15

# Feature extraction: techniques for landmark based navigation system

Molaletsa Namoshe[1,2], Oduetse Matsebe[1,2] and Nkgatho Tlale[1]
[1]*Department of Mechatronics and Micro Manufacturing, Centre for Scientific and Industrial Research,*
[2]*Department of Mechanical Engineering, Tshwane University of Technology, Pretoria, South Africa*

## 1. Introduction

A robot is said to be fully autonomous if it is able to build a navigation map. The map is a representation of a robot surroundings modelled as 2D geometric features extracted from a proximity sensor like laser. It provides succinct space description that is convenient for environment mapping via data association. In most cases these environments are not known prior, hence maps needs to be generated automatically. This makes feature based SLAM algorithms attractive and a non trivial problems. These maps play a pivotal role in robotics since they support various tasks such as mission planning and localization. For decades, the latter has received intense scrutiny from the robotic community. The emergence of stochastic map proposed by seminal papers of (Smith et al., 1986; Moutarlier et al., 1989a; Moutarlier et al., 1989b & Smith et al., 1985), however, saw the birth of joint posterior estimation. This is a complex problem of jointly estimating the robot's pose and the map of the environment consistently (Williams S.B et al., 2000) and efficiently. The emergence of new sensors systems which can provide information at high rates such as wheel encoders, laser scanners and sometimes cameras made this possible. The problem has been research under the name Simultaneous Localization and Mapping (SLAM) (Durrant-Whyte, H et al. 2006 Part I and II) from its inception. That is, to localize a mobile robot, geometric features/ landmarks (2D) are generated from a laser scanner by measuring the depth to these obstacles. In office like set up, point (from table legs), line (walls) and corner (corner forming walls) features makes up a repeated recognisable pattern formed by a the laser data. These landmarks or features can be extracted and used for navigation purposes. A robot's perception of its position relative to these landmarks increases, improving its ability to accomplish a task. In SLAM, feature locations, robot pose estimates as well feature to robot pose correlations statistics are stochastically maintained inside an Extended Kalman filter increasing the complexity of the process (Thorpe & Durrant-Whyte, 2001). It is also important to note that, though a SLAM problem has the same attributes as estimation and tracking problems, it is not fully observable but detectable. This has a huge implication in the solution of SLAM problem. Therefore, it is important to develop robust extraction algorithms of geometric features from sensor data to aid a robot navigation system.

Accurate and reliable maps generated autonomously guarantees improved localization especially in GPS denied surroundings like indoor (Hough, P.V.C, 1959). The use of odometry is not sufficient for position estimation due unbounded position errors. Therefore, since office like environments consists of planar surfaces, a 2D space model is adequate to describe the robot surroundings because objects are predominantly straight line segments and right angle corners. Coincidentally, line segments and corner representation are the two most popular methods for indoor modelling from a laser rangefinder. The focus in this paper however is corner extraction methods. A number of line and corner extraction techniques first transform scan data into Cartesian space then a linear regression method or corner extraction algorithm is applied. Some algorithms employ Hugh transform (Hough, P.V.C, 1959). & (Duda, R. O, 1972) a popular tool for line detection from scan data due to its robustness to noise and missing data. It works in sensor measurement space. However, the computational cost associated to its voting mechanism renders real-time implementation impossible. On the other hand, an early work by (Crowley, J, 1989) paved the way to subsequent line extraction methods from a range sensor. In their work, a process for extracting line segments from adjacent co-linear range measurements was presented. The Kalman filter update equations were developed to permit the correspondence of a line segment to the model to be applied as a correction to estimated position. The approach was recently extended by (Pfister, S.T et al. 2003), first providing an accurate means to fit a line segment to a set of uncertain points via maximum likelihood formalism. Then weights were derived from sensor noise models such that each point's influence on the fit is according to its uncertainty. Another interesting work is one by (Roumeliotis & Bekey, 2000), where two Extended Kalman filters are used to extract lines from the scan data. In the algorithm, one Kalman filter is used to track the line segments while the other estimates line parameters. The combination of the two filters makes it possible to detect edges and straight line segments within the sensor field of view. There are many features types one can extract from a laser sensor, and are dependent on the obstacles found in the room. If the room has chair and table, one would be tempted to extract point features from their legs. Size, shape and texture of objects contribute to the type of feature to extract from the sensor. The use of generalised algorithms is not uncommon, i.e. algorithms which extract lines from wall, point features from table legs and arcs to categorise circular objects (Mendes, & Nunes, 2004). The parameters that distinguish each extracted feature makes up the map or state estimate.

The key to a successful robot pose estimation lies in its ability to effectively extract useful information about its location from observations (Li & Jilkov, 2003). Therefore we proposed an improved corner detection method to reduce computational cost and improved robustness.

The paper is structured as follows; section 2 deals with feature extraction, section 3 discuss the EKF-SLAM process. Section 4 is result and analysis, while section 5 covers conclusion and future work.

## 2. Feature Extraction

Feature extraction forms the lower part of the two layered procedure of feature detection. The top tier is the data segmentation process, which creates clusters of points deemed to originate from the same obstacle. It groups measurements of a scan into several clusters according to the distances between consecutive scans. These segments sectors then are fed to

the feature extraction algorithms, where features like corners or lines are considered. These features are well defined entities which are recognisable and can be repeatedly detected.
In this paper, real laser data from the sensor onboard a robot is processed to extract corner like features, common in most indoor environments. A robot used for this experiment is called Meer-Cat and was developed in house, depicted by Figure 1 below.

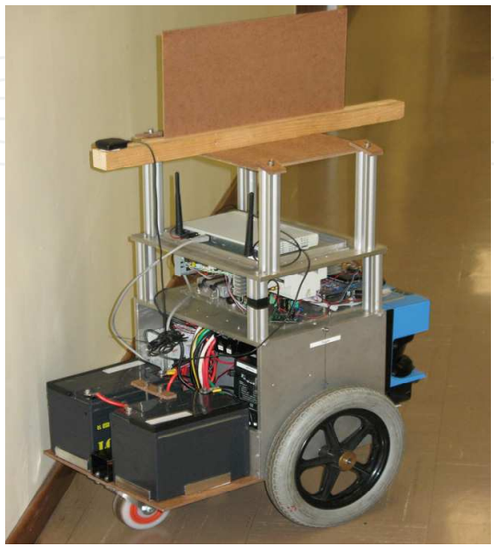


Fig. 1. Meer-Cat mobile platform equipped with Sick laser scanner. The robot has an upright board at the top used for tracking purposes via another laser sensor.

### 2.1 Corner Extraction

Most corner detection algorithms utilises a sliding window technique (Spinello, L, 2007) or picking out the ends points of a line segment as a corners, e.g. slight-and- Merge (Pfister, S.T et al. 2003). This is normally where two line segments meet. Although, an algorithm by (Einsele, T, 2001) is a Split and Merge procedure and it determine corners likewise, it has a slight variation in data processing. The following subsections discuses methods of corner extraction, to be used by an indoor navigation system.

#### 2.1.1 Sliding window corner detector

The sliding window technique has three main parts; vectors determination from three points (Cartesian points), Angle check between the vectors, and the backward check when a corners angle is satisfied. Firstly the size of a window is determined by pre-setting a midpoint position. That is, a window sector size of 11 sample scans has midpoint at $6^{th}$ sample data, 13 at $7^{th}$, and 15 at $8^{th}$ and so on. The window is broken into two vectors ($vi$ and $vj$), such that for an 11 sample size window, the first and the eleventh samples are terminal points of these vectors. Therefore, the algorithm assumes a corner if the vectors forms a triangular shape with the midpoint sample being one of its vertexes. An iterative search for a corner angle is carried out by sliding the window step by step over the entire scan. If conditions are met a corner is noted at midpoint. That is, an up bound for the angle

between the vectors as well as the minimum allowable opposite distance c as shown in figure 2b below are set prior. A corner is normally described by angles less than 120 degrees, while the separation distance is tightly related to the angular resolution of the laser rangefinder. The distance c is set to very small values; computations greater than this value are passed as corners. If a corner is detected, an 'inward' search is conducted. This is done by checking for a corner angle violation/ existence between the 2nd and 10th, 3rd and 9th, and so on, for sample sector of 11 data points. This is from the assumption that a linear fit can be performed on the vectors. The searching routine of this method already demand high computation speed, therefore inward search will undoubtedly increase the complexity.

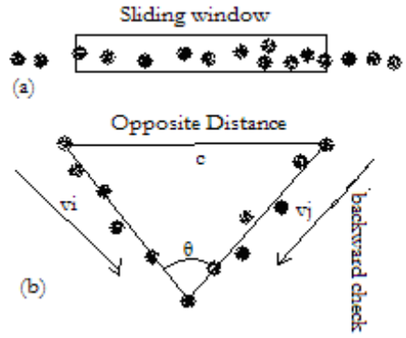



Fig. 2. (a), Sliding window technique. (b) Shows how two vectors centred at the midpoint are derived if a corner if found. The terminal points are at the first and the eleventh point given that the midpoint of the sector is 6.

The angle is calculated using cosine rule, that is,

$$\theta = \cos^{-1}(vi.vj / (\|vi\|\|vj\|)) . \tag{1}$$

Using the above methods one runs into the problem of mapping outliers as corners. This has huge implication in real time implementation because computation complexity of the SLAM process is quadratic the number of landmarks mapped. The outliers or 'ghost' landmarks corrupt the EKF SLAM process.

### 2.1.2 Split and Merge

Laser sensor produces range scans which describes a 2D slice of the environment. Each range point is specified in polar coordinates system whose origin is the location of the sensor on board the robot. Scan data from a laser range finder has almost negligible angular uncertainty, and the noise on range measurement is assumed to follow Gaussians distribution properties. Data segments originating from the same object can be represented by a line. And traditionally, straight lines are represented by the following parameters

$$y = mx + c \tag{2}$$

where $c$ and $m$ is the $y$ -intercept and slope of a line respectively. The shortcoming with this representation is that vertical lines require infinite $m$ (gradient).

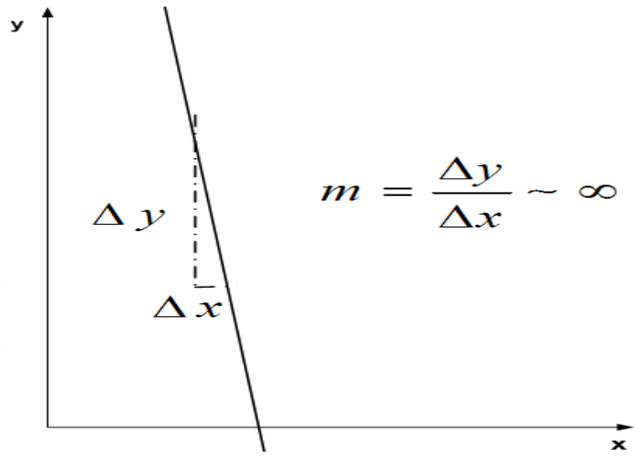



Fig. 3. As the line become vertical, the slope approaches infinity.

If objects in an environment can be represented by polygonal shapes, then line fitting is a suitable choice to approximate objects shapes. During data segmentation, clusters are formed, and a cluster can be represented by a set of lines, defined as follows:

$$C = \{l_i = [P_i, P_f, m, b]^T : 0 \le i < n\} \tag{3}$$

where $P_i$ and $P_f$ are respectively the Cartesian coordinates of the initial and the end of a line. While $m$ and $b$ are the parameters of an i$^{th}$ line. A method proposed by [14] is used to search for a breaking point of a cluster, which occurs at the maximum perpendicular distance to a line. The process starts by connecting the first and last data points of a cluster by a straight line ( $Ax + By + C = 0$ ), where

$A = y_f - y_i ; B = x_f - x_i ; C = -(By_f - Ax_f)$ . Then for all data points between the extreme points, a perpendicular distance $d_\perp$ to the line is calculated. Such that

$$d_{\perp,k} = \frac{Ax_k + By_k + C}{\sqrt{A^2 + B^2}}. \tag{4}$$

If a tolerance value is violated by the $d_\perp$ then a break point is determined, this is done recursively until the point before last. The final step is to determine straight line parameters, i.e. an orthogonal regression method (Mathpages 2010-04-23) is applied to determine linear fit that minimizes quadratic error. The process is graphically represented by the figure below

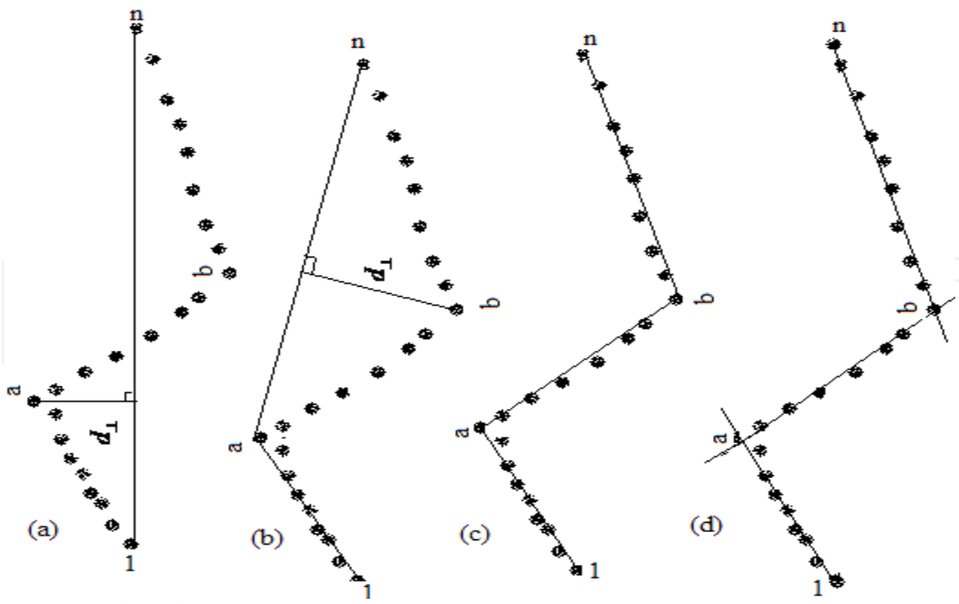



Fig. 4. Recursive line fitting

To mitigate the infinite slope problem, a polar representation or Hessen form is used. In the method, each point in the Cartesian coordinate space adds a sinusoid in the ( $\rho$ , $\theta$ ) space. This is shown the figure 5 below.

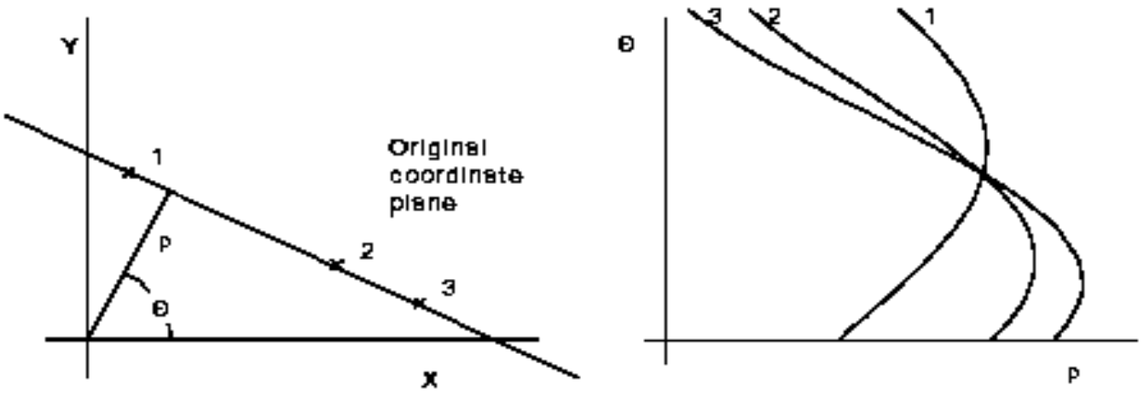



Fig. 5. Mapping between the Cartesian space and the polar Space.

The polar form used to represent lines is given as follows

$$\rho = x\cos(\theta) + y\sin(\theta) \tag{5}$$

where $\rho \geq 0$ is the perpendicular distance of the line to the origin. The angle $\theta$ is bounded by $-\pi < \theta \geq \pi$ and is the angle between the x axis and the normal of the line as shown in the figure 6 below.

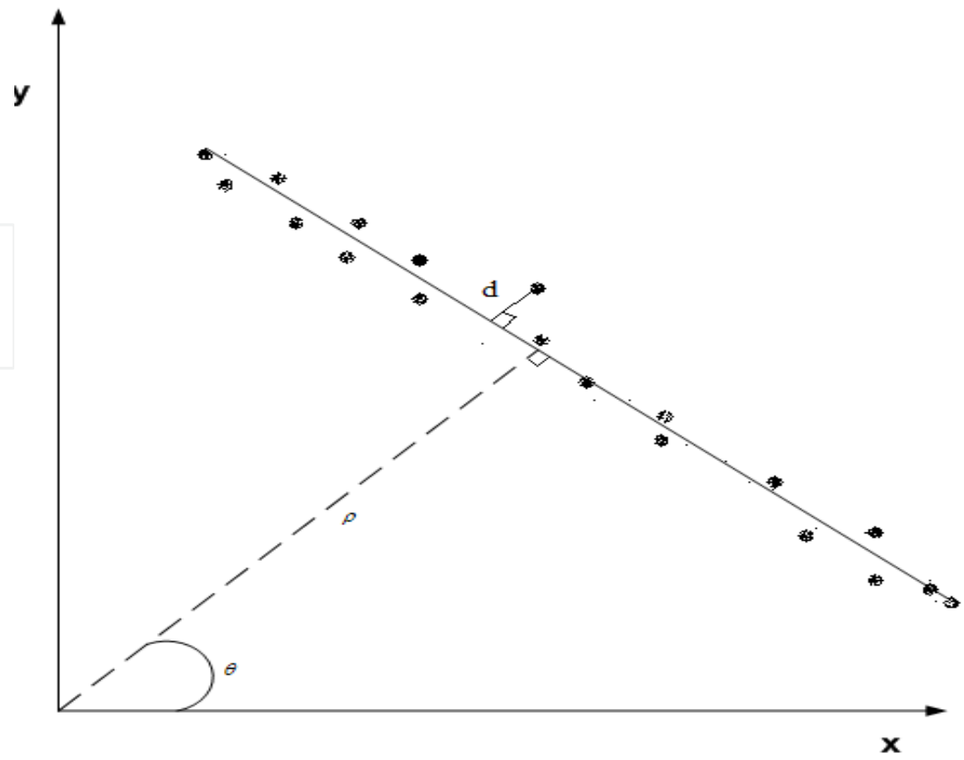



Fig. 6. Fitting line parameters. $d$ is the fitting error we wish to minimize. A line is expressed in polar coordinates ( $\rho$ and $\theta$ ). ( $x$ , $y$ ) is the Cartesian coordinates of a point on the line.

Using the above representation, the split-and-merge algorithm recursively subdivides scan data into sets of collinear points, approximated as lines in total least square sense. The algorithm determines corners by two main computations, the line extraction and collection of endpoints as corners. Initially, scanned data is clustered into sectors assumed to come from the same objects. The number of data points within a certain cluster as well as an identification of that cluster is stored. Clusters are then passed to a line fitting algorithm (Lu & Milios, 1994). When we perform a regression fit of a straight line to a set of ( $x$ , $y$ ) data points we typically minimize the sum of squares of the "vertical" distance between the data points and the line (Mathpages 2010-04-23).Therefore, the aim of the linear regression method is to minimize the mean squared error of

$$d^2 = \sum_i \left( \rho - \{ x_i \cos(\theta) + y_i \sin(\theta) \} \right)^2 \tag{6}$$

such that $(x_i, y_i)$ are the inputs points in Cartesian coordinates. The solution to the line parameters can be found by taking the first derivative of the equation 6 above with respect to $\rho$ and $\theta$ respectively. We assume that

$$\frac{\partial d^2}{\partial \rho} = 0 \qquad and \qquad \frac{\partial d^2}{\partial \theta} = 0 \tag{7}$$

Line parameters can be determined by the following

$$\tan(2\theta) = \frac{-2\sum (y_m - y_i)(x_m - x_i)}{\sum [(y_m - y_i)^2 - (x_m - x_i)^2]}$$
$$\theta = 0.5a \tan 2\left( \frac{-2\sum (y_m - y_i)(x_m - x_i)}{\sum [(y_m - y_i)^2 - (x_m - x_i)^2]} \right) \tag{8}$$

if we assume that the Centroid is on the line then $\rho$ can be computed using equation 4 as:

$$\rho = x_m \cos(\theta) + y_m \sin(\theta) \tag{9}$$

where

$$x_m = {}^{1}\!/_{N} \sum x_i$$
$$and \tag{10}$$
$$y_m = {}^{1}\!/_{N} \sum y_i$$

are ( $x_m$ , $y_m$ ) are Cartesian coordinates of the Centroid, and N is the number of points in the sector scan we wish to fit line parameter to.

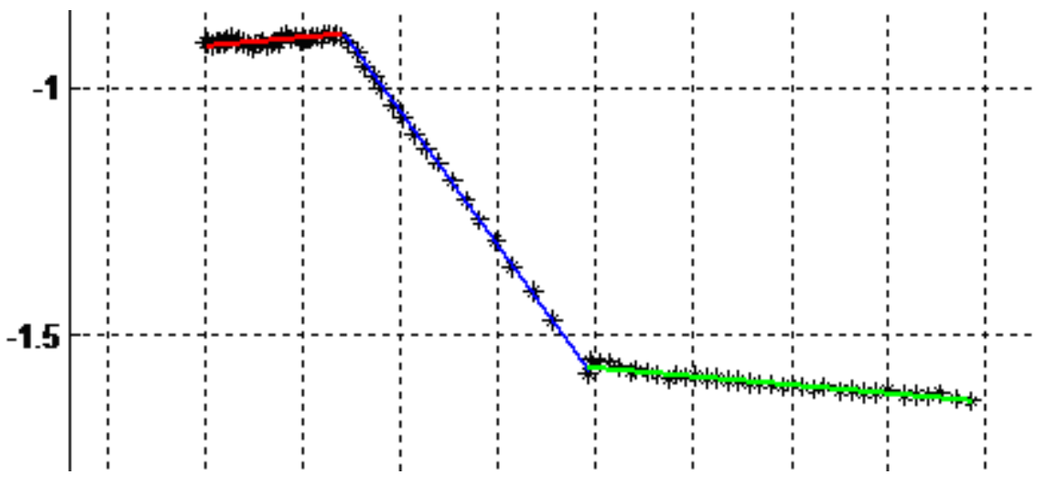



Fig. 7. Fitting lines to a laser scan. A line has more than four sample points.

During the line fitting process, further splitting positions within a cluster are determined by computing perpendicular distance of each point to the fitted line. As shown by figure 6. A point where the perpendicular distance is greater than the tolerance value is marked as a candidate splitting position. The process is iteratively done until the whole cluster scan is made up of linear sections as depicted by figure 7 above. The next procedure is collection of endpoints, which is joining points of lines closest to each other. This is how corner positions are determined from split and merge algorithm. The figure below shows extracted corners defined at positions where two line meet. These positions (corners) are marked in pink.

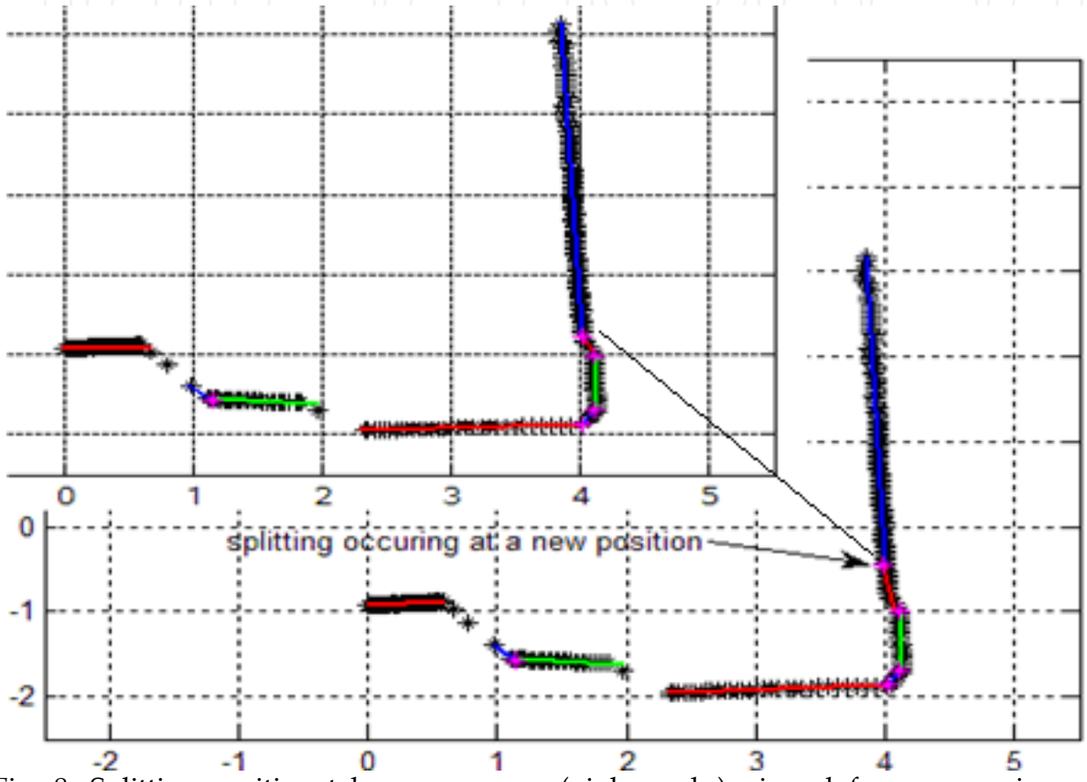



Fig. 8. Splitting position taken as corners (pink marks) viewed from successive robot positions. The first and second extraction shows 5 corners. Interestingly, in the second extraction a corner is noted at a new position, In SLAM, the map has total of 6 landmarks in the state vector instead of 5. The association algorithm will not associate the corners; hence a new feature is mapped corrupting the map.

The split and merge corner detector brings up many possible corners locations. This has a high probability of corrupting the map because some corners are 'ghosts'. There is also the issue of computation burden brought about by the number of landmarks in the map. The standard EKF-SLAM requires time quadratic in the number of features in the map (Thrun, S et al. 2002).This computational burden restricts EKF-SLAM to medium sized environments with no more than a few hundred features.

### 2.1.3 Proposed Method

We propose an extension to the sliding window technique, to solve the computational cost problem and improve the robustness of the algorithm. We start by defining the limiting bounds for both angle $\theta$ and the opposite distance c. The first assumption we make is that a corner is determined by angles between 70° to 110°. To determine the corresponding lower and upper bound of the opposite distance c we use the minus cosine rule. Following an explanation in section 2.1.1, lengths vectors of are determined by taking the modulus of $vi$ and $vj$ such that $a = \|v_i\|$ and $b = \|v_j\|$. Using the cosine rule, which is basically an extension of the Pythagoras rule as the angle increases/ decreases from the critical angle (90˚), the minus cosine function is derived as:

$$c^2 = a^2 + b^2 + 2abf(\theta)$$
$$where \tag{11}$$
$$f(\theta) = \frac{c^2 - (a^2 + b^2)}{2ab}$$

where $f(\theta)$ is minus cosine $\theta$. The limits of operating bounds for c can be inferred from the output of $f(\theta)$ at corresponding bound angles. That is, $\theta$ is directly proportion to distance c. Acute angles give negative results because the square of $c$ is less than the sum of squares of $a$ and $b$. The figure 9 below shows the angle-to-sides association as well as the corresponding $f(\theta)$ results as the angle grows from acuteness to obtuseness.

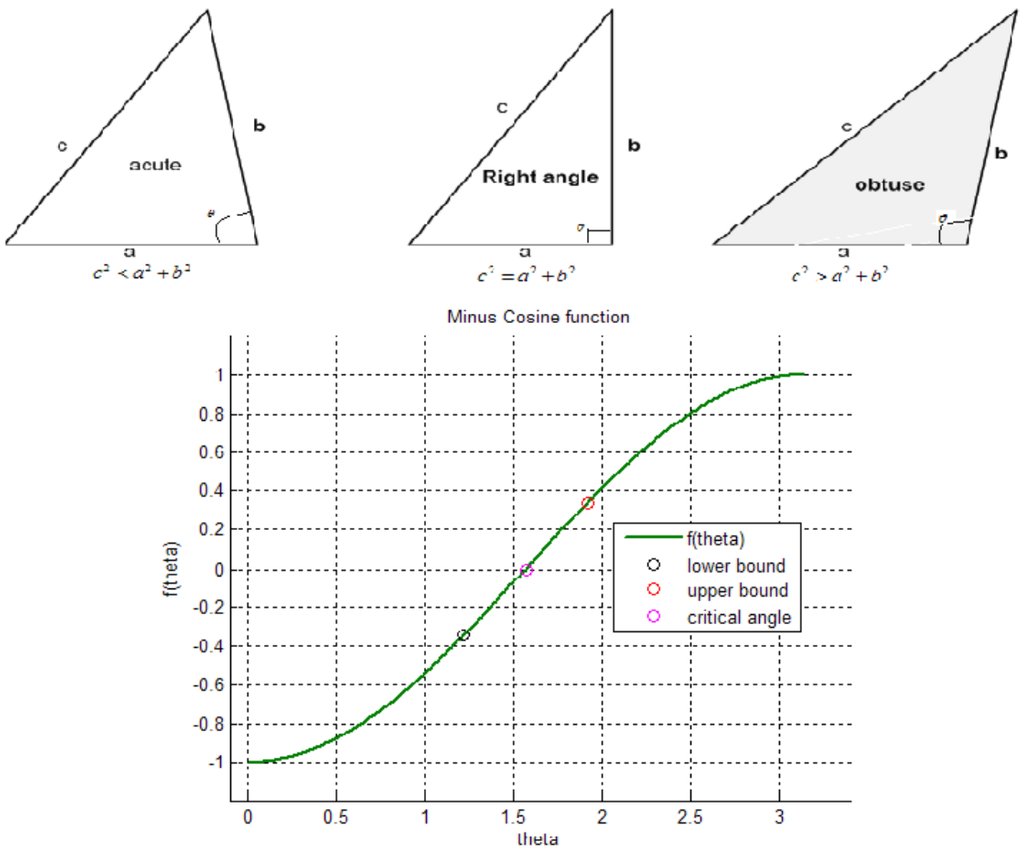



Fig. 9. The relation of the side lengths of a triangle as the angle increases. Using minus cosine function, an indirect relationship is deduced as the angle is increased from acute to obtuse.

The $f(\theta)$ function indirectly has information about the minimum and maximum allowable opposite distance. From experiment this was found to be within [-0.3436 0.3515]. That is, any output within this region was considered a corner. For example, at 90˚ angle $c^2 = a^2 + b^2$, outputting zero for $f(\theta)$ function. As the angle $\theta$ increases, acuteness ends and obtuseness starts, the relation between $c^2$ and $a^2 + b^2$ is reversed.

The main aim of this algorithm is to distinguish between legitimate corners and those that are not (outliers). Corner algorithms using sliding window technique are susceptible to mapping outlier as corners. This can be shown pictorial by the figure below

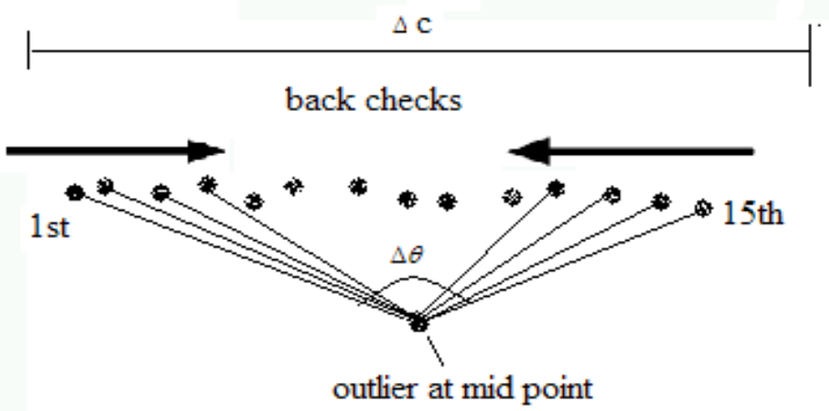



Fig. 10. Outlier corner mapping

where $\Delta\theta$ is the change in angle as the algorithm checks consecutively for a corner angle between points. That is, if there are 15 points in the window and corner conditions are met, corner check process will be done. The procedure checks for corner condition violation/ acceptance between the 2nd & 14th, 3rd & 13th, and lastly between the 4th & 12th data points as portrayed in figure 10 above. If $\Delta\theta$ does not violate the pre-set condition, i.e. (corner angles $\leq$ 120˚) then a corner is noted. $\Delta c$ is the opposite distance between checking points. Because this parameter is set to very small values, almost all outlier corner angle checks will pass the condition. This is because the distances are normally larger than the set tolerance, hence meeting the condition.

The algorithm we propose uses a simple and effect check, it shifts the midpoint and checks for the preset conditions. Figure 11 below shows how this is implemented

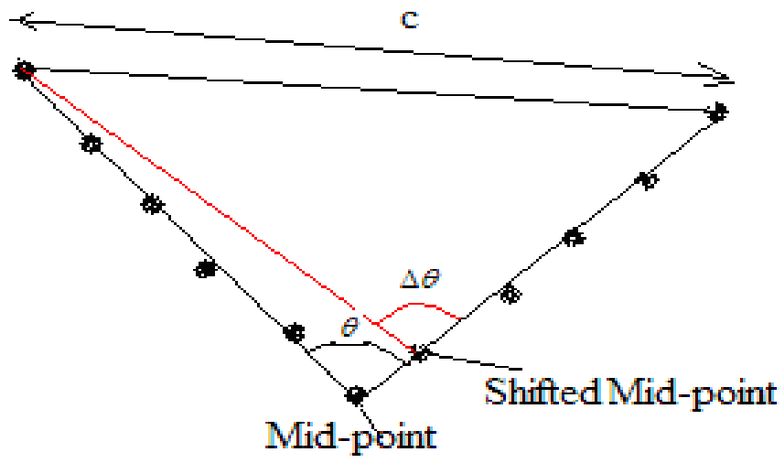



Fig. 11. Shifting the mid-point to a next sample point (e.g. the 7th position for a 11 sample size window) within the window

As depicted by figure 11 above, $\theta$ and $\Delta\theta$ angles are almost equal, because the angular resolution of the laser sensor is almost negligible. Hence, shifting the Mid-point will almost give the same corner angles, i.e. $\Delta\theta$ will fall with the $f(\theta)$ bounds. Likewise, if a Mid-

point coincides with the outlier position, and corner conditions are met, i.e. $\theta$ and $c$ (or $f(\theta)$ conditions) are satisfies evoking the check procedure. Shifting a midpoint gives a results depicted by figure 12 below.

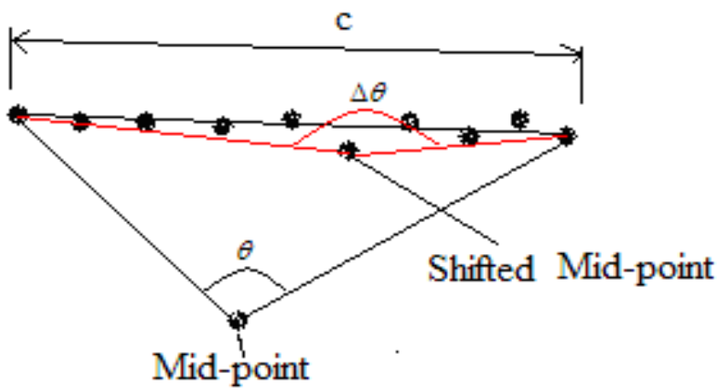



Fig. 12. If a Mid-point is shifted to the next consecutive position, the point will almost certainly be in-line with other point forming an obtuse triangle.

Evidently, the corner check procedure depicted above will violate the corner conditions. We expect $\Delta\theta$ angle to be close to 180˚ and the output of $f(\theta)$ function to be almost 1, which is outside the bounds set. Hence we disregard the corner findings at the Mid-point as ghost, i.e. the Mid-point coincide with an outlier point. The figure below shows an EKF SLAM process which uses the standard corner method, and mapping an outlier as corner.

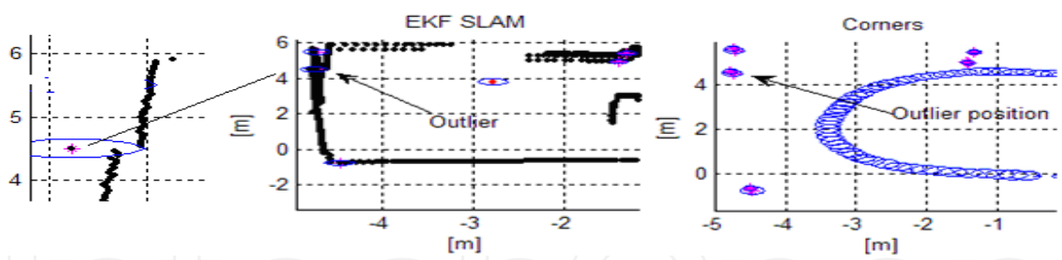



Fig. 13. Mapping outliers as corners largely due to the limiting bounds set. Most angle and opposite distances pass the corner test bounds.

```
[Rho, theta]= GetLaserMeasurements

lenMeasurements = length (Rho,2)  %Number of scan points

[x,y] = pol2cart (theta, Rho); % change to Cartesian coordinates

z = [ ]; % corner position in polar form

for i=1: lenMeasurements,
    if(isnan(Rho (i)) == 0)
      % check points
      corner = Corner_check ([x;y], i);

      if(corner)
         z = [z [Rho(i), theta(i)]]; %saving as z
      end
   end
end

function corner = Corner_check (XY, i)

FisrtTerminalPoint = i;
MidPoint = i +5;   % mid point position
SecondTerminalPoint = MidPoint +5;

[vi, vj] = calculate the two vectors

f(θ) = calculate the minus sign output from lengths of vectors

If ( f(θ) satisfies conditions)

        If ( Δθ is acceptable)

        Corner = 1;
        else
          corner = 0;
      end

else
corner = 0;

end
```

Fig. 14. A pseudo code for the proposed corner extractor.

A pseudo code in the figure is able to distinguish outlier from legitimate corner positions. This is has a significant implication in real time implementation especially when one maps large environments. EKF-SLAM's complexity is quadratic the number of landmarks in the map. If there are outliers mapped, not only will they distort the map but increase the computational complexity. Using the proposed algorithm, outliers are identified and discarded as ghost corners. The figure below shows a mapping result when the two algorithms are used to map the same area

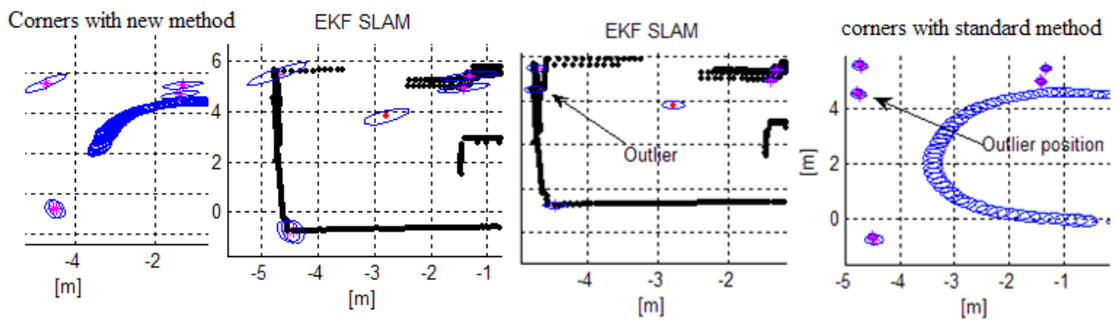



Fig. 15. Comparison between the two algorithms (mapping the same area)

## 3. EKF-SLAM

The algorithm developed in the previous chapter form part of the EKF-SLAM algorithms. In this section we discuss the main parts of this process. The EKF-SLAM process consists of a recursive, three-stage procedure comprising prediction, observation and update steps. The EKF estimates the pose of the robot made up of the position $(x_r, y_r)$ and orientation $\psi_r$, together with the estimates of the positions of the $N$ environmental features $\boldsymbol{x}_{f,i}$ where $i = 1 \cdots N$, using observations from a sensor onboard the robot (Williams, S.B et al. 2001).

SLAM considers that all landmarks are stationary; hence the state transition model for the $i^{th}$ feature is given by:

$$\boldsymbol{x}_{f,i}(k) = \boldsymbol{x}_{f,i}(k-1) = \boldsymbol{x}_{f,i} \tag{12}$$

It is important to note that the evolution model for features does have any uncertainty since the features are considered static.

### 3.1 Process Model

Implementation of EKF-SLAM requires that the underlying state and measurement models to be developed. This section describes the process models necessary for this purpose.

#### 3.1.1 Dead-Reckoned Odometry Measurements

Sometimes a navigation system will be given a dead reckoned odometry position as input without recourse to the control signals that were involved. The dead reckoned positions can

be converted into a control input for use in the core navigation system. It would be a bad idea to simply use a dead-reckoned odometry estimate as a direct measurement of state in a Kalman Filter (Newman, P, 2006).

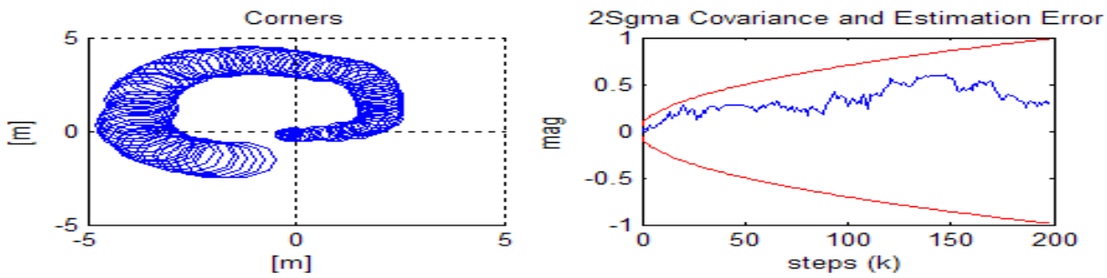



Fig. 16. Odometry alone is not ideal for position estimation because of accumulation of errors. The top left figure shows an ever increasing $2\sigma$ bound around the robot's position.

Given a sequence $\boldsymbol{x}_0(1), \boldsymbol{x}_0(2), \boldsymbol{x}_0(3), \cdots \boldsymbol{x}_0(k)$ of dead reckoned positions, we need to figure out a way in which these positions could be used to form a control input into a navigation system. This is given by:

$$\boldsymbol{u}_o(k) = \Theta \boldsymbol{x}_o(k-1) \oplus \boldsymbol{x}_o(k) \tag{13}$$

This is equivalent to going back along $\boldsymbol{x}_0(k-1)$ and forward along $\boldsymbol{x}_0(k)$. This gives a small control vector $\boldsymbol{u}_0(k)$ derived from two successive dead reckoned poses. Equation 13 subtracts out the common dead-reckoned gross error (Newman, P, 2006). The plant model for a robot using a dead reckoned position as a control input is thus given by:

$$\boldsymbol{X}_r(k) = \boldsymbol{f}(\boldsymbol{X}_r(k-1), \boldsymbol{u}(k)) \tag{14}$$

$$\boldsymbol{X}_r(k) = \boldsymbol{X}_r(k-1) \oplus \boldsymbol{u}_o(k) \tag{15}$$

$\Theta$ and $\oplus$ are composition transformations which allows us to express robot pose described in one coordinate frame, in another alternative coordinate frame. These composition transformations are given below:

$$\boldsymbol{x}_1 \oplus \boldsymbol{x}_2 = \begin{bmatrix} x_1 + x_2 \cos\theta_1 - y_2 \sin\theta_1 \\ y_1 + x_2 \sin\theta_1 + y_2 \cos\theta_1 \\ \theta_1 + \theta_2 \end{bmatrix} \tag{16}$$

$$\Theta \boldsymbol{x}_1 = \begin{bmatrix} -x_1 \cos\theta_1 - y_1 \sin\theta_1 \\ x_1 \sin\theta_1 - y_1 \cos\theta_1 \\ -\theta_1 \end{bmatrix} \tag{17}$$

### 3.2 Measurement Model

This section describes a sensor model used together with the above process models for the implementation of EKF-SLAM. Assume that the robot is equipped with an external sensor capable of measuring the range and bearing to static features in the environment. The measurement model is thus given by:

$$\boldsymbol{z}(k) = \boldsymbol{h}(\boldsymbol{X}_r(k), x_i, y_i) + \boldsymbol{\gamma}_h(k) = \begin{bmatrix} r_i(k) \\ \theta_i(k) \end{bmatrix} \tag{18}$$

$$r_i = \sqrt{\left(x_i - x_r\right)^2 + \left(y_i - y_r\right)^2} \tag{19}$$

$$\theta_i = \tan^{-1}\left[\frac{y_i - y_r}{x_i - x_r}\right] - \psi_r \tag{20}$$

$(x_i, y_i)$ are the coordinates of the $i^{th}$ feature in the environment. $\boldsymbol{X}_r(k)$ is the $(x, y)$ position of the robot at time $k$. $\gamma_h(k)$ is the sensor noise assumed to be temporally uncorrelated, zero mean and Gaussian with standard deviation $\sigma$. $r_i(k)$ and $\theta_i(k)$ are the range and bearing respectively to the $i^{th}$ feature in the environment relative to the vehicle pose.

$$\boldsymbol{\gamma}_h(k) = \begin{bmatrix} \sigma_r \\ \sigma_\theta \end{bmatrix} \tag{21}$$

The strength (covariance) of the observation noise is denoted $\boldsymbol{R}$.

$$\boldsymbol{R} = diag\begin{pmatrix} \sigma_r^2 & \sigma_\theta^2 \end{pmatrix} \tag{22}$$

### 3.3 EKF-SLAM Steps

This section presents the three-stage recursive EKF-SLAM process comprising prediction, observation and update steps. Figure 17 below summarises the EKF - SLAM process described here.

$\boldsymbol{x}_{0|0} = 0; \boldsymbol{P}_{0|0} = 0$ Map initialization

$[z_0, R_0] = GetLaserSensorMeasuremet$

If ( $z_0$ ! =0)

$$\left[\boldsymbol{x}_{0|0}, \boldsymbol{P}_{0|0}\right] = AugmentMap(\boldsymbol{x}_{0|0}; \boldsymbol{P}_{0|0}, \boldsymbol{z}_0, \boldsymbol{R}_0)$$

End

For k = 1: NumberSteps (=N)

$$\left[\boldsymbol{x}_{R,kk-1}, \boldsymbol{Q}_k\right] = GetOdometryMeasurement$$

$$\left[\boldsymbol{x}_{k|k-1}, \boldsymbol{P}_{k|k-1}\right] = EKF\_\mathrm{Pr}edict(\boldsymbol{x}_{k-1|k-1}; \boldsymbol{P}_{k-1|k-1}, \boldsymbol{x}_{Rk|k-1})$$

$[z_k, R_k] = GetLaserSensorMeasuremet$

$H_k = DoDataAssociation(\boldsymbol{x}_{k|k-1}, \boldsymbol{P}_{k|k-1}, \boldsymbol{z}_k, R_k)$

$\left[\boldsymbol{x}_{k|k}, \boldsymbol{P}_{k|k}\right] = EKF\_Update(\boldsymbol{x}_{k|k-1}; \boldsymbol{P}_{k|k-1}, \boldsymbol{z}_k, R_k, H_k)$ {If a feature exists in the map}

$\left[\boldsymbol{x}_{k|k}, \boldsymbol{P}_{k|k}\right] = AugmentMap(\boldsymbol{x}_{k|k-1}; \boldsymbol{P}_{k|k-1}, \boldsymbol{z}_k, R_k, H_k)$ {If it's a new feature}

**If** ( $z_k$ = =0)

$$\left[\boldsymbol{x}_{k|k}, \boldsymbol{P}_{k|k}\right] = \left[\boldsymbol{x}_{k|k-1}, \boldsymbol{P}_{k|k-1}\right]$$

**end**

end

Fig. 17. EKF- SLAM pseudo code

### 3.3.1 Map Initialization

The selection of a base reference $^{B}$ to initialise the stochastic map at time step 0 is important. One way is to select as base reference the robot's position at step 0. The advantage in choosing this base reference is that it permits initialising the map with perfect knowledge of the base location (Castellanos, J.A et al. 2006).

$$\boldsymbol{X}_0^B = \boldsymbol{X}_r^B = 0 \tag{23}$$

$$\boldsymbol{P}_0^B = \boldsymbol{P}_r^B = 0 \tag{24}$$

This avoids future states of the vehicle's uncertainty reaching values below its initial settings, since negative values make no sense. If at any time there is a need to compute the vehicle location or the map feature with respect to any other reference, the appropriate transformations can be applied. At any time, the map can also be transformed to use a

feature as base reference, again using the appropriate transformations (Castellanos, J.A et al. 2006).

### 3.3.2 Prediction using Dead-Reckoned Odometry Measurement as inputs

The prediction stage is achieved by a composition transformation of the last estimate with a small control vector calculated from two successive dead reckoned poses.

$$\boldsymbol{X}_r(k \mid k-1) = \boldsymbol{X}_r(k-1 \mid k-1) \oplus \boldsymbol{u}_o(k) \tag{25}$$

The state error covariance of the robot state $\boldsymbol{P}_r(k \mid k-1)$ is computed as follows:

$$\boldsymbol{P}_r(k \mid k-1) = \boldsymbol{J}_1(\boldsymbol{X}_r, \boldsymbol{u}_o)\boldsymbol{P}_r(k-1 \mid k-1)\boldsymbol{J}_1(\boldsymbol{X}_r, \boldsymbol{u}_o)^T + \boldsymbol{J}_2(\boldsymbol{X}_r, \boldsymbol{u}_o)\boldsymbol{U}_O(k)\boldsymbol{J}_1(\boldsymbol{X}_r, \boldsymbol{u}_o)^T \tag{26}$$

$\boldsymbol{J}_1(\boldsymbol{X}_r, \boldsymbol{u}_o)$ is the Jacobian of equation (16) with respect to the robot pose, $\boldsymbol{X}_r$ and $\boldsymbol{J}_2(\boldsymbol{X}_r, \boldsymbol{u}_o)$ is the Jacobian of equation (16) with respect to the control input, $\boldsymbol{u}_o$. Based on equations (12), the above Jacobians are calculated as follows:

$$\boldsymbol{J}_1(\boldsymbol{x}_1, \boldsymbol{x}_2) = \frac{\partial(\boldsymbol{x}_1 \oplus \boldsymbol{x}_2)}{\partial \boldsymbol{x}_1} \tag{27}$$

$$\boldsymbol{J}_1(\boldsymbol{x}_1, \boldsymbol{x}_2) = \begin{bmatrix} 1 & 0 & -x_2 \sin\theta_1 - y_2 \cos\theta_1 \\ 0 & 1 & -x_2 \cos\theta_1 - y_2 \sin\theta_1 \\ 0 & 0 & 1 \end{bmatrix} \tag{28}$$

$$\boldsymbol{J}_2(\boldsymbol{x}_1, \boldsymbol{x}_2) = \frac{\partial(\boldsymbol{x}_1 \oplus \boldsymbol{x}_2)}{\partial \boldsymbol{x}_2} \tag{29}$$

$$J_2(\boldsymbol{x}_1, \boldsymbol{x}_2) = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{30}$$

### 3.3.3 Observation

Assume that at a certain time $k$ an onboard sensor makes measurements (range and bearing) to $m$ features in the environment. This can be represented as:

$$\boldsymbol{z}_m(k) = [\boldsymbol{z}_1 \quad . \quad . \quad \boldsymbol{z}_m] \tag{31}$$

### 3.3.4 Update

The update process is carried out iteratively every $k^{th}$ step of the filter. If at a given time step no observations are available then the best estimate at time $k$ is simply the prediction $\boldsymbol{X}(k \mid k-1)$. If an observation is made of an existing feature in the map, the state estimate can now be updated using the optimal gain matrix $\boldsymbol{W}(k)$. This gain matrix provides a weighted sum of the prediction and observation. It is computed using the innovation covariance $\boldsymbol{S}(k)$, the state error covariance $\boldsymbol{P}(k \mid k-1)$ and the Jacobians of the observation model (equation 18), $\boldsymbol{H}(k)$.

$$\boldsymbol{W}(k) = \boldsymbol{P}(k \mid k-1)\boldsymbol{H}(k)\boldsymbol{S}^{-1}(k), \tag{32}$$

where $\boldsymbol{S}(k)$ is given by:

$$\boldsymbol{S}(k) = \boldsymbol{H}(k)\boldsymbol{P}(k \mid k-1)\boldsymbol{H}^{T}(k) + \boldsymbol{R}(k) \tag{33}$$

$\boldsymbol{R}(k)$ is the observation covariance.

This information is then used to compute the state update $\boldsymbol{X}(k \mid k)$ as well as the updated state error covariance $\boldsymbol{P}(k \mid k)$.

$$\boldsymbol{X}(k \mid k) = \boldsymbol{X}(k \mid k-1) + \boldsymbol{W}(k)\nu(k) \tag{34}$$

$$\boldsymbol{P}(k \mid k) = \boldsymbol{P}(k \mid k-1) - \boldsymbol{W}(k)\boldsymbol{S}(k)\boldsymbol{W}(k)^{T} \tag{35}$$

The innovation, $\boldsymbol{v}(k)$ is the discrepancy between the actual observation, $\boldsymbol{z}(k)$ and the predicted observation, $\boldsymbol{z}(k \mid k-1)$.

$$\boldsymbol{v}(k) = \boldsymbol{z}(k) - \boldsymbol{z}(k \mid k-1), \tag{36}$$

where $\boldsymbol{z}(k \mid k-1)$ is given as:

$$\boldsymbol{z}(k \mid k-1) = \boldsymbol{h}\left(\boldsymbol{X}_{r}(k \mid k-1), \boldsymbol{x}_{i}, \boldsymbol{y}_{i}\right) \tag{37}$$

$X_r(k \mid k-1)$ is the predicted pose of the robot and $(x_i, y_i)$ is the position of the observed map feature.

### 3.4 Incorporating new features

Under SLAM the system detects new features at the beginning of the mission and when exploring new areas. Once these features become reliable and stable they are incorporated into the map becoming part of the state vector. A feature initialisation function $\boldsymbol{y}$ is used for this purpose. It takes the old state vector, $\boldsymbol{X}(k \mid k)$ and the observation to the new

feature, $\boldsymbol{z}(k)$ as arguments and returns a new, longer state vector with the new feature at its end (Newman 2006).

$$\boldsymbol{X}(k|k)^* = \boldsymbol{y}\left[\boldsymbol{X}(k|k), \quad \boldsymbol{z}(k)\right] \tag{338}$$

$$\boldsymbol{X}(k|k)^* = \begin{bmatrix} \boldsymbol{X}(k|k) \\ x_r + r\cos(\theta + \psi_r) \\ y_r + r\sin(\theta + \psi_r) \end{bmatrix} \tag{39}$$

Where the coordinates of the new feature are given by the function $\boldsymbol{g}$:

$$\boldsymbol{g} = \begin{bmatrix} x_r + r\cos(\theta + \psi_r) \\ y_r + r\sin(\theta + \psi_r) \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \end{bmatrix} \tag{40}$$

$r$ and $\theta$ are the range and bearing to the new feature respectively. $(x_r, y_r)$ and $\psi_r$ are the estimated position and orientation of the robot at time $k$.
The augmented state vector containing both the state of the vehicle and the state of all feature locations is denoted:

$$\boldsymbol{X}(k|k)^* = [\boldsymbol{X}_r^T(k) \quad \boldsymbol{x}_{f,1}^T \quad . \quad . \quad \boldsymbol{x}_{f,N}^T] \tag{41}$$

We also need to transform the covariance matrix $\boldsymbol{P}$ when adding a new feature. The gradient for the new feature transformation is used for this purpose:

$$\boldsymbol{g} = \begin{bmatrix} x_r + r\cos(\theta + \psi_r) \\ y_r + r\sin(\theta + \psi_r) \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \end{bmatrix} \tag{42}$$

The complete augmented state covariance matrix is then given by:

$$\boldsymbol{P}(k|k)^* = \boldsymbol{Y}_{x,z} \begin{bmatrix} \boldsymbol{P}(k|k) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R} \end{bmatrix} \boldsymbol{Y}_{x,z}^T, \tag{43}$$

where $\boldsymbol{Y}_{x.z}$ is given by:

$$\boldsymbol{Y}_{x,z} = \begin{bmatrix} \boldsymbol{I}_{nxn} & \boldsymbol{0}_{nx2} \\ [\boldsymbol{G}_{x_r} \quad zeros(nstates - n)] & \boldsymbol{G}_z \end{bmatrix} \tag{44}$$

where *nstates* and $n$ are the lengths of the state and robot state vectors respectively.

$$\boldsymbol{G}_{X_r} = \frac{\partial g}{\partial X_r} \tag{45}$$

$$\boldsymbol{G}_{X_r} = \begin{bmatrix} \frac{\partial g_1}{\partial x_r} & \frac{\partial g_1}{\partial y_r} & \frac{\partial g_1}{\partial \psi_r} \\ \frac{\partial g_2}{\partial x_r} & \frac{\partial g_2}{\partial y_r} & \frac{\partial g_2}{\partial \psi_r} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -r\sin(\theta+\psi_r) \\ 0 & 1 & r\cos(\theta+\psi_r) \end{bmatrix} \tag{46}$$

$$\boldsymbol{G}_z = \partial g / \partial z \tag{47}$$

$$\boldsymbol{G}_z = \begin{bmatrix} \frac{\partial g_1}{\partial r} & \frac{\partial g_1}{\partial \theta} \\ \frac{\partial g_2}{\partial r} & \frac{\partial g_2}{\partial \theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta+\psi_r) & -r\sin(\theta+\psi_r) \\ \sin(\theta+\psi_r) & r\cos(\theta+\psi_r) \end{bmatrix} \tag{48}$$

### 3.5 Data association

In practice, features have similar properties which make them good landmarks but often make them difficult to distinguish one from the other. When this happen the problem of data association has to be addressed. Assume that at time $k$ , an onboard sensor obtains a set of measurements $\boldsymbol{z}_i(k)$ of $m$ environment features $\boldsymbol{E}_i(i=1,\ldots,m)$. Data Association consists of determining the origin of each measurement, in terms of map features $F_j, j=1,\ldots,n.$ The results is a hypothesis:

$$\boldsymbol{H}_k = \begin{bmatrix} j_1 & j_2 & j_3 \ldots\ldots j_m \end{bmatrix}, \tag{49}$$

matching each measurement $\boldsymbol{z}_i(k)$ with its corresponding map feature. $F_{ji}(j_i = 0)$ indicates that the measurement $\boldsymbol{z}_i(k)$ does not come from any feature in the map. Figure 2 below summarises the data association process described here. Several techniques have been proposed to address this issue and more information on some these techniques can be found in (Castellanos, J.A et al. 2006) and (Cooper, A.J, 2005).

Of interest in this chapter is the simple data association problem of finding the correspondence of each measurement to a map feature. Hence the Individual Compatibility Nearest Neighbour Method will be described.

### 3.5.1 Individual Compatibility

The IC considers individual compatibility between a measurement and map feature. This idea is based on a prediction of the measurement that we would expect each map feature to generate, and a measure of the discrepancy between a predicted measurement and an actual measurement made by the sensor. The predicted measurement is then given by:

$$\boldsymbol{z}_j(k|k-1) = \boldsymbol{h}(\boldsymbol{X}_r(k|k-1), x_j, y_j) \tag{50}$$

The discrepancy between the observation $\boldsymbol{z}_i(k)$ and the predicted measurement $\boldsymbol{z}_j(k|k-1)$ is given by the innovation term $\boldsymbol{v}_{ij}(k)$:

$$\boldsymbol{v}_{ij}(k) = \boldsymbol{z}_i(k) - \boldsymbol{z}_j(k|k-1) \tag{51}$$

The covariance of the innovation term is then given as:

$$\boldsymbol{S}_{ij}(k) = \boldsymbol{H}(k)\boldsymbol{P}(k|k-1)\boldsymbol{H}^T(k) + \boldsymbol{R}(k) \tag{52}$$

$\boldsymbol{H}(k)$ is made up of $\boldsymbol{H}_r$, which is the Jacobian of the observation model with respect to the robot states and $\boldsymbol{H}_{Fj}$, the gradient Jacobian of the observation model with respect to the observed map feature.

$$\boldsymbol{H}(k) = \begin{bmatrix} \boldsymbol{H}_r & 0\,0 & \cdots & 0\,0 & \boldsymbol{H}_{Fj} & 0\,0 \end{bmatrix} \tag{53}$$

Zeros in equation (53) above represents un-observed map features.

To deduce a correspondence between a measurement and a map feature, Mahalanobis distance is used to determine compatibility, and it is given by:

$$D_{ij}^2(k) = \boldsymbol{v}_{ij}^T(k)\boldsymbol{S}_{ij}^{-1}(k)\boldsymbol{v}_{ij}(k) \tag{54}$$

The measurement and a map feature can be considered compatible if the Mahalanobis distance satisfies:

$$D_{ij}^2(k) < \chi_{d,1-\alpha}^2 \tag{55}$$

Where $d = \dim(v_{ij})$ and $1-\alpha$ is the desired level of confidence usually taken to be $95\%$. The result of this exercise is a subset of map features that are compatible with a particular measurement. This is the basis of a popular data association algorithm termed Individual

Compatibility Nearest Neighbour. Of the map features that satisfy IC, ICNN chooses one with the smallest Mahalanobis distance (Castellanos, J.A et al. 2006).

### 3.6 Consistency of EKF-SLAM

EKF-SLAM consistency or lack of was discussed in (Castellanos, J.A et al. 2006), (Newman, P.M. (1999), (Cooper, A.J, 2005), and (Castellanos, J.A et al. 2006), It is a non-linear problem hence it is necessary to check if it is consistent or not. This can be done by analysing the errors. The filter is said to be unbiased if the Expectation of the actual state estimation error, $\tilde{\boldsymbol{X}}(k)$ satisfies the following equation:

$$E[\tilde{X}] = 0 \tag{56}$$

$$E\left[\left(\tilde{\boldsymbol{X}}(k)\right)\left(\tilde{\boldsymbol{X}}(k)\right)^{T}\right] \leq \boldsymbol{P}(k \mid k-1) \tag{57}$$

where the actual state estimation error is given by:

$$\tilde{\boldsymbol{X}}(k) = \boldsymbol{X}(k) - \boldsymbol{X}(k \mid k-1) \tag{58}$$

$\boldsymbol{P}(k \mid k-1)$ is the state error covariance. Equation (57) means that the actual mean square error matches the state covariance. When the ground truth solution for the state variables is available, a chi-squared test can be applied on the normalised estimation error squared to check for filter consistency.

$$\left(\tilde{\boldsymbol{X}}(k)\right)^{T}\left(\boldsymbol{P}(k \mid k-1)\right)^{-1}\left(\tilde{\boldsymbol{X}}(k)\right) \leq \chi^{2}_{d,1-\alpha} \tag{59}$$

where DOF is equal to the state dimension $d = \dim\left(x(k)\right)$ and $1-\alpha$ is the desired confidence level. In most cases ground truth is not available, and consistency of the estimation is checked using only measurements that satisfy the innovation test:

$$\boldsymbol{v}_{ij}^{T}(k)\boldsymbol{S}_{ij}^{-1}\boldsymbol{v}_{ij}(k) < \chi^{2}_{d,1-\alpha} \tag{60}$$

Since the innovation term depends on the data association hypothesis, this process becomes critical in maintaining a consistent estimation of the environment map.

## 4. Result and Analysis

Figure 19 below shows offline EKF SLAM results using data logged by a robot. The experiment was conducted inside a room of 900 cm x 720cm dimension with a few obstacles. Using an EKF-SLAM algorithm which takes data information (corners locations & odometry); a map of the room was developed. Corner features were extracted from the laser data. To initialize the mapping process, the robot's starting position was taken reference. In figure 19 below, the top left corner is a map drawn using odometry; predictably the map is skewed because of accumulation of errors. The top middle picture is an environment drawn using EKF SLAM map (corners locations). The corners were extracted using an algorithm we proposed, aimed at solving the possibility of mapping false corners. When a corner is re-

observed a Kalman filter update is done. This improves the overall position estimates of the robot as well as the landmark. Consequently, this causes the confidence ellipse drawn around the map (robot position and corners) to reduce in size (bottom left picture).

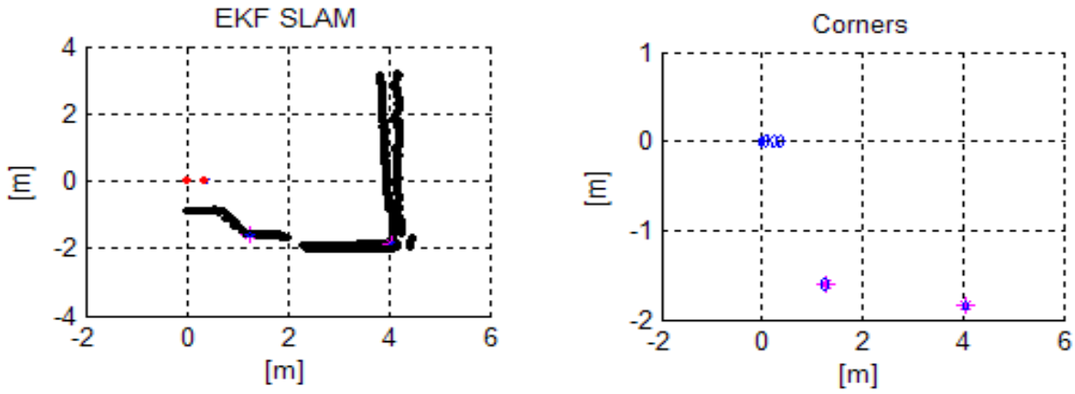



Fig. 18. In figure 8, two consecutive corner extraction process from the split and merge algorithm maps one corner wrongly, while in contrast our corner extraction algorithm picks out the same two corners and correctly associates them.

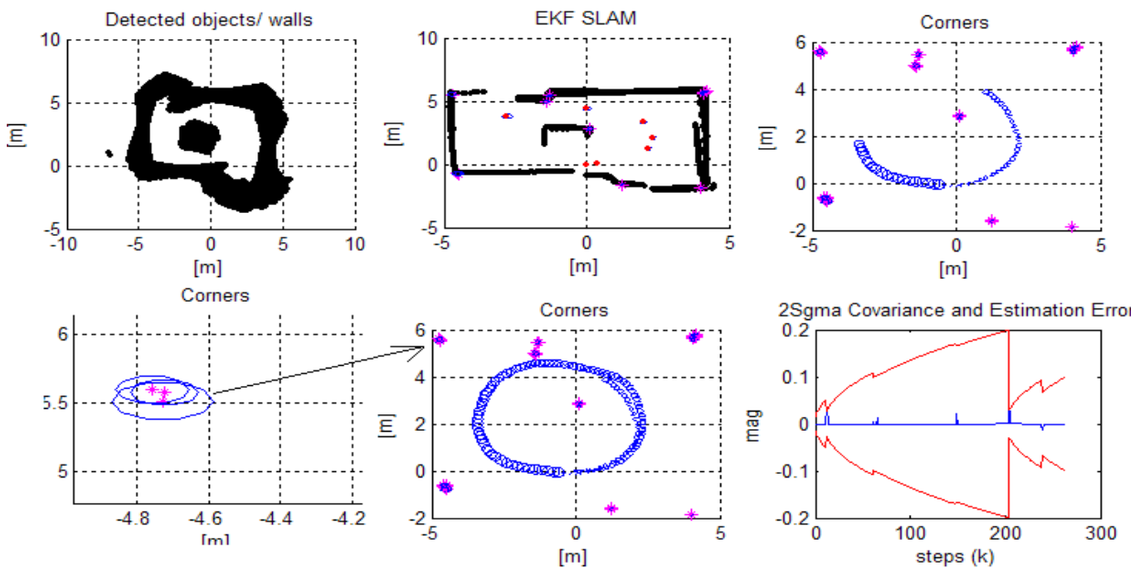



Fig. 19. EKF-SLAM simulation results showing map reconstruction (top right) of an office space drawn from sensor data logged by the Meer Cat. When a corner is detected, its position is mapped and a $2\sigma$ confidence ellipse is drawn around the feature position. As the number of observation of the same feature increase the confidence ellipse collapses (top right). The bottom right picture depict x coordinate estimation error (blue) between $2\sigma$ bounds (red). Perceptual inference

Expectedly, as the robot revisits its previous position, there is a major decrease in the ellipse, indicating robot's high perceptual inference of its position. The far top right picture shows a reduction in ellipses around robot position. The estimation error is with the $2\sigma$, indicating consistent results, bottom right picture. During the experiment, an extra laser sensor was

user to track the robot position, this provided absolute robot position. An initial scan of the environment (background) was taken prior by the external sensor. A simple matching is then carried out to determine the pose of the robot in the background after exploration. Figure 7 below shows that as the robot close the loop, the estimated path and the true are almost identical, improving the whole map in the process.

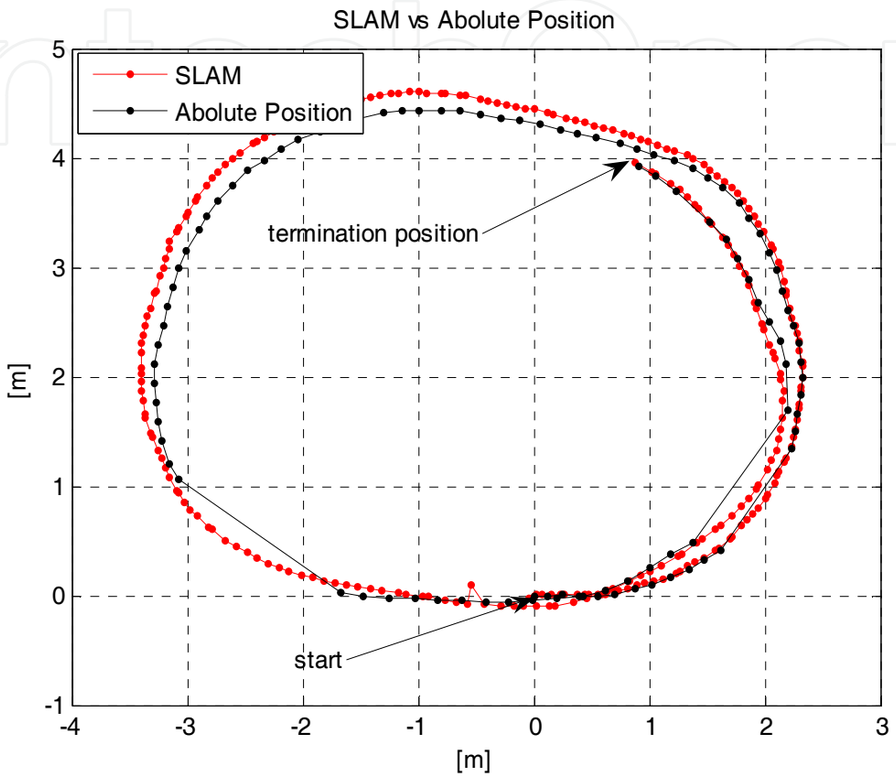



Fig. 20. The figure depicts that as the robot revisits its previous explored regions; its positional perception is high. This means improved localization and mapping, i.e. improved SLAM output.

## 5. Conclusion and future work

In this paper we discussed the results of an EKF SLAM using real data logged and computed offline. One of the most important parts of the SLAM process is to accurately map the environment the robot is exploring and localize in it. To achieve this however, is depended on the precise acquirement of features extracted from the external sensor. We looked at corner detection methods and we proposed an improved version of the method discussed in section 2.1.1. It transpired that methods found in the literature suffer from high computational cost. Additionally, there are susceptible to mapping 'ghost corners' because of underlying techniques, which allows many computations to pass as corners. This has a major implication on the solution of SLAM; it can lead to corrupted map and increase computational cost. This is because EKF-SLAM's computational complexity is quadratic the number of landmarks in the map, this increased computational burden can preclude real-

time operation. The corner detector we developed reduces the chance of mapping dummy corners and has improved computation cost. This offline simulation with real data has allowed us to test and validate our algorithms. The next step will be to test algorithm performance in a real time. For large indoor environments, one would employ a try a regression method to fit line to scan data. This is because corridors will have numerous possible corners while it will take a few lines to describe the same space.

## 6. Reference


Bailey, T and Durrant-Whyte, H. (2006), Simultaneous Localisation and Mapping (SLAM): Part II State of the Art. Tim. *Robotics and Automation Magazine*, September.

Castellanos, J.A., Neira, J., and Tard´os, J.D. (2004) Limits to the consistency of EKF-based SLAM. In *IFAC Symposium on Intelligent Autonomous Vehicles*.

Castellanos, J.A.; Neira, J.; Tardos, J.D. (2006). Map Building and SLAM Algorithms, *Autonomous Mobile Robots: Sensing, Control, Decision Making and Applications*, Lewis, F.L. & Ge, S.S. (eds), 1st edn, pp 335-371, CRC, 0-8493-3748-8, New York, USA

Collier, J, Ramirez-Serrano, A (2009)., "Environment Classification for Indoor/Outdoor Robotic Mapping," *crv, Canadian Conference on Computer and Robot Vision* , pp.276-283.

Cooper, A.J. (2005). A Comparison of Data Association Techniques for Simultaneous Localisation and Mapping, Masters Thesis, Massachusets Institute of Technology

Crowley, J. (1989). World modeling and position estimation for a mobile robot using ultrasound ranging. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*.

Duda, R. O. and Hart, P. E. (1972) "Use of the Hough Transformation to Detect Lines and Curves in Pictures," *Comm. ACM, Vol. 15*, pp. 11–15 ,January.

Durrant-Whyte, H and Bailey, T. (2006). *Simultaneous Localization and Mapping (SLAM): Part I The Essential Algorithms,* Robotics and Automation Magazine.

Einsele, T. (2001) "Localization in indoor environments using a panoramic laser range finder," Ph.D. dissertation, Technical University of München, September.

Hough ,P.V.C., Machine Analysis of Bubble Chamber Pictures. (1959). *Proc. Int. Conf. High Energy Accelerators and Instrumentation*.

Li, X. R. and Jilkov, V. P. (2003). Survey of Maneuvering Target Tracking.Part I: Dynamic Models. *IEEE Trans. Aerospace and Electronic Systems*, AES-39(4):1333.1364, October.

Lu, F. and Milios, E.E..(1994). Robot pose estimation in unknown environments by matching 2D range scans. In *Proc. of the IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 935–938.

Mathpages, "Perpendicular regression of a line" http://mathpages.com/home/kmath110.htm. *(2010-04-23)*

Mendes, A., and Nunes, U. (2004)"Situation-based multi-target detection and tracking with laser scanner in outdoor semi-structured environment", IEEE/RSJ Int. *Conf. on Systems and Robotics*, pp. 88-93.

Moutarlier, P. and Chatila, R. (1989a). An experimental system for incremental environment modelling by an autonomous mobile robot. In *ISER*.

Moutarlier, P. and Chatila, R. (1989b). Stochastic multisensory data fusion for mobile robot location and environment modelling. In *ISRR )*.

Newman, P.M. (1999). On the structure and solution of the simultaneous localization and mapping problem. *PhD Thesis, University of Sydney*.

Newman, P. (2006) EKF Based Navigation and SLAM, *SLAM Summer School*.

Pfister, S.T., Roumeliotis, S.I., and Burdick, J.W. (2003). Weighted line fitting algorithms for mobile robot map building and efficient data representation. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*.

Roumeliotis S.I. and Bekey G.A. (2000). SEGMENTS: A Layered, Dual-Kalman filter Algorithm for Indoor Feature Extraction. In Proc. *IEEE/RSJ International Conference on Intelligent Robots and Systems,* Takamatsu, Japan, Oct. 30 - Nov. 5, pp.454-461.

Smith, R., Self, M. & Cheesman, P. (1985). On the representation and estimation of spatial uncertainty. SRI TR 4760 & 7239.

Smith, R., Self, M. & Cheesman, P. (1986). Estimating uncertain spatial relationships in robotics, *Proceedings of the 2nd Annual Conference on Uncertainty in Artificial Intelligence, (UAI-86)*, pp. 435–461, Elsevier Science Publishing Company, Inc., New York, NY.

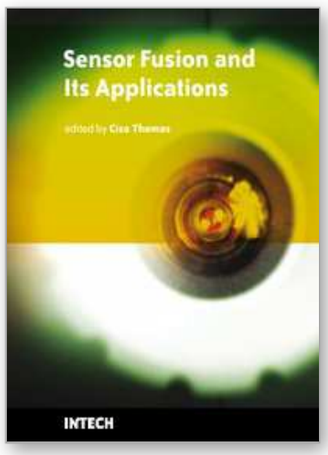
Spinello, L. (2007). Corner extractor, *Institute of Robotics and Intelligent Systems*, Autonomous Systems Lab, http://www.asl.ethz.ch/education/master/mobile_robotics/year2008/year2007, ETH Zürich

Thorpe, C. and Durrant-Whyte, H. (2001). Field robots. In *ISRR'*.

Thrun, S., Koller, D., Ghahmarani, Z., and Durrant-Whyte, H. (2002) Slam updates require constant time. Tech. rep., School of Computer Science, Carnegie Mellon University

Williams S.B., Newman P., Dissanayake, M.W.M.G., and Durrant-Whyte, H. (2000.). Autonomous underwater simultaneous localisation and map building. *Proceedings of IEEE International Conference on Robotics and Automation*, San Francisco, USA, pp. 1143-1150,

Williams, S.B.; Newman, P.; Rosenblatt, J.; Dissanayake, G. & Durrant-Whyte, H. (2001). Autonomous underwater navigation and control, *Robotica*, vol. 19, no. 5, pp. 481-496.




**Sensor Fusion and its Applications**

Edited by Ciza Thomas

ISBN 978-953-307-101-5
Hard cover, 488 pages
**Publisher** Sciyo
**Published online** 16, August, 2010
**Published in print edition** August, 2010

This book aims to explore the latest practices and research works in the area of sensor fusion. The book intends to provide a collection of novel ideas, theories, and solutions related to the research areas in the field of sensor fusion. This book is a unique, comprehensive, and up-to-date resource for sensor fusion systems designers. This book is appropriate for use as an upper division undergraduate or graduate level text book. It should also be of interest to researchers, who need to process and interpret the sensor data in most scientific and engineering fields. The initial chapters in this book provide a general overview of sensor fusion. The later chapters focus mostly on the applications of sensor fusion. Much of this work has been published in refereed journals and conference proceedings and these papers have been modified and edited for content and style. With contributions from the world's leading fusion researchers and academicians, this book has 22 chapters covering the fundamental theory and cutting-edge developments that are driving this field.



INTECH
open science | open minds

**InTech Europe**
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

**InTech China**
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen