COMP0130 - ROBOT VISION AND NAVIGATION

Coursework 2: Sparse Feature-Based Visual SLAM

Name: **Yinji Zhu** ID: **18062537** Email: **ucaby22@ucl.ac.uk**

March 15, 2019

Introduction

The assignment is divided into two parts. In the first part, the implementation of the odometry for both the Kalman filter-based and iSAM based localization systems will be completed and compared [1]. After that, the implementations of the GPS-based fusion for both the Kalman filter and the GTSAM system will be completed and compared with the situation without GPS measurements [1]. In the second part, a SLAM system will be built based on a GTSAM based localization system [1]. After that, the comparison with odometry and GPS will be done. Firstly, the effects of enabling and disabling odometry will be compared [1]. Secondly, the effect of adding GPS measurements will be compared [1].

Part 1: Odometry and GPS Fusion

Task 1.1: Odometry

In this part, the odometry for both the Kalman filter-based and iSAM based localization systems were implemented. Functions were written in *answers.kalman.OdometryOnlyLocalizationSystem* and *answers.gtsam.OdometryOnlyLocalizationSystem* classes respectively.

Methods

Knowing that the vehicle state is a camera-based type, it is represented a Pose3, which has both of 3 dimensional position (x, y, z) and 3 axis rotation (φ, θ, ψ) [1]. In the whole

assignment, the vehicle driving on the ground is assumed to be flat and level. In this case, variables z, φ, and θ can be regarded as constant variables. Consequently, all of z, φ, and θ equal to 0 in the assignment and the platform state at time step k can be written as:

$$\mathbf{x}_k = \begin{bmatrix} x_k & y_k & \psi_k \end{bmatrix}^\top$$

where ψ is heading.

The odometry uses the last vehicle state and the current odometry to predict the current vehicle state. The odometry system returns the wheel speed and the rate of change of heading, which can be represented as:

$$u_k = [V_k, \dot{\psi_k}]$$

Knowing that the time interval is ΔT_k, the predicted current position and orientation of the robot are [1]:

$$\begin{aligned} x_k^- &= x_{k-1}^+ + V_k \Delta T_k \cos\left(\psi_{k-1} + 0.5\Delta T_k \dot{\psi_k}\right) \\ y_k^- &= y_{k-1}^+ + V_k \Delta T_k \sin\left(\psi_{k-1} + 0.5\Delta T_k \dot{\psi_k}\right) \\ \psi_k^- &= \psi_{k-1}^+ + \Delta T_k \dot{\psi_k} \end{aligned}$$

Assuming that the environment has Gaussian noise, the Kalman filter can be used in this stage. Also, the equations can be represented as [2]:

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1}$$

where w_k is process noise.

Deriving the formulation, get:

$$\hat{x}_k^- = A\hat{x}_{k-1}^+ + Bu_{k-1}$$

Given a linear representation:

$$x_k \approx \tilde{x}_k + A\left(x_{k-1} - \hat{x}_{k-1}\right) + Ww_{k-1}$$

Therefore, the process model A is the derivative of the equation verse x [3]:

$$A_{[i,j]} = \frac{\partial f_{[i]}}{\partial x_{[j]}}\left(\hat{x}_{k-1}, u_{k-1}, 0\right)$$

Also, the (1,3) and (2,3) values can be approximated to the derivatives of x in terms of ψ.

According to Kalman Filter [4],

$$P_k = E\left[\left(x_k - \hat{x}_k\right)\left(x_k - \hat{x}_k\right)^T\right]$$

Get the error covariance matrix [2],

$$P_k^- = AP_{k-1}A^T + Q$$

where Q is the system noise covariance and the noises in P are independent. In this case, values only exist in the diagonal of P.

Since the control input enters into a nonlinear way, Q is created to linearise again to figure out how the noise enters. Therefore, Q is related to the changes from input, the 3-by-2 matrix Q can be represented as:

$$\mathrm{Q} = \begin{bmatrix} \Delta T_k \cos\left(\psi_{k-1} + 0.5\Delta T_k \dot{\psi}_k\right) & -0.5V_k \Delta T_k \sin\left(\psi_{k-1} + 0.5\Delta T_k \dot{\psi}_k\right) \\ \Delta T_k \sin\left(\psi_{k-1} + 0.5\Delta T_k \dot{\psi}_k\right) & 0.5V_k \Delta T_k \cos\left(\psi_{k-1} + 0.5\Delta T_k \dot{\psi}_k\right) \\ 0 & \Delta T_k \end{bmatrix}$$

Using the equation [5]:

$$Q = QR_U Q^T$$

In this case, the error covariance matrix P can be obtained. Returning x_k^-, F_d, and Q, the simulation of Kalman odometry can be obtained.

As for GTSAM system, since GTSAM smooths the whole poses rather than filtering the last one, it has different representing method. The relative translation and rotation are used as the relative pose. However, it has the same equations of Q. Returning the relative pose and Q, the simulation of GTSAM odometry can be obtained.

Result and Discussion

Fig.1 shows the differences of Kalman filter over time. Comparing the two figures, the vehicle covariance increases over time. It is because the vehicle state do not have the process of update, errors will be gradually increase over time.
Fig.2 shows the differences of GTSAM system over time. Comparing the two figures, the vehicle covariance increases over time as well. Identically, it is because the errors do not cancel out over time.

Comparing the convariance of two algorithms versus time step, the figure can be seen in fig.3. It seems that the convariance in Kalman filter and GTSAM system are similar. Running the code several times, Kalman filter and GTSAM system can always obtain similar results.

Fig.4 shows the curves of the time required to run the optimiser. The blue curve is the Kalman optimization time with step increase and the red curve is the GTSAM optimization time with step increase respectively. As what can be seen in the graph, the

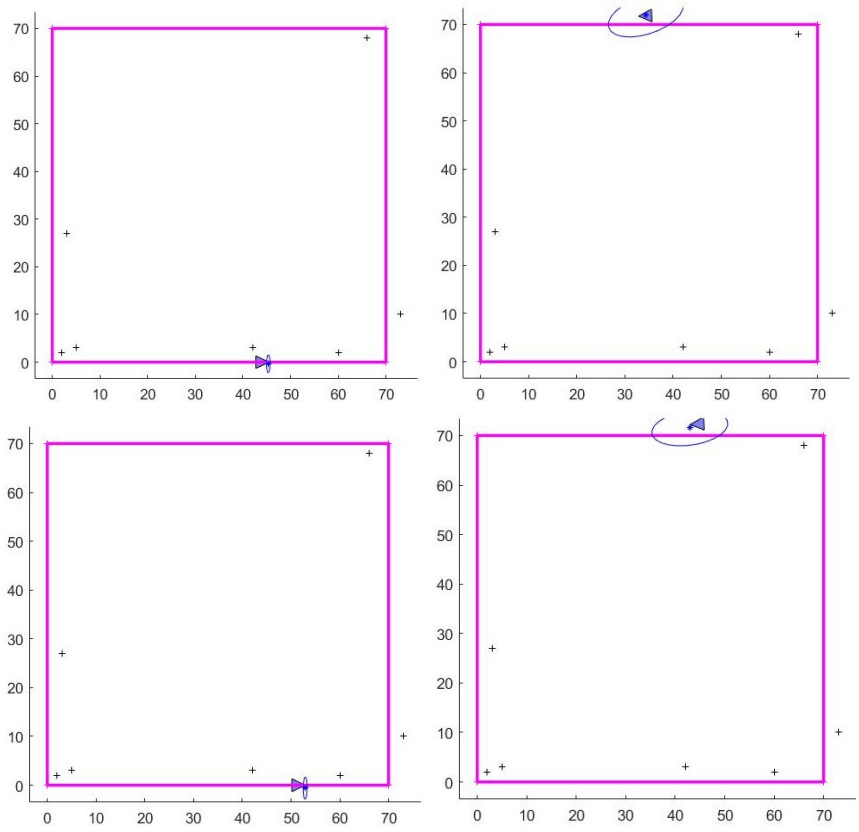

Figure 1: The differences of Kalman filter over time (top left and right)

Figure 2: The differences of GTSAM system over time (bottom eft and right)

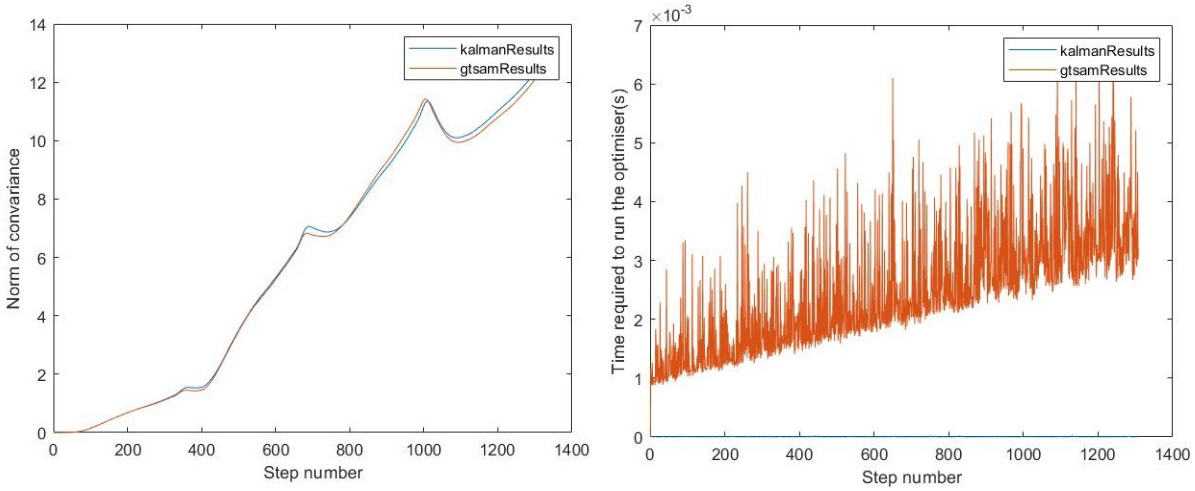

Figure 3: The convariance of two algorithms versus time step

Figure 4: Optimization time verse time step

Kalman filter keeps a low optimization time since Kalman filter only filters the data from the last state and it has similar computation cost in each step. Consequently, it has the plot which is similar to a constant line. By contrast, the GTSAM system smooths the whole states, so the computation cost gradually increases to calculate more states. Therefore, the result seems to be reasonable.

Task 1.2: GPS

In this part, the GPS for both the Kalman filter-based and iSAM based localization systems were implemented. Functions were written in *answers.kalman.GPSLocalizationSystem* and *answers.gtsam.GPSLocalizationSystem* classes respectively.

Methods

GPS is used to update both of Kalman filter and GTSAM system. Since GPS only measures the vehicle position, the estimated value of heading still uses the heading which was predicted before.

In Kalman filter, knowing that the vehicle state has three variables ($x_k \quad y_k \quad \psi_k$), the measurement matrix can be written as [5]:

$$\mathbf{H}_k = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The Kalman gain matrix (weight) can be computed using [4]:

$$\mathbf{W}_k = \mathbf{P}_k^- \mathbf{H}_k^{\mathrm{T}} \left(\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k \right)^{-1}$$

where R_k is the measurement noise covariance matrix.

Knowing the value of W_k, the estimated state and covariance can be updated, which are [4]:

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{W}_k(\widetilde{x}_k - \hat{x}_k^-) \text{ and } \mathbf{P}_k^+ = \left(\mathbf{I} - \mathbf{W}_k \mathbf{H}_k\right) \mathbf{P}_k^-$$

In GTSAM system, the definition way of measurement noise covariance matrix is the same. Sending the value of current measured GPS data and measured noise to GTSAM system using *PriorFactorPose2*, the simulation of GTSAM GPS system can be obtained.

Result and Discussion

Comparing the convariance of two algorithms versus time, the figure can be seen in fig.5. As what can be seen from the graph, the curves of Kalman filter and GTSAM system are similar. Moreover, the vehicle slows down in the corner. When the vehicle has a low speed, the covariance will decrease. When the vehicle has a high speed, the convariance

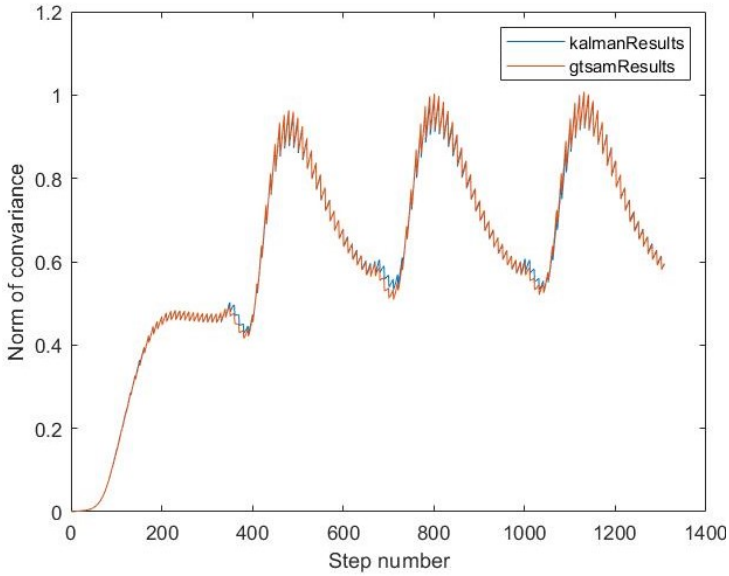

Figure 5: Optimization time verse time step

will increase. It might cause by the more similar vehicle states between measured values and predicted values. Each small decrease of the curve is because of update. The plot of GPS optimization time is similar to the plot of odometry optimization time since they have the same principle so the figure of optimization time was not shown in this task.

Part 2: SLAM System

Task 2.1: SLAM System Development

In this part, the development of a SLAM system was asked to be completed. To achieve this requirement, The method *processLaserObservation* in *LaserSensor2DSLAMSystem.m* was implemented.

Methods

In this part, the method implements both the initialisation of new landmarks and the update of existing landmarks with new measurements. Knowing that the range-bearing sensor returns the azimuth, elevation and range to a 3D landmark, it needs to be changed to 2D coordinate $\mathbf{Z} = \begin{bmatrix} r & \beta \end{bmatrix}^\top$ in the assignment, where r represents range and β represents bearing [1]. Then, the connected landmarks judged. If the landmarks have been register , get the landmark key. If not, the new landmark key needs to be created and inserted into the values map [2]. Since the value of observation is the difference between vehicle and landmarks. The initial value of landmarks can be written as:

$$\mathbf{m} = \begin{bmatrix} x_k + r\cos(\beta + \psi_k) & y_k + r\sin(\beta + \psi_k) \end{bmatrix}^\top$$

After constructing the laser observation model, adding bearing measurement factors to the graph, the simulation of SLAM system development can be obtained.

Result and Discussion

Fig.6 shows the changes of the 4^{th} landmark covariances over time. As what can be seen

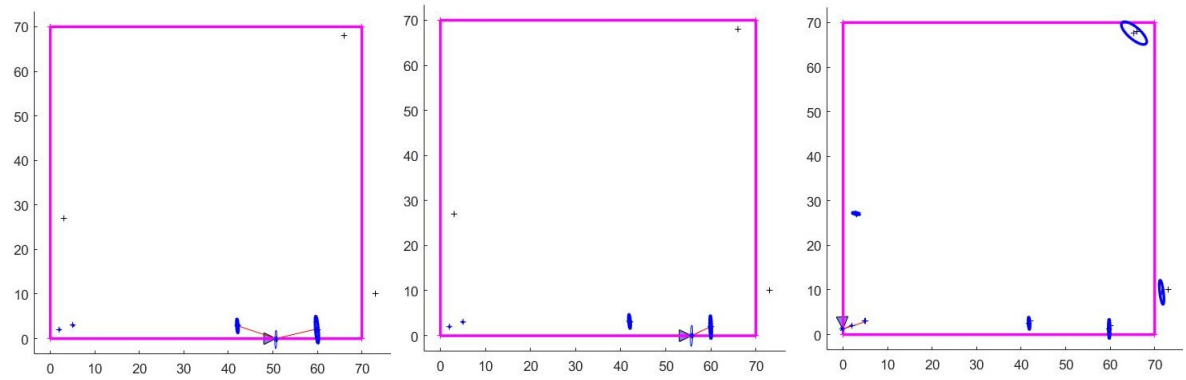

Figure 6: The changes of the 4^{th} landmark covariances over time

in the graph, with the increase of the number of landmarks, the convariance of the latter landmarks are larger than the previous one. Comparing the first graph with the second graph, the covariance initially declines rapidly and then converges to a value. It is because the first few measurements reduce any observation noise in the landmark. In this case, the position of predicted landmarks just change a bit. Furthermore, when the landmarks cannot be detected by the vehicle, the convariance will not be updated. Comparing the second graph with the third graph, when the vehicle observes the registered landmarks again, loop closing will reduce the landmark platform uncertainty and largely decrease the convariance of the vehicle state. In this stage, all the previous state and landmarks will be updated as well. Consequently, landmark covariance will decline further. In this case, the position of predicted landmarks may change a lot since the landmark platform uncertainty has changed.

Furthermore, the vehicle convariance increases over time before the vehicle observes the registered landmarks since GPS is not used in this stage. However, when the vehicle observes the registered landmarks, the vehicle convariance decreases. It is caused by the position corrected from registered landmark and then it helps to decrease the covariance of the previous landmarks.

Fig.7 illustrates the curves of the time required to run the optimiser. When the step number is less than 1200, the optimization time change is the same as the GTSAM optimization time in Task1.1. However, when the step number is larger than 1200, the optimization time has a large increment. It is because the closed loop updates the whole dataset rather than updates the local dataset. Consequently, it takes more time than before.

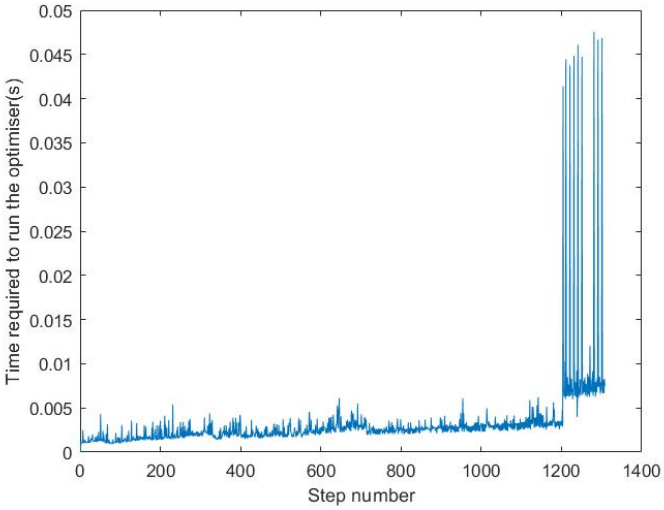

Figure 7: Optimization time verse time step in SLAM System Development

Task 2.2: Comparison with Odometry and GPS

In this part, the Matlab script *task22SLAMComparison.m* was used directly to compare different SLAM systems under 5 different conditions. The first one is a SLAM system with the effect of disabling odometry and without GPS measurements. The second one is a SLAM system with the effect of enabling odometry and without GPS measurements. These 2 situations are used to compare the effects of enabling / disabling odometry. In the all next 3 conditions, both the odometry and GPS measurements are enabled. However, the period of GPS measurements under these 3 conditions is different and set to 1 second, 2 seconds and 5 seconds. In this case, the effect of adding GPS measurements is supposed to be found out.

Methods

In this part, the code only uses the algorithms which were written in previous parts. In the Matlab script *task22SLAMComparison.m*, there are 3 options can be set to "true" or "false", Odometry, GPS and Laser. To increase the speed of the running of the SLAM system, a parameter in the script, *mainloop.m* was adjusted from 3 to 500.

Result and Discussion

Fig.8 demonstrates the differences of enabling and disabling odometry. Comparing the enabling and disabling odometry, the simulation without odometry has a large covariance at the start of simulation, which is caused by the measurement error from laser observation. By contrast, odometry makes few measurement error in the beginning. In this case, the covariance at the start of simulation with odometry is much smaller. Both of the algorithm's covariances increase when the landmarks are registered behind, which is because of the increment of the vehicle covariance. However, having odometry to ad-

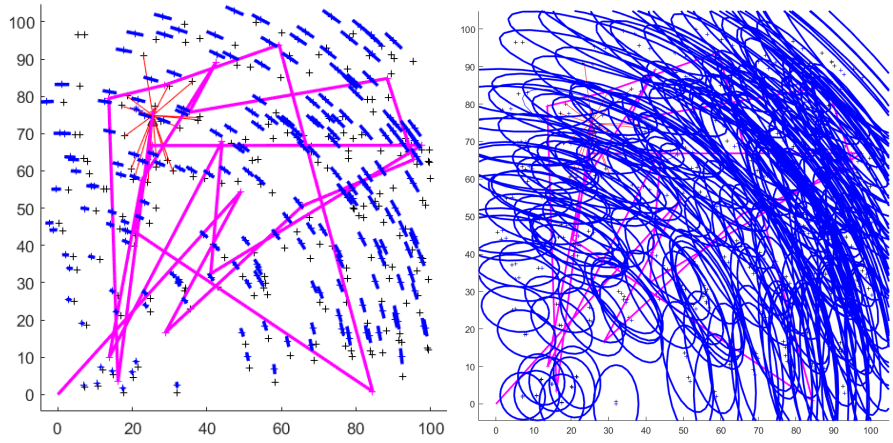

Figure 8: The simulation of enabling (left) and disabling (right) odometry without GPS

just the observations obtained from laser sensor, the simulation with odometry has much lower increment to convariance.

Fig 9. shows the effect of adding GPS measurements by different time interval with odometry. Comparing the graphs to the left graph of fig.8, the covariances decrease by

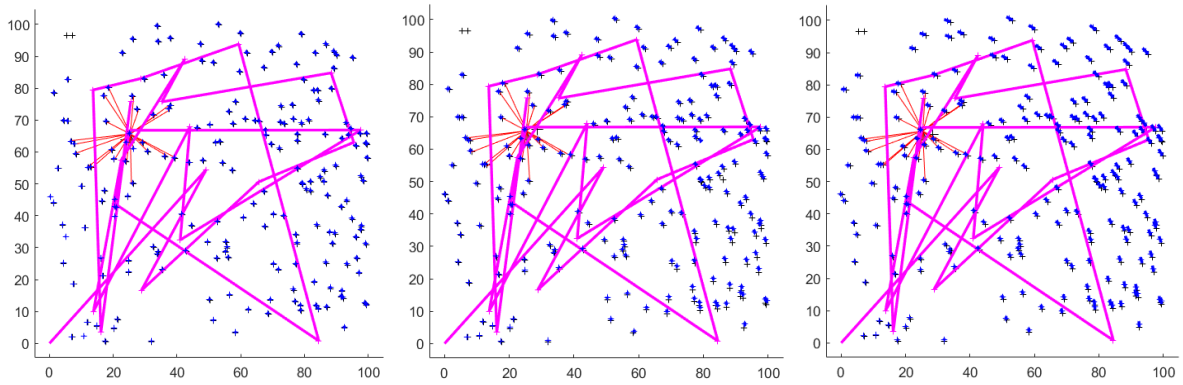

Figure 9: The simulation of adding GPS measurements by different time interval (1:left, 2:middle, 5:right) with odometry

adding GPS measurement. Knowing that GPS is a kind of absolute position measurement, so the convariance of GPS keeps to a fixed value. Using GPS to correct laser observations and odometry, it can help to smooth the whole vehicle and landmark states. Also, comparing the three graphs in fig.9, it seems that if GPS has smaller time interval, both of uncertainty and covariance have lower values. The reason for it is the more GPS states are given, the more accurate predicted vehicle state is. In this case, the landmark states will be more accurate if increasing the frequency of GPS measurements.

Moreover, the graphical model approach has one limitation is, the size of the graph will grow over time and get harder and harder to solve. One possible way to reduce the computational cost might be to subsampling vehicle states uniformly. To preserve features, if change of heading exceed an established threshold, the vehicle state will also

be collected. In this case, the vertices from the map will be pruned so the calculation states are reduced. Consequently, the computational cost can be decreased.

Conclusions

In this coursework, there are two parts. In the first part, the implementation of the odometry for both the Kalman filter-based and iSAM based localization systems were completed, with comparison. After that, the GPS measurements were added in the Kalman filter and GTSAM implementations, with comparison. In the second part, firstly, a SLAM system was completed by implementing the initialisation of new landmarks and the update of existing landmarks with new measurements. Secondly, the effects of odometry and GPS measurements on the developed SLAM system are researched by experiments. Furthermore, the period of the GPS measurements was set to 1 second, 2 seconds and 5 seconds with comparisons to research the effect of GPS. Finally, the results were plotted and discussed in this report.

References

[1] S. Ju1ier. (2019) Sparse feature-based visual slam. [Online]. Available: https://moodle-1819.ucl.ac.uk/pluginfile.php/368928/mod_resource/content/4/comp0130-coursework-02-spec.pdf

[2] S. Julier. (2018) Compgx04: Robotic vision and navigation probabilistic foundations of visual odometry and slam. [Online]. Available: https://moodle-1819.ucl.ac.uk/pluginfile.php/368982/mod_resource/content/2/GTSAM.pdf

[3] G. Welch and G. Bishop, "An introduction to the kalman filter," Chapel Hill, NC, USA, Tech. Rep., 1995.

[4] P. Groves. (2019) Comp0130: Robot vision and navigation workshop 2: Aircraft navigation using gnss and kalman filtering. [Online]. Available: https://moodle-1819.ucl.ac.uk/pluginfile.php/368975/mod_resource/content/11/COMP0130Workshop2instructions.pdf

[5] S. Ju1ier. (2019) Comp0130 worksheet 04: Introduction to graph-based optimisation and slam. [Online]. Available: https://moodle-1819.ucl.ac.uk/pluginfile.php/72373/course/section/170875/Workshop_04.pdf?time=1550749462956