:** Point in the radar frame at time i in polar coordinates.

$\mathbf{s}^i = \begin{bmatrix} d & a \end{bmatrix}^T$**:** The radar image coordinates for the polar image at time i.

$\mathbf{p}^i = \begin{bmatrix} p_1 & p_2 \end{bmatrix}^T$ The world frame coordinate at time i.

B. Sensor model

The radars considered here are mechanically scanning radars. They have a directional antenna that rotates continuously while simultaneously transmitting a pulsed signal and receiving the echo with a small delay. The radar measures both the time between the emission of the pulse and the reception of its backscatter and the power level of the backscatter. The returned power will be noise plus the emitted signal returned at different times, corresponding to noisy backscatter from different distances. The measured returned power for each distance is stacked into a vector and constitutes one row of the radar image. The rest of the rows are generated by repeating this process for a fixed set of angles in the scan.

The radial coordinate of a radar image linearly relates to the distance away from the sensor. If we define the constant to K_{mps} where mps is meters per sample, the resulting distance becomes:

$$d = D(r) = \frac{1}{K_{mps}} r. \tag{3}$$

Likewise, the angular coordinate is a scaling from radians to pixel widths:

$$a = \frac{a_{\max}}{2\pi}\theta, \tag{4}$$

where $a_{\max}$ is the number of rows in the radar image.

Each row in the radar image is captured at slightly different timestamps due to the rotating nature of the sensor. The timestamp t_a for each row in a scan started at t_k will be

$$t_a = t_k + \frac{\tau_a}{2\pi}\theta \tag{5}$$

where τ_a is the duration of a single radar scan.

The time offset from the right summand in eq. (5) complicates the model and the estimation by introducing a time dependency between the discrete radar images. The simplest possible solution is to neglect it such that

$$t_a \approx t_k \tag{6}$$

Using the simplification from (6) we define a similar notation for the pose:

$\mathbf{T}_w^{b(t)}$**:** The pose(body frame expressed in the world frame) of the ship at the arbitrary time t.

$\mathbf{T}_w^k$**:** The pose of the ship at the scans which starts at time $t_k = k \cdot \tau_a$. I.e. the k'th scan to be received.

I.e.

$$\mathbf{T}_w^k = \mathbf{T}_w^{b(t_k)}. \tag{7}$$

We divide the radar model in two: the transformation from Cartesian coordinates (Λ) and the scaling to image coordinates (Ψ):

$$\mathbf{s} = \begin{bmatrix} d \\ a \end{bmatrix} = \Psi(\mathbf{q}) = \begin{bmatrix} D(r) \\ \frac{a_{\max}}{2\pi}\theta \end{bmatrix} \tag{8a}$$

$$\mathbf{q} = \Lambda\left(\mathbf{p}^b, w_r, w_\theta\right) = \begin{bmatrix} \sqrt{p_1^2 + p_2^2} \\ \operatorname{atan2}(p_2, p_1) \end{bmatrix} + \begin{bmatrix} w_r \\ w_\theta \end{bmatrix} \tag{8b}$$

$$w_r \sim \mathcal{N}\left(0, \sigma_r^2\right) \tag{8c}$$

$$w_\theta \sim \mathcal{N}\left(0, \sigma_\theta^2\right) \tag{8d}$$

where

$\Lambda : \mathbb{R}^2 \to \mathbb{R}^2$**:** Conversion between Cartesian coordinates and polar coordinates.

$\Psi : \mathbb{R}^2 \to \mathbb{R}^2$**:** Scaling from metric polar coordinates to image coordinates.

$\sigma_r^2, \sigma_\theta^2$**:** The measurement covariance of the sensor in the radial and tangential directions.

The measurement equation then becomes:

$$\mathbf{s} = h\left(\mathbf{T}_w^{b(t_a)}, \mathbf{p}^w\right) + \mathbf{w}_h = \Psi\left(\Lambda\left(\mathbf{T}_w^{b(t_a)} \cdot \mathbf{p}^w\right)\right) + \mathbf{w}_h \tag{9a}$$

$$\mathbf{w}_h = \begin{bmatrix} \frac{1}{K_{mps}} w_r \\ \frac{a_{\max}}{2\pi} w_\theta \end{bmatrix}, \tag{9b}$$

where $\mathbf{T}_w^{b(t_a)} \in SE(2)$ now is the pose of the ship at the time the radar hits the point $\mathbf{p}^w$.

To fit this estimation problem into discrete frames, we use the simplification from (6) to associate the scan to only a single timestamp:

$$\mathbf{s} = h\left(\mathbf{T}_w^k, \mathbf{p}^w\right) + \mathbf{w}_h = \Psi\left(\Lambda\left(\mathbf{T}_w^k \cdot \mathbf{p}^w\right)\right) + \mathbf{w}_h \tag{10a}$$

$$\mathbf{w}_h = \begin{bmatrix} \frac{1}{K_{mps}} w_r \\ \frac{a_{\max}}{2\pi} w_\theta \end{bmatrix}. \tag{10b}$$

The only differences between (10) and (9) is that we use $\mathbf{T}_w^k$ instead of $\mathbf{T}_w^{b(t_a)}$.

We cannot measure the global location with only relative measurements, but we define our world frame to be the body frame at the timestep of the first radar scan such that

$$\mathbf{T}_w^1 = \mathbb{I}. \tag{11}$$

C. Deviations from the model

This radar model is a simplification and some effects specific to radars are omitted. Most notably, we do not model shadowing effects. Islands close to the radar may occlude the islands behind and the far end of the island. E.g. in Fig. 1, there are some dark areas on the islands, which likely are occluded parts of the island. This impacts the shape of the islands in the radar image. This work neglects this impact and treats it as noise and general outliers.

This work assumes that each scattering real-world point has the same RCS observed from different angles. This is one of the underlying assumptions from KLT, described further in Sec. III-D. This is not theoretically the case, but it is typically not far from it either. The shape of the target influences RCS, which may change with changing direction of observation.

To simplify the model, we assume that all returns are stationary land. This will fail for several reasons, but most commonly, other ships and sea clutter. Other ship's returns will move as the ship moves, and sea clutter is returns from the waves at sea. In addition, interference from other ships' radars introduces very short bursts of very high power on our radar. This is visible in Fig. 1 as horizontal lines.

Fig. 1. An example radar image in polar coordinates. The left side is close to the radar, and the right side is far from the radar. The y-axis is the direction, with the top and bottom representing straight ahead. The bright areas are islands around the ship.

III. System description

A. Overview

This system is heavily inspired by visual odometry and visual SLAM and leverages established methods from visual odometry and visual SLAM. We conceptually separate the system into four parts: Preprocessing, feature detection, feature tracking, and estimation, each corresponding to its subsection below.

Fig. III-A shows the data flow in the algorithm. It starts from the top for each new full rotation of the radar and ends with a new estimate of the trajectory.

B. Preprocessing

We do two tricks with the radar images to reduce errors in the feature detection and feature tracking modules. We employ an interference reduction filter to remove the high power disturbances from nearby radars. It works by comparing the

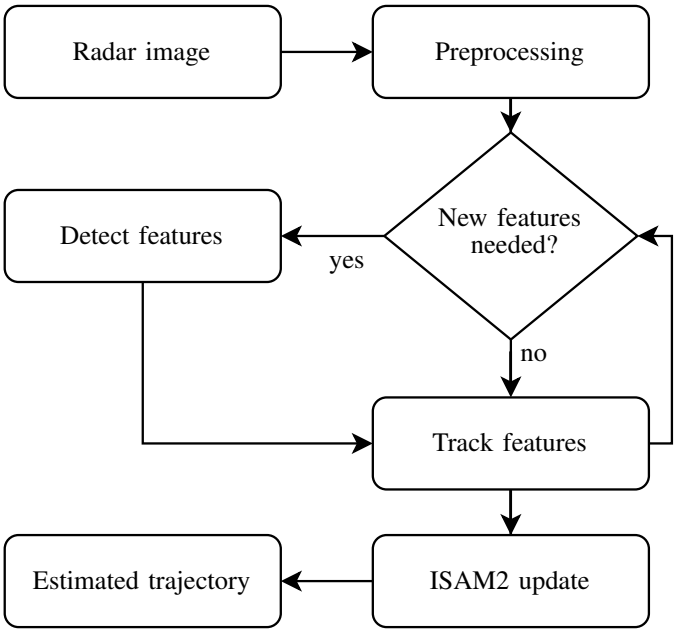

Fig. 2. The data flow of the algorithm. The two blocks "Track features" and "ISAM2 update" store their results for use with the next radar frame.

return strength of each pixel with the pixel above and below. If the middle pixel is significantly higher, it is considered a disturbance from another radar and set to the average of the neighboring pixels.

The second trick is to remove weak returns under a fixed threshold. Weak returns may still produce strong gradients but will more often than other strong gradients originate from sea clutter. The downside is that this would also leave out potentially useful features, so we tune it to a level that still leaves some sea clutter and most land features. In our implementation, which works with 8-bit color depth, we set the threshold to 10.

C. Feature detection

Inspired by visual feature detection and tracking, we use the Shi-Tomashi score[15] to find features suitable for KLT-tracking[16].

We store the coordinates of the features, which makes the set of newly found features in frame k: $\mathcal{S}_k^f$. Together with the set of features tracked from the previous frame $\mathcal{S}_k^t$, we get the set of features in frame k, $\mathcal{S}_k$:

$$\mathcal{S}_k^f = \left\{ \mathbf{s}_i^k \in \mathbb{R}^2 \quad \forall i \right\} \tag{12a}$$

$$\mathcal{S}_k = \mathcal{S}_k^f \cup \mathcal{S}_k^t. \tag{12b}$$

All detected points are added to the map by using the pose of the radar at the time of detection.

$$\mathcal{P}_k^f = \left\{ \mathbf{p}_m^w = \left(\mathbf{T}_w^k\right)^{-1} \Lambda^{-1} \left(\Psi^{-1}\left(\mathbf{s}_i^k\right)\right) \forall \mathbf{s}_i^k \in \mathcal{S}_k^f \right\} \tag{13a}$$

$$\mathcal{P}_k = \mathcal{P}_{k-1} \cup \mathcal{P}_k^f. \tag{13b}$$

D. Feature tracking

We track each feature from the preceding frame to the current frame. I.e. we extracts a new patch at each feature location in every frame, which it tracks to the next frame. Each feature is then tracked back again from the current frame to the preceding frame to validate the track. If the track back again ends up in a different location than where it started from, the track is discarded.

The tracking gives a correspondence between the features in the previous frame such that

$$\mathcal{S}_{k+1}^t = \left\{ \mathbf{s}_i^{k+1} \leftarrow \mathbf{s}_i^k \quad \forall \mathbf{s}_i^k \in \mathcal{S}_k \setminus \mathcal{S}_{k,\text{failed}} \right\}, \tag{14}$$

where $\mathcal{S}_{k,\text{failed}}$ are the features that failed tracking this frame.

We do the feature tracking in image coordinates, but we use the linear part of the sensor model to convert them to radian x metric coordinates before using them in the estimation.

$$\mathcal{Q}_k = \left\{ q_i^k = \Psi^{-1}\left(s_i^k\right) \forall s_i^k \in \mathcal{S}_k \right\}. \tag{15}$$

E. Feature management

We use two tunable thresholds in the feature management, the minimum and maximum number of features. If the number of successfully tracked features goes below the minimum, new features are found until either we reach the maximum or there are no feature candidates with a sufficiently high Shi-Tomashi score left. Between the frames we add features, the number of features monotonically decreases by removing all features that fail tracking once.

There are two reasons for features to be discarded:

- The texture in the feature becomes too small. I.e. it becomes difficult to track due to a "flat" patch.
- The KLT backtrack from the most recent frame to the previous frame ends up too far away from the original point.

F. Estimation

The motion estimate is based on a factor graph model, which is incrementally solved for each new frame [13].

For frame a we optimize this cost function:

$$\min_{\mathcal{T}_w^a, \mathcal{P}_a} \sum_{k \in 1 \ldots a} \sum_{m \in \mathcal{M}_k} \rho\left(\left\| E\left(\mathbf{T}_w^k, \mathbf{q}_m^k, \mathbf{p}_m^w\right) \right\|_{\Sigma^{-1}} \right), \tag{16}$$

where

$\mathcal{T}_w^a$**:** The trajectory, i.e., the poses of at all timesteps up to scan a: $\mathbf{T}_w^1, \ldots, \mathbf{T}_w^a$.

$\mathcal{P}_a$**:** The set of all landmarks observed up until frame a.

$\mathcal{M}_k$**:** The index set of the landmarks which is observed in frame k.

ρ**:** The Huber norm.

$\mathbf{q}_m^k \in \mathcal{Q}_k$**:** The observation of landmark m from frame k in polar coordinates.

$\mathbf{p}_m^w \in \mathcal{P}_k$ **:** The map location of landmark m expressed in NED.

Σ**:** The covariance matrix for E.

The residual $E(.)$ is given by

$$E(\mathbf{T}_w^k, \mathbf{q}_m^k, \mathbf{p}_m^w) = \Lambda(\mathbf{T}_w^k \cdot \mathbf{p}_m^w) - \mathbf{q}_m^k, \tag{17}$$

where

$\mathbf{T}_w^k \in SE(2)$**:** The transformation between frame k and the world frame.

$\mathbf{q}_m^k \in \mathbb{R}^2$**:** The observation of point $\mathbf{p}_m^w$ measured in frame k in polar coordinates.

$\mathbf{p}_m^w \in \mathbb{R}^2$**:** The coordinates of the landmark m in world coordinates.

$\Lambda : \mathbb{R}^2 \to \mathbb{R}^2$**:** The transformation between the radar image pixel coordinates and their real world geometrical location relative to the ship, given by the geometrical sensor model.

The covariance matrix, Σ, is diagonal and given as

$$\Sigma = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\theta^2 \end{bmatrix}, \tag{18}$$

where σ_r^2 and σ_θ^2 are the covariances in the radial and tangential directions. We typically do not have an analytical way of obtaining these covariances, so they will have to be tuned based on the data available.

It is worth noting that we don't have any between factors, motion models, or any other way of constraining the optimization problem with dynamics. It is plausible that a good motion model would improve accuracy, but now we are not constrained to any type of motion.

The underlying assumption that all the radar returns are stationary is violated in several cases, most commonly by other ships in transit and sea clutter. These outliers would have degraded performance considerably if not handled. We could have tried to detect and prune such cases explicitly, but instead, we use the Huber norm in the cost function to be robust to these outliers.

G. Implementation notes

For KLT-tracking and the feature detection we use the opencv-implementation [2].

The estimation is based on the library GTSAM [17] and the build-in constructor `BearingRangeFactor2D`. In frame k for landmark m we give it the keys to the pose $\mathbf{T}_w^k$ and the landmark $\mathbf{p}_m^w$, the measured distance $\mathbf{q}_m^k$ and the constant covariances σ_r^2 and σ_θ^2.

To do arithmetic on lie groups we use the Manif-library [18].

The initialization of the optimization is following two principles:

1) New poses are initialized on top of the most recent pose and new landmarks are added at their measured position, relative to the pose that observed it.
2) The first pose is the identity pose, which corresponds to defining the first frame as the world frame.

The problem appears to be sufficiently convex for this to be adequate.

[2]https://opencv.org/

TABLE I
SPECIFICATIONS FOR THE DIFFERENT RADARS USED.

	Radar0	Radar1	Radar2
Band	S-band	S-band	X-band
Height above sea	21 m	27 m	24 m
Beamwidth	3.1°	3.1°	1.3°
Meters per range bin	3.4067	3.4067	3.4067
Recorded range	19 km	19 km	19 km
Rotation rate	ca. 2.4 Hz^{-1}	ca. 2.3 Hz^{-1}	2.1 Hz^{-1}

TABLE II
DIFFERENCES BETWEEN THE DIFFERENT DATASETS USED. SEA CLUTTER AND THE NUMBER OF ISLETS ARE QUALITATIVELY EVALUATED BY VISUALLY INSPECTING THE RADAR IMAGES AND ARE RELATIVE TO THE OTHER SEQUENCES.

Dataset	Radar	Length [mm:ss]	Sea clutter	Islets	Movement
a	Radar1	17:43	Much	Many	Small S-turn
b	Radar0	18:03	Much	None	Several turns
c	Radar1	19:41	Little	None	Turn
d	Radar0	18:03	Much	Many	Small turn
e	Radar0	18:03	Much	Some	Small turn
f	Radar2	17:04	Little	Many	S-turn
g	Radar0	19:03	Much	Many	Turn
h	Radar0	18:31	Little	None	Straight
i	Radar1	17:33	Some	None	S-turn

IV. RESULTS

We have evaluated our algorithm and Yeti [19] on recorded radar data provided by Kongsberg Maritime. It consists of 9 sequences of several minutes of simultaneous recording of radar scans and GNSS for ground truth. Each sequence uses only one radar, but the different sequences use one of 3 different radars described in Table I. Table II shortly describes the datasets. Some of the datasets are in environments with many small islets, while others are in environments that are dominated by larger continuous islands and land. There are also considerable variations in the amount of sea clutter visible in the different images. Neither of these effects are easy to describe quantitatively, but Table II gives the author's subjective qualitative assessment of the relative amount of sea clutter and frequency of islets. To our knowledge, there are no open-source maritime radar odometry algorithms, so we compared our implementation against one from the automobile community. In their work with [19], the authors developed Yeti[3]. To test the impact of neglecting the continuous motion of scanning radars, they developed two variants of an ORB-based odometry estimation scheme. One compensated for the motion of the radar, and one did not. They tested it on automobile radar data and found that compensating for the radar motion gave considerably better accuracy. They later developed HERO[20], which is also open-source. HERO is built on a deep learning-based feature extractor. It is pre-trained on automobile radar images, and we don't have enough data to retrain their artificial neural networks for maritime radar

[3]https://github.com/keenan-burnett/Yeti_radar_odometry

TABLE III
THE RMSE OF THE TRANSLATIONAL RPE FOR THE DIFFERENT SEQUENCES. THE VALUE IS IN METERS AND THE RPE INTERVAL IS 5 FRAMES.

Dataset	Ours	Yeti
a	3.412	26.969
b	5.907	24.57
c	4.255	6.681
d	2.852	11.515
e	3.029	8.345
f	2.217	9.617
g	3.405	9.594
h	5.298	236.381
i	4.304	12.242

data. We, therefore, try to compare our method against Yeti. We use their implementation of motion-compensated Random Sample Consensus (RANSAC), which they concluded was the preferred alternative. We have tried tuning it for the maritime scenario with the different implemented feature detectors, but the cen2018[21] detector also works best here. To make a more fair comparison, we preprocess the data for yeti by removing weak returns and interference from other radars as described in Sec. III-B.

A. Evaluation metrics

We evaluate both the Relative Pose Error (RPE) and Absolute Trajectory Error (ATE) [22] for the different variations of the algorithm. They are commonly formulated for 3D, but in this work, we reformulate them for 2D:

$$\mathbf{T}_{\text{rpe},i} = \left(\left(\mathbf{T}_i^{w,\text{gt}}\right)^{-1}\mathbf{T}_{i+\Delta}^{w,\text{gt}}\right)^{-1}\left(\left(\mathbf{T}_i^{w}\right)^{-1}\mathbf{T}_{i+\Delta}^{w}\right) \tag{19a}$$

$$\mathbf{T}_{\text{ate},i} = \left(\mathbf{T}_i^{w,\text{gt}}\right)^{-1}\mathbf{T}_w^{w,\text{gt}}\mathbf{T}_i^{w}, \tag{19b}$$

where $\mathbf{T}_w^{w,\text{gt}}$ is the transformation between the first frame in the estimated sequence and the first frame in the ground truth sequence. Δ is the RPE-interval, which determines how long durations the RPE considers.

To obtain a single number to compare we can calculate the Root Mean Square Error (RMSE) or the maximum value for the sequence. The magnitude of the pose does not have a meaningful physical unit, so we separate it into the Euclidean distance of the translation and the angle:

$$E_{RMSE,e,ROT} = \sum_{i=0}^{N} \|ROT(\mathbf{T}_{e,i})\|^2 \tag{20a}$$

$$E_{RMSE,e,TRANS} = \sum_{i=0}^{N} \|TRANS(\mathbf{T}_{e,i})\|^2 \tag{20b}$$

$$E_{MAX,e,ROT} = \max \|ROT(\mathbf{T}_{e,i})\| \tag{20c}$$

$$E_{MAX,e,TRANS} = \max \|TRANS(\mathbf{T}_{e,i})\|, \tag{20d}$$

where N is the number of frames for the whole sequence and e is either rpe or ate. ROT and $TRANS$ are functions that return the angle or the NED position of the pose.

Tables III and IV report the RMSE RPE on the different datasets. They show that Yeti generally have higher drift, both in translation and rotation. RPE is in some sense more relevant than ATE to measure drift for odometry methods, but it may obscure large jumps in the estimate and the accumulated drift. Therefore we also present the maximum ATE in Table V and Table VI. This is equivalent to the largest deviation from the ground truth during the sequence and will typically increase for long sequences. Table V shows that our method has a lower maximum deviation from ground truth in the orientation, but the results for translation are mixed. Yeti has a lower maximum error on datasets c and d, while our method has a lower maximum error in the others. Dataset h has a particularly large difference in both orientation and translation. If we consider Fig. 3, we can see that most of the error arise from 3 large sudden jumps in the estimate. This is likely due to faulty associations and will, for many controllers, be worse to handle in a feedback system than slower accumulation of error. Our implementation does not have these jumps and therefore manages a far lower accumulated error. Considering a case where Yeti had a lower maximum error than our method, e.g.

TABLE IV
THE RMSE OF THE ROTATIONAL RPE FOR THE DIFFERENT SEQUENCES. THE VALUE IS IN DEGREES AND THE RPE INTERVAL IS 5 FRAMES.

Dataset	Ours	Yeti
a	0.165	0.579
b	0.172	0.504
c	0.16	0.287
d	0.152	0.273
e	0.121	0.212
f	0.165	0.209
g	0.187	0.31
h	0.13	0.924
i	0.122	0.313

TABLE V
THE MAXIMUM OF THE ROTATIONAL ATE FOR THE DIFFERENT SEQUENCES. THE VALUE IS IN DEGREES.

Dataset	Ours	Yeti
a	0.213	1.026
b	2.961	3.432
c	0.434	4.042
d	0.689	1.447
e	0.25	1.15
f	0.485	1.15
g	0.893	0.986
h	0.251	5.711
i	1.795	3.254

TABLE VI
THE MAXIMUM OF THE TRANSLATIONAL ATE FOR THE DIFFERENT SEQUENCES. THE VALUE IS IN METERS.

Dataset	Ours	Yeti
a	30.064	269.169
b	154.967	340.431
c	105.74	95.587
d	103.606	86.431
e	66.356	216.871
f	48.514	182.831
g	92.966	162.833
h	34.4	878.457
i	117.687	229.723

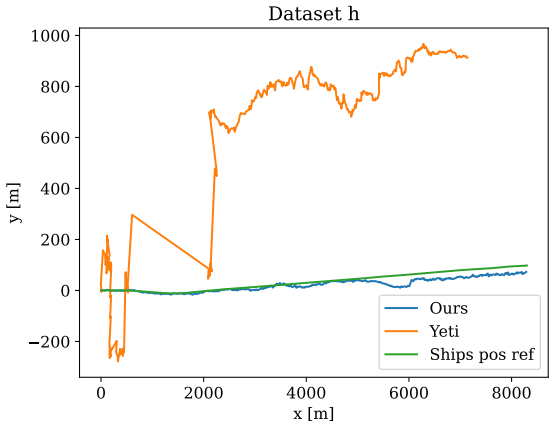

Fig. 3. The trajectory as seen from above generated from the dataset h. The trajectory is plotted in the frame of reference for the first pose in the sequence.

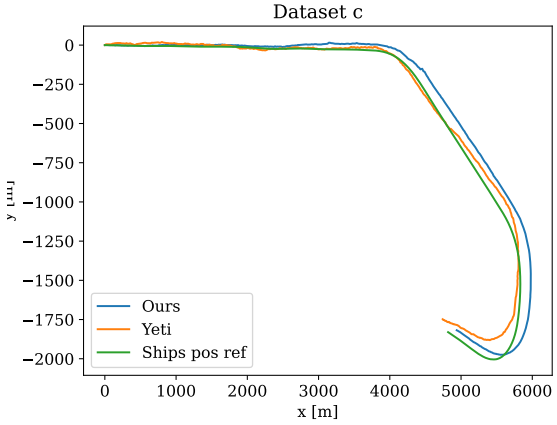

Fig. 4. The trajectory as seen from above generated from dataset c. The trajectory is plotted in the frame of reference for the first pose in the sequence.

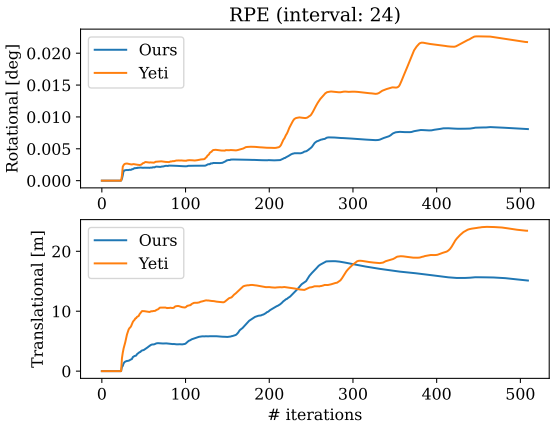

Fig. 5. The translational RPE during the dataset c.

dataset c, Fig. 5 shows that despite accumulating more drift, our method has a persistently lower short-term drift. Fig. 4 indicates that this might be due to Yeti drifting both back and forth around the true value while our method consistently drifts to the same side.

V. Discussion

The tests indicate that our method which is developed and tuned for maritime radar images outperforms yeti which is developed for automobile scenarios. Both the short-term drift is lower and we avoid discontinuities in the estimate in all the datasets. Neither of the methods explicitly prohibits these discontinuities by eg. between factors, but the KLT-tracking appears to be sufficiently stable to avoid it.

It is difficult to say why Yeti fail some sequences and survive others, but it could be one of several differences between our method and Yeti. E.g. we include a robust Huber norm in our cost function which makes it more robust to non-stationary objects. E.g. in dataset h, there are several small dots leading to weak non-stationary returns. The ORB-based feature tracker is also more susceptible to repetitive terrain. Our method does not compensate for the rotation of the radar in the same way as Yetis motion compensation, but this disadvantage seems to be counteracted by the others. This effect is particularly strong during sharp turns, which can be seen around iterations 150-250 in the translational part of Fig. 5. During the turns, the RPE of our method increases drastically while Yeti's RPE increases slowly(but from a higher level).

VI. Conclusion

We presented a method for maritime radar odometry which is inspired by the progress in the visual (inertial) odometry field. From the tests on radar data collected from a large ship, we have indications that our implementation is robust but susceptible to drift. This is despite that we do not explicitly compensate for the effects like the rotation rate of the radar, sea clutter, or shadows in the radar image. Still, we expect that compensating for these effects and other peculiarities of the radar sensor can improve accuracy and reduce drift and therefore consider this future work.

References

[1] Jungwook Han, Yonghoon Cho, and Jinwhan Kim. Coastal SLAM with Marine Radar for USV Operation in GPS-Restricted Situations. *IEEE Journal of Oceanic Engineering*, 44(2):300–309, April 2019. doi: 10.1109/JOE.2018.2883887. Publisher: Institute of Electrical and Electronics Engineers Inc.

[2] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J. Leonard. Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age. *IEEE Transactions on Robotics*, 32(6):1309–1332, December 2016. ISSN 1941-0468. Conference Name: IEEE Transactions on Robotics.

[3] César Debeunne and Damien Vivet. A Review of Visual-LiDAR Fusion based Simultaneous Localization and Mapping. *Sensors*, 20(7):2068, January 2020. ISSN 1424-8220. Number: 7 Publisher: Multidisciplinary Digital Publishing Institute.

[4] Jonas Callmer, David Törnqvist, Fredrik Gustafsson, Henrik Svensson, and Pelle Carlbom. Radar SLAM using visual features. *EURASIP Journal on Advances in Signal Processing*, 2011(1):1–11, September 2011. ISSN 1687-6180. doi: 10.1186/1687-6180-2011-71. Publisher: SpringerOpen.

[5] John Mullane, Samuel Keller, Akshay Rao, Martin Adams, Anthony Yeo, Franz S. Hover, and Nicholas M. Patrikalakis. X-band radar based SLAM in Singapore's off-shore environment. *11th International Conference on Control, Automation, Robotics and Vision, ICARCV 2010*, pages 398–403, 2010. doi: 10.1109/ICARCV.2010.5707835.

[6] Jungwook Han, Jeonghong Park, Jinwhan Kim, and Nam-sun Son. GPS-less Coastal Navigation using Marine Radar for USV Operation. *IFAC-PapersOnLine*, 49 (23):598–603, January 2016. ISSN 2405-8963. doi: 10.1016/j.ifacol.2016.10.500.

[7] Chao Wu, Qing Wu, Feng Ma, and Shuwu Wang. A novel positioning approach for an intelligent vessel based on an improved simultaneous localization and mapping algorithm and marine radar:. *https://doi.org/10.1177/1475090218784449*, 233(3):779–792, July 2018. Publisher: SAGE PublicationsSage UK: London, England.

[8] Carl H. Schiller, Bruno Arsenali, Deran Maas, and Stefano Maranó. Improving Marine Radar Odometry by Modeling Radar Resolution and Exploiting Additional Temporal Information. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8436–8441, October 2022. doi: 10.1109/IROS47612.2022.9981293. ISSN: 2153-0866.

[9] Vladyslav Usenko, Nikolaus Demmel, David Schubert, Jörg Stückler, and Daniel Cremers. Visual-Inertial Mapping With Non-Linear Factor Recovery. *IEEE Robotics and Automation Letters*, 5(2):422–429, April 2020. ISSN 2377-3766. doi: 10.1109/LRA.2019.2961227. Conference Name: IEEE Robotics and Automation Letters.

[10] Antoni Rosinol, Marcus Abate, Yun Chang, and Luca Carlone. Kimera: an Open-Source Library for Real-Time Metric-Semantic Localization and Mapping. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1689–1696, May 2020. ISSN: 2577-087X.

[11] Andrei Cramariuc, Lukas Bernreiter, Florian Tschopp, Marius Fehr, Victor Reijgwart, Juan Nieto, Roland Siegwart, and Cesar Cadena. maplab 2.0 – A Modular and Multi-Modal Mapping Framework. *IEEE Robotics and Automation Letters*, 8(2):520–527, February 2023. ISSN 2377-3766, 2377-3774. arXiv:2212.00654 [cs].

[12] Tong Qin, Peiliang Li, and Shaojie Shen. VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, August 2018. ISSN 1941-0468. doi: 10.1109/TRO.2018.2853729. Conference Name: IEEE Transactions on Robotics.

[13] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John Leonard, and Frank Dellaert. iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering. In *2011 IEEE International Conference on Robotics and Automation*, pages 3281–3288, May 2011. ISSN: 1050-4729.

[14] Tixiao Shan, Brendan Englot, Drew Meyers, Wei Wang, Carlo Ratti, and Daniela Rus. LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5135–5142, October 2020. ISSN: 2153-0866.

[15] Jianbo Shi and Tomasi. Good features to track. In *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, June 1994. ISSN: 1063-6919.

[16] Simon Baker and Iain Matthews. Lucas-Kanade 20 Years On: A Unifying Framework. *International Journal of Computer Vision*, 56(3):221–255, February 2004. ISSN 1573-1405.

[17] Frank Dellaert and GTSAM Contributors. borglab/gtsam, May 2022. URL https://github.com/borglab/gtsam).

[18] Jérémie Deray and Joan Solà. Manif: A micro Lie theory library for state estimation in robotics applications. *Journal of Open Source Software*, 5(46):1371, February 2020. ISSN 2475-9066.

[19] Keenan Burnett, Angela P. Schoellig, and Timothy D. Barfoot. Do We Need to Compensate for Motion Distortion and Doppler Effects in Spinning Radar Navigation? *IEEE Robotics and Automation Letters*, 6(2):771–778, April 2021. ISSN 2377-3766. doi: 10.1109/LRA.2021.3052439. Conference Name: IEEE Robotics and Automation Letters.

[20] Keenan Burnett, David J. Yoon, Angela P. Schoellig, and Tim Barfoot. Radar Odometry Combining Probabilistic Estimation and Unsupervised Feature Learning. volume 17, July 2021. ISBN 978-0-9923747-7-8.

[21] Sarah H. Cen and Paul Newman. Precise Ego-Motion Estimation with Millimeter-Wave Radar under Diverse and Challenging Conditions. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 6045–6052, September 2018. doi: 10.1109/ICRA.2018.8460687. Publisher: Institute of Electrical and Electronics Engineers Inc.

[22] Jrgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580, Vilamoura-Algarve, Portugal, October 2012. IEEE. ISBN 978-1-4673-1736-8 978-1-4673-1737-5 978-1-4673-1735-1.