University of Twente Students Journal Of Biometrics and Computer Vision 2017

MonoSLAM: A SINGLE CAMERA SLAM

Kirelloss Shokry, kirellossshokrysamirshokry@student.utwente.nl, s1818694 , Embedded System
Mohammed Alhawary, mohammedmostafaabdelghanialhawary@student.utwente.nl, s1786059,
Systems and Control

***Abstract*— Simultaneous Localization and Mapping (SLAM) became well established in the robotics community in the last decade and led to many innovations. This paper represents a monoSLAM algorithm, using a single camera as a sensor. The algorithm achieves both the localization of a RC car and building a full map of the track simultaneously in real time. A full map is drawn from sparse points using interpolation. One of the key contributions of this algorithm is that there is no need for any initial information about the width of the track, the positions of any landmarks, or the initial position of the RC car which makes it generic and suitable for different environments. Another main contribution is that although we depend here on a single camera, the depth could be estimated from the first frame knowing the height of the camera above the motion surface. Localization is achieved by tracking SURF points already initialized in the map, where the position of the car is updated using extended Kalman filter optimal estimation algorithm.**

***Index Terms*— 3D vision, SLAM, Extended Kalman filter, Tracking**

I. INTRODUCTION

Simultaneous localization and mapping (SLAM) is the process in which a robot builds a map of the surrounding environment and at the same time determines its location [4]. SLAM became well-established in the robotics community in the last ten years [1] with a wide range of applications such as self-driving cars and Google Cartographer system for indoor 3D mapping where there could be no available information from GPS. SLAM seems to be a chicken-egg problem, where accurate localization is essential for accurate mapping and accurate mapping is essential for accurate localization [2]. However, there are several known algorithms for solving it. This means that SLAM could be considered as a solved problem [3]. However, it is an active research area as new algorithm try to present more generic solutions and solve issues related to decreasing computational complexity, real-time applications and dynamic environments [4].

The use of the camera as a sensor for SLAM didn't receive enough attention until recently [1]. Some reasons might be: the high input rate, the lack of a direct depth estimation method and the difficulty of features extraction and tracking. On the other hand, the usage of a camera is cheap, accurate and well established because of the large image processing and computer vision research community [1].

The presented solution depends only on a single camera as a sensor, which simplifies the needed hardware. One of the main contributions of our algorithm is that it represents a generic solution for the problem as no initial information is needed about the environment or the navigating robot. This would be shown in section (II.A.1-8). Other main contribution, is that depth estimation of features takes place from the first frame with a single camera only. The main idea is to make use of the known camera fixed height above the motion surface as an additional information allow depth estimation using single camera. This allows entering the process of prediction and measurement from the first frame without the need for any initialization preprocessin. In this paper, only spare points are mapped, and interpolation is used to achieve full mapping.

The proposed solution consists of three major steps. These are developing a measurement model (section II.A.(1-5)), a motion prediction model (section II.A.6) and a combining optimal estimation algorithm (section II.A.7)). Camera calibration has been implemented to obtain camera intrinsic parameters. Consequently, a depth estimation algorithm has been developed given the camera pinhole model and the fixed height of the camera above the motion surface. Regions of interest in the image have been extracted via a Fourier-based template matching algorithm. Unique landmark points included are extracted using a Shi & Tomasi corner detector and then tracked between frames using a SURF-feature matching algorithm. The resultant measurement data are combined with a velocity model-based prediction of the vehicle motion using an extended Kalman filter optimal estimation algorithm.

II. METHODS AND MATERIALS

A. Materials

In this section, the materials used in our algorithm of "MonoSLAM" are discussed, explaining their mathematical background and how they were modified to be incorporated in the project.

1) Measurement model depending on camera pinhole model

Camera calibration was performed to obtain the camera intrinsic parameters, expressed in matrix "K".

$$K = \begin{bmatrix} f & s & u \\ 0 & \alpha f & v \\ 0 & 0 & 1 \end{bmatrix} \quad (1)$$

Where, "f" is the focal length of the camera. "s" is the skew factor. "α" is the ratio between the size of pixels in y and x direction. (u, v) is the principle point, which is the intersection of the optical axis with the image plane.

The measurement model was used in mapping by determining the 3D world co-ordinates of sparse points. Also, it was used for localization by tracking points already initialized in the map to determine the car translation and rotation. The pinhole camera model was the basis of the measurement model. This model relates the position of a point (X, Y, Z) in 3D camera co-ordinate system to its (r, c) position in the image as follows:

$$c = \frac{X*f}{Z} + u \quad (3)$$

$$r = \frac{Y*f}{Z} + v \quad (4)$$

The origin of the world co-ordinates is the same as the origin of the camera co-ordinates only in the first frame. Yet, this model just represents the 3D camera co-ordinates of a point but provides no information about its 3D world co-ordinates if the world co-ordinates are not aligned with the camera co-ordinates. That's why this model is extended with the rotation matrix representing the possible steering of the vehicle around the y-axis with angle "Θ" as follows:

$$\begin{bmatrix} c \\ r \\ 1 \end{bmatrix} = K. \begin{bmatrix} cos(\theta) & 0 & -sin(\theta) \\ 0 & 1 & 0 \\ sin(\theta) & 0 & cos(\theta) \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (5)$$

The translation of the RC car as well as the car rotation should be considered by the model for proper localization.

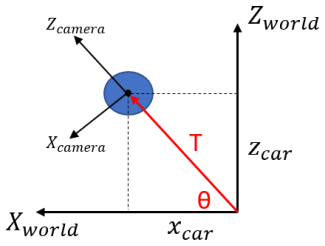

Figure 1: CCS with respect to WCS

The car has two degrees of freedom: steering with angle "Θ" with the x-axis as has been shown and translation "T" along its heading. "T" could be decomposed along the x-axis and z-axis and the model represented in (5) could be extended as follows:

$$\Delta X = T cos(\theta) = x_{car}$$
$$\Delta Z = T sin(\theta) = z_{car}$$

Thus, it is needed to include the translation of the origin of the camera coordinate system in the pinhole camera model. The following equation shows this impact:

$$\begin{pmatrix} r \\ c \end{pmatrix} = K * [I_{3x3}|0_{3x1}] * \begin{pmatrix} \cos(\theta) & 0 & -\sin(\theta) & x_{car} \\ 0 & 1 & 0 & 0 \\ \sin(\theta) & 0 & \cos(\theta) & z_{car} \\ 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \quad (6)$$

The main challenge in this project is that utilizing the position of a point in image is not sufficient to directly calculate its 3D world co-ordinates and initialize it in the map. However, the planar property of the surface where the car translates implies that the Y coordinate of each point on this plane will be fixed all the time and it is equal to the height on the camera above that plane h. Thus, the measurement model will be:

$$\begin{pmatrix} r \\ c \end{pmatrix} = K * [I_{3x3}|0_{3x1}] * \begin{pmatrix} \cos(\theta) & 0 & -\sin(\theta) & x_{car} \\ 0 & 1 & 0 & 0 \\ \sin(\theta) & 0 & \cos(\theta) & z_{car} \\ 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} X \\ h \\ Z \\ 1 \end{pmatrix}$$

The use of this measurement model in mapping and localization will be explained in more details in section III.

2) Pattern Recognition

Landmarks must represent patterns lying on the planer surface where the vehicle translates. Thus, the successive alternating black and white areas on the track border has been selected to represent landmarks as shown in figure 2.

Figure 2: Example patterns for Fourier-based template matching

Fast-Fourier-Transform-Based correlation algorithm will be used to locate the landmark pattern in each extracted video frame. The algorithm can be described by the following steps:

- Obtain the dot product of the FFT for the template and the image.
- Obtain the inverse FFT of the resultant image.
- Find the maximum correlation value in the image.
- Set a suitable threshold as a portion of the maximum correlation value.
- Dilate image locations greater than the threshold to extract desired pattern.

The previous algorithm will lead to extracting each pattern occurring in the image. Two left and right ROIs should be windowed from the extracted pattern. The canter of each ROI by implementing an area-based template matching between the template and the image. This is achieved using TemplateMatcher library provided by MATLAB CV toolbox. The algorithm will result in two patterns left and right centres. Finally, a window around each centre is selected. The dimensions of each window are tuned manually based on the projected track image captured by the camera.

3) Shi & Tomasi corner detection

The output of the previous algorithm is two left and right ROIs containing potential corner points to be extracted. In order to extract corner points, a Shi and Tomasi Minimum Eigenvalue corner detection algorithm is adopted and implemented using detectMinEigenFeatures() MATLAB function. The output of this function is the XY location of the interest corner points.

4) Outliers Removal

There is a possibility that the corner detection algorithm spots wrong corner points. This motivates developing an outlier removal algorithm. The criteria of evaluating the corner points is that their locations fit in a common straight line. If one or more points do not fit in the straight line formed by most of the

points, they are considered as outliers and popped out of the corners extracted vector. These are the rational steps of the outlier removal algorithm:

- Separate the corners locations into two x and y data vectors.
- Select the baseline fitting model to be linear.
- Evaluate the function value of the x data vector based on the selected baseline model.
- Set an evaluation threshold as a function of the standard deviations of the y data vectors.
- Identify outliers as points at a distance greater than the threshold.
- Exclude the outliers from the corners vector.

The output of this algorithm is filtered corner points extracted from planar patterns and can be considered as qualified landmarks. The locations values of these points are substituted in the camera model to find out their XYZ location with respect to the camera coordinate system.

5) SURF feature extraction matching

It is essential to determine the POSE of the camera coordinate system with respect to the world coordinate system. Re-observing static landmarks is required to localize the camera coordinate system based on their new observed locations. Thus, it is essential to track each observed landmark between frames. In our case, our extracted corner points can be tracked from one frame to another by matching their describing features. Thus, it is vital to select the most appropriate feature descriptor. After testing all the descriptor extraction methods provided by the extractFeature() MATLAB function, the decision has been made up to use SURF (Speeded-Up Robust Features) descriptor. The function takes as an input the intensity image and the interest corner points to be tracked. Then, it extracts SURF points in a window around each corner point. The feature descriptor for each point is the orientation of these extracted SURF points relative to it in radian. The output of the function is the extracted feature vectors and the SURF points. In the following frame, the process is repeated to produce feature vectors of the extracted corner points in this frame. Finally, extracted features from each two successive frames are compared using matchFeatures () MATLAB function. The method adopted by the function to match features is the 'Exhaustive' method. In this method, the algorithm computes the pairwise distance between feature vectors in the given feature vectors pair. The output of this function is the indices of the corresponding matched features between the given feature vectors pair (figure 3).

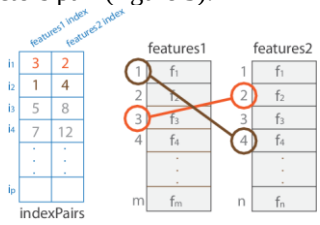

Figure 3: Matched indices between two features sets

Finally, the two points that each matching index refers to their features are identified as matched and hence tracked points (figure 4).

Figure 4: Matched visual features between two images

6) Motion model

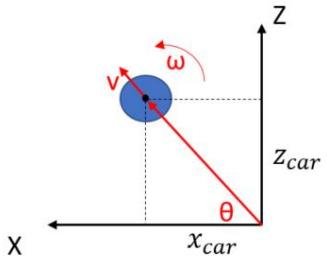

Figure 5: Velocity-based car motion model

A velocity-based motion model is used to predict the car's POSE while moving inside the XZ plane. Thus, the model provides predictions for the state vector $[x, z, \theta]^T$ given the car's velocity input vector $[v\ \omega]^T$. By analysing the figure 5, the motion model to be used for prediction is:

$$\begin{pmatrix} x_{new} \\ z_{new} \\ \theta_{new} \end{pmatrix} = \begin{pmatrix} x_{old} \\ z_{old} \\ \theta_{old} \end{pmatrix} + \begin{pmatrix} -\frac{v}{\omega}\sin(\theta) + \frac{v}{\omega}\sin(\theta + \omega\Delta t) \\ -\frac{v}{\omega}\cos(\theta) - \frac{v}{\omega}\cos(\theta + \omega\Delta t) \\ \omega\Delta t \end{pmatrix}$$

The motivation to use this model is that it is feasible to measure the average linear velocity v_{avg} and the average angular velocity ω_{avg} each second while conducting the experiment then use them as inputs to the model.

7) *Extended Kalman Filter*

Kalman filter is an algorithm used to refine noisy camera measurements over time and produce corrected estimate for each unknown variable the camera measures. This is based on predictions obtained from the motion model. Kalman filter provides a joint probability distribution over each state variable in the model. It provides the optimal combination between information obtain through model predictions and measurements to achieve more accurate estimates. The theory guarantees converging to better accurate measurements since it reduces the model dependencies on measured points with higher uncertainties. In this project, an extended Kalman filter is used because both the motion model and the measurement model are non-linear.

It is required to define a vector containing the state variables wanted to be estimated. In a SLAM problem for a mobile robot moving over a 2D plane, the state vector must represent information about both the car and the landmarks. Thus,

$$\mu_t = [\, x_{car} z_{car} \theta_{car} l_{1x} l_{1y} \dots l_{nx} l_{ny}]$$

Where n is the number of observed landmarks. Assuming stationary landmarks, we depend on the motion model represented in section II.A.6 to provide a prediction about the mean of the state variables.

$$\bar{\mu}_t = g(u_t, \mu_{t-1})$$

Where,

$$u_t = [v\ \omega]^T$$

The uncertainty of each state variable is represented by the covariance matrix Σ

$$\Sigma = \begin{pmatrix} \Sigma_{xx} & \Sigma_{xl} \\ \Sigma_{lx} & \Sigma_{ll} \end{pmatrix}$$

Then the predicted covariance that can be estimated from the model is

$$\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$$

Where G_t is the Jacobean of the motion model $g(\mu_t, \mu_{t-1})$ and the R_t is the process noise matrix. Initial predictions for the mean and the covariance for each state variable $\bar{\mu}_t$and $\bar{\Sigma}_t$have been obtained from the motion model. It is required to correct these predictions optimally using data obtained by the camera. The measurement $\tilde{z}_t$ is related to the state variables by the equation (6). Thus

$$\tilde{z}_t = h(\bar{\mu}_t)$$

The Kalman optimal gain $K_t$is

$$K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1}$$

Where $H_t$is the Jacobean of the measurement model $h(\bar{\mu}_t)$ and Q_t is the measurement noise matrix. Now predictions and measurements are combined optimally using K_t

$$\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$$
$$\Sigma_t = (I - K_t H_t)\ \bar{\Sigma}_t$$

8) Environment model

Available information about track geometry can be used to assist the measurement model. The width of the track can be used to set upper limits for the solver that solves for the x coordinate of each visual landmark. Similarly, the depth of the track can be used to set upper limits for the z coordinate of each visual landmark. Moreover, the height of the camera coordinate system above the XZ plane (h) is known and will be substituted in the measurement model. In fact, this height is the y coordinate of any detected planar point inside the map. Substituting it in the measurement model will allow to calculate the XZ coordinates of any planar point given its pixel coordinates $[r, c]^T$.

B. Methods

1) Visual landmarks extraction and tracking

As mentioned in the previous section, Visual landmarks are corner points lying on planar patterns extracted from the left and the right sides of the car track. These corner points locations in the image plane are fed to the pinhole camera model to calculate the XYZ coordinates of the track border points solving the mapping problem. Moreover, the localization is achieved by retrieving the vehicles POSE relative to re-observed landmarks. Thus, tracking the extracted corner points allows to feed their current and previous locations on the image plane to the pinhole camera model and hence retrieve the change in the car's POSE. Below, pseudo code of the adopted algorithm is presented:

Algorithm visual landmarks tracking
Input RC care race video
Output Pixel coordinates of tracked points lying on the track two sides

- Extract suitable pattern template.
- **FOR** Each frame in the video **DO**
  - **Evaluate** correlation in the Fourier domain Re[IFFT(FFT(frame). FFT(template))]
  - **Set** a threshold T to be portion of the max correlation value.
  - **Dilate** around image locations with correlation value > T
  - **Extract** the centroid of each dilated region.
  - **Extract** a ROI around each centroid using a window with tunable dimensions.
  - **Apply** the template matcher object over each extracted ROI.
  - **Extract a** ROI around each matched point given by the template matcher.
  - **Detect** Shi & Tomasi corners in each ROI
  - **For** each extracted corner set **DO**
    - **IF** size (corners vector) > 0 **THEN**
      - **Separate** x and y coordinates
      - **Evaluate** a straight line fitting most the data
      - **Exclude outliers**
  - **Concatenate** all filtered corners in 1 vector
  - **Extract** current_f SURF feature descriptors around each corner.
  - **IF** (Frame number > 1) **THEN**
    - **Match** current_f and previous_f
    - **Obtain** matched points locations out of matched features indices.
  - **previous_f = current_f**

2) Car localization

When the vehicle moves on the XZ plane, the camera coordinate system moves with respect to the world coordinate system. Thus, the origin of the camera coordinate system expressed in the world CS to be:

$$O_{ccs} = \begin{pmatrix} x_{car} \\ 0 \\ z_{car} \end{pmatrix}$$

Thus, it is essential to localize the car in the world coordinate (solve for $[x_{car}, z_{car}]$) to keep track of the moving Camera CS while constructing the map. By having the origin vector of the camera CS O_{ccs}, it will be possible to construct the map of landmarks locations in the world coordinate taken into account the movement of the Camera CS. We will depend on the following pinhole camera model:

$$\begin{pmatrix} r_{lm} \\ c_{lm} \end{pmatrix} = K * [I_{3x3}|0_{3x1}] * \begin{pmatrix} \cos(\theta) & 0 & -\sin(\theta) & x_{car} \\ 0 & 1 & 0 & 0 \\ \sin(\theta) & 0 & \cos(\theta) & z_{car} \\ 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} x \\ h \\ z \\ 1 \end{pmatrix}$$

The vector $[r_{lm}, c_{lm}]^T$is the location of the landmark in the pixel coordinate system extracted by the previous algorithm. The vector $[x, h, z, 1]^T$ represents the homogenous coordinate

of the landmark in the world coordinate. Θ is the heading of the vehicle with respect to the Z axis of the world coordinate, and is related to the car location by:

$$\theta = \tan^{-1}(\frac{z_{car}}{x_{car}})$$

Thus, for each landmark with known location in the world coordinate and if correctly tracked, solving the previous nonlinear algebraic equation will result in calculating the car location in the world CS. The following algorithm puts a framework to obtain the car XZ location in the world CS given a group of tracked visual landmarks.

Algorithm Car localization
Input Pixel coordinates of tracked visual landmarks and their XZ location in the world coordinate.
Output The measured car location $[x_{car}, z_{car}, \theta]^T$

IF (Frame number > 1) **THEN**
- Extract the world XZ coordinates **ONLY** for each tracked visual landmark. To form the following pair for each point
$$p_i: [r_{new} c_{new}]^T \rightarrow [X_{wcs}\ Z_{wcs}]^T$$
- **FOR** each tracked visual landmark **DO**:
  - Substitute the pair p_i in the pinhole camera model
  - Set the upper and lower limits to the solver for x_{car} and z_{car} variables to be:
    - $x_{car}: \left[-\frac{w}{2} \rightarrow \frac{w}{2}\right], w: track\ width$
    - $z_{car}: [z_{current} \rightarrow z_{current} + v]$ $, v: expected\ velocity\ in\ cm/frame$
  - Solve the nonlinear pinhole camera model for x_{car} and z_{car} variables.
- Take the average of the calculated x_{car} and z_{car} from each i tracked landmark to obtain the final car location:
  - $x_{current} = \sum_i x_{car}$
  - $z_{current} = \sum_i z_{car}$

3) Landmarks mapping

It is intuitive that new visual landmarks are detected and extracted while the vehicle is moving. Thus, map of the XZ world coordinates of each newly detected landmark should be constructed. This can be obtained only localizing the vehicle in the map to calculate the landmark locations based on the new car location using the pinhole camera model:

$$\begin{pmatrix} r_{lm} \\ c_{lm} \end{pmatrix} = K * [I_{3x3}|0_{3x1}] * \begin{pmatrix} \cos(\theta) & 0 & -\sin(\theta) & x_{car} \\ 0 & 1 & 0 & 0 \\ \sin(\theta) & 0 & \cos(\theta) & z_{car} \\ 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} x \\ h \\ z \\ 1 \end{pmatrix}$$

Now, the car location $[x_{car}, z_{car}, \theta]^T$ is known from the localization algorithm. The pixel coordinate of each visual landmark $[r_{lm} c_{lm}]^T$ is determined by the visual landmark extraction algorithm. Thus, solving the pinhole camera model will result in the XZ world coordinate for each detected visual landmark.

Algorithm Landmarks mapping
Input Pixel coordinates $[r_{lm} c_{lm}]^T$ of tracked visual landmarks and the car location in the world CS $[x_{current} z_{current}]$.
Output Visual landmarks XZ coordinates in the world CS

- **IF** (Frame number ==1) **THEN**
  - Set car location $[x_{car} z_{car} \theta]^T = [0\ 0\ 0]^T$

  ELSE
  - Use $[x_{car} z_{car} \theta]^T$ obtained from localization algorithm.
- **FOR** each detected landmark on the left side **DO**:
  - Substitute the car location $[x_{car} z_{car} \theta]^T$ and the pixel coordinate $[r_{lm} c_{lm}]^T$ in the pinhole camera model.
  - Set the upper and the lower limits to the solver for the XZ coordinates of the left-sided landmark:
    - $X: \left[x_{car} \rightarrow \frac{w}{2}\right], w: track\ width$
    - $Z: [z_{car} \rightarrow d], d: track\ depth$
  - Solve the nonlinear pinhole camera model for X and Z variables.
  - Store the XZ coordinates together with the pixel coordinates of the landmark in 1 vector to be used in the localization algorithm.
    - $[r_{lm} c_{lm} X\ Z]$
- **FOR** each detected landmark on the right side **DO**:
  - Substitute the car location $[x_{car} z_{car} \theta]^T$ and the pixel coordinate $[r_{lm} c_{lm}]^T$ in the pinhole camera model.
  - Set the upper and the lower limits to the solver for the XZ coordinates of the right-sided landmark:
    - $X: \left[-\frac{w}{2} \rightarrow x_{car}\right], w: track\ width$
    - $Z: [z_{car} \rightarrow d], d: track\ depth$
  - Solve the nonlinear pinhole camera model for X and Z variables.
  - Store the XZ coordinates together with the pixel coordinates of the landmark in 1 vector to be used in the localization algorithm.
    - $[r_{lm} c_{lm} X\ Z]$

4) Kalman filter estimation

To uniquely identify a landmark, an ID is assigned to it once it appears to the scene. Given any new landmark observation

$$z_t = [l_x l_z, Id]$$

It is required to use the motion model to correct the estimation of this observation based on Kalman filer. By defining the state variables vector to be

$$\mu_t = [\ x_{car} z_{car} \theta_{car} l_{1x} l_{1z} \dots l_{nx} l_{nz}]$$

With covariance matrix

$$\Sigma = \begin{pmatrix} \Sigma_{xx} & \Sigma_{xl} \\ \Sigma_{lx} & \Sigma_{ll} \end{pmatrix}$$

Initially we know the exact location of the car with 0 uncertainty. Thus

$$\mu_0 = [0,0,0]$$
$$\Sigma_0 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

For each new observed landmark, we augment the μ and Σ matrices. The number of landmarks at each time step is N_t.

Algorithm Kalman filter estimation
Input $\mu_{t-1}, \Sigma_{t-1}, z_t, N_{t-1}$
Output μ_t, Σ_t, N_t

- Calculate the prediction $\bar{\mu}_t$from the motion model (section II.A.6)

$$\bar{\mu}_t = g(u_t, \mu_{t-1})$$

- Calculate the Jacobian of the motion model

$$G_t = \frac{\partial g}{\partial \mu_t}$$

- Calculate the covariance prediction

$$\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$$

- **For** each observed landmark z_t^i in z_t**DO**
  - Check z_t^i Id
  - **IF** (new landmark) **THEN**
    - Add z_t^i to $\bar{\mu}_t$. $\bar{\mu}_t = [\bar{\mu}_t, z_t^i]^T$
    - Augment $\bar{\Sigma}_t$

$$\bar{\Sigma}_t = \begin{pmatrix} \bar{\Sigma}_t & 0_{2,1} \\ 0_{1,2} & \infty_{2,2} \end{pmatrix}$$

    - $N_t = N_{t-1} + 1$
  - Calculate measurement prediction form measurement model (equation 6)

$$\check{z}_t^i = h(\bar{\mu}_t)$$

  - Calculate the Jacobean of the measurement model

$$H_t^i = \frac{\partial h}{\partial \mu_t}$$

  - Calculate the Kalman gain K_t^i

$$K_t^i = \bar{\Sigma}_t H_t^{iT} \left(H_t^i \bar{\Sigma}_t H_t^{iT} + Q_t \right)^{-1}$$

  - Calculate corrected $\bar{\mu}_t, \bar{\Sigma}_t$

$$\bar{\mu}_t = \bar{\mu}_t + K_t^i (z_t^i - \check{z}_t^i)$$
$$\bar{\Sigma}_t = (I - K_t^i H_t^i) \bar{\Sigma}_t$$

- Return μ_t, Σ_t

$$\mu_t = \bar{\mu}_t$$
$$\Sigma_t = \bar{\Sigma}_t$$

2) Performance evaluation:

To evaluate the implemented algorithm, a real race track was built to provide a guidance for the car motion (figure 6). The real experiment can be viewed using the video attached to this report.

Figure 6: The track used for the real experiment

The parameters of this experiment are presented in following table

Parameter	**Value**
Camera height (h)	14 cm
Track width (w)	10 cm
Average lineal velocity (v_l)	$6cm/s$
Average angular velocity (ω)	$0.02\ rad/s$

Table 1: Real Experiment parameters.

a) Evaluating visual landmarks tracking*:*

For proper operation of the algorithm, each corner point on the track sides should be detected and tracked properly between frames.

I. Experiment Protocol: Proper tracking was evaluated by marking all the detected points in "green" in one image frame and marking them in "red" in the next frame.

II. Outcome Evaluation: A proper tracking is detected if each red point points correctly at the corresponding green point.

b) Evaluating Car localization:

For proper car localization, the car coordinates in the XZ plane must be properly estimated in *cm* at each frame given tracked visual features.

I. Experiment Protocol: Car localization was evaluated by storing the estimated location in the XZ plane and plotting it with time.

II. Outcome Evaluation: A proper car localization is detected if the estimated car location corresponds to the true car trajectory within the experiment.

c) Evaluating Landmarks mapping:

For proper landmark mapping, the coordinate of each visual landmark must be properly estimated in *cm* at each frame.

I. Experiment protocol: Landmarks estimated locations in world coordinates are stored and plotted with time.

II. Outcome Evaluation: A proper landmark mapping is achieved if the plotted map corresponds to the geometry of the real track sides.

d) Evaluating Kalman filter estimation:

For optimal estimation the estimated value for each landmark position must converge to its true value with decaying in its uncertainty value.

I. Experiment Protocol: The algorithm is applied to each observed points and plots of corrected measured landmark are created.

II. Outcome Evaluation: A proper Kalman

filter estimation is achieved if the landmark locations converge to their actual values with time.

III. Results

A. *Experimentation results*:

The first experiment was implemented successfully. All different components of the algorithm were validated. Proper ROI were correctly selected on both sides of the track as shown in figure 7. Corner points were detected and tracked from one frame to the next (figure 8).

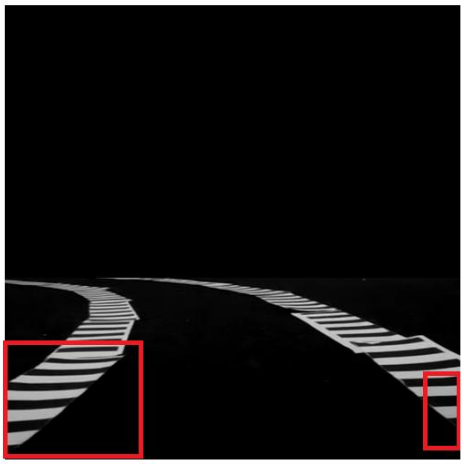

Figure 7: Extracted left and right ROIs using Fourier based template matching

Figure 8: Tracked visual features between two successive frames using SURF-features matching

The result of the localization experiment shows that the car movement in the z-Axis directions is estimated correctly (figure 9). However, the detected motion along the x-Axis direction is scaled with respect to the true motion of the car. The mapping experiment directly depends on the quality of the localization experiment. It provides accurate estimate for the map along the x-Axis direction. It always estimates a track width of 10 cm. It also provides accurate estimate of the curvature shape, but the map points are mapped to-scale with respect to their true locations (figure 10). This problem specially occurs with further landmarks. The quality of mapping directly influences the localization. Thus, these inaccuracies cause the localization drift in the x-Axis direction in the beginning of the motion (figure 9).

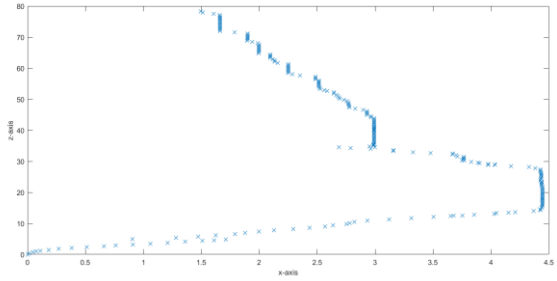

Figure 9: The retrieved car trajectory using the developed SLAM algorithm

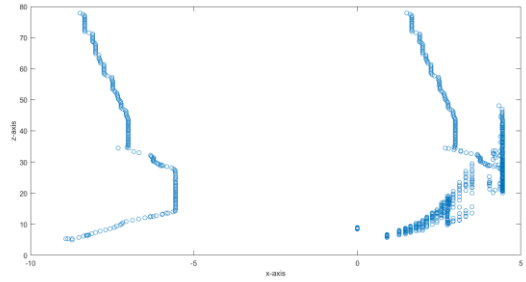

Figure 10: The constructed map using the developed SLAM algorithm

The last experiment showed that integrating Kalman filter in the SLAM algorithm did not improve the quality of the SLAM results. These reasons for this result are analyzed in the discussion section (IV)

B. *Project outcome*:

This project provides a complete algorithm structure to implement a single-camera SLAM algorithm for race cars using MATLAB. The first algorithm "Visual landmarks extracting and tracking" succeeds in locating the pattern of each track side in the image using Fourier-based pattern recognition technique. Then it extracts Shi & Tomasi corner points and tracking them using SURF feature matching approach.

The second algorithm "Car localization" provides a modified version of the pinhole camera model that considers the movement of the camera in the world coordinate system. This modified model succeeds in retrieving a scaled trajectory of the can motion above a planar surface.

The third algorithm "Landmarks mapping" uses the modified pinhole camera model to map extracted landmark corners with respect to the moving camera. It succeeds in providing a preliminary map of the race track.

The last algorithm “Kalman filter estimation” did not succeed in improving the quality of the SLAM algorithm for the reasons discussed in the following section.

IV. Discussion

The implemented algorithm results in generating scaled map and car trajectory. These are some reasons that are believed to cause this problem. The first reason is that there are uncertainties in values of the intrinsic parameter matrix of the camera “K”. The calibration experiment was done several times using MATLAB camera calibrator app. The parameters obtained from the calibration experiment were used to estimate the depth of a single point of a known location. Each time, the estimated location was inaccurate. This inaccuracy increases if the point is getting further from the camera.

The idea used to estimate the depth using a single camera assumes that the origin of the camera CS keeps its height above the surface of motion. However, this assumption was sometimes violated while conducting the experiment since the camera tends to tilt to the back while the car is moving forward. This violation results in unmodeled rotational motion and leads to inaccuracies while estimating the depth of the map points.

Another problem was the similarity between all the detected visual landmarks. Since they are all represent a black-while alternating patterns. This causes mismatch between corner points which results in errors while estimating the pose of the car.

The integration of an extended Kalman filter algorithm did not improve the quality of the measured locations. The reason for this failure is that the algorithm assumes reobservation of the same landmark for a sufficient period to allow the algorithm converges to the true value of its location with increasing confidentiality. However, in our case, each observed landmark does not appear in the scene for sufficient frames. Each visual landmark disappears from the scene while the vehicle is moving forward. Thus, the covariance of the state representing its location does not decay sufficiently. This covariance increases with time since all new observed landmarks have also high uncertainty leading the algorithm to diverge.

V. Conclusion

This paper leads a first step to make a great leap forward a single-camera SLAM. The major advantages of this algorithm are that there is no need for any initial information about the width of the track, the positions of any landmarks, or the initial position of the RC car which makes it generic and suitable for different environments. The key contribution is that although we depend here on a single camera, the depth could be estimated from the first frame knowing the fixed height of the camera above the motion surface of the car.

The paper presents a modified measurement model that incorporates the car planar motion into the pinhole camera model. It also shows the steps implemented to extract visual landmarks from the race track to implement the SLAM algorithm. The extracted landmarks together with the measurement model are used to estimate the car trajectory and the map track simultaneously. Due to real-time experiment limitations presented in IV, the map and the trajectory obtained are to-scale.

Finally, the paper presents a velocity-based model to the motion of the vehicle and investigates the use of an extended Kalman filter algorithm to improve the quality of the measurement. Testing the algorithm showed that it did not provide the expected improvements since the requirements needed for it to converge are not met in our SLAM measurement system.

References

[1] Davison, Andrew J., et al. "MonoSLAM: Real-time single camera SLAM."
IEEE transactions on pattern analysis and machine intelligence 29.6 (2007): 1052-1067.

[2] Mur-Artal, Raul, Jose Maria Martinez Montiel, and Juan D. Tardos.
"ORB-SLAM: a versatile and accurate monocular SLAM system."
IEEE Transactions on Robotics 31.5 (2015): 1147-1163.

[3] Se, Stephen, David Lowe, and Jim Little.
"Vision-based mobile robot localization and mapping using scale-invariant features."
Robotics and Automation, 2001. Proceedings 2001 ICRA.
IEEE International Conference on. Vol. 2. IEEE, 2001.

[4] Bailey, Tim, and Hugh Durrant-Whyte.
"Simultaneous localization and mapping (SLAM): Part II."
IEEE Robotics & Automation Magazine 13.3 (2006): 108-117..