2013 CACS International Automatic Control Conference (CACS)
December 2-4, Sun Moon Lake, Taiwan

Indoor Navigation with Smartphone-based Visual SLAM and Bluetooth-connected Wheel-Robot

Wei-Wen Kao
Department of Mechanical Engineering
National Taiwan University of Science and Technology
Taipei, Taiwan
wwkao@mail.ntust.edu.tw

Bui Quang Huy
Department of Mechanical Engineering
National Taiwan University of Science and Technology
Taipei, Taiwan
huy.buiquang86@gmail.com

***Abstract*— A smartphone-based positioning system suitable for indoor robot application is developed in this research by integrating image and other sensors on a smartphone with wheel odometer feedback from a Bluetooth-connected robot vehicle. WiFi signal, inertial sensors, and CMOS sensor in the smartphone were considered to provide measurements for positioning calculations. The real-time performances of SIFT, SURF, and ORB image feature detection and tracking algorithms are compared in the smartphone and ORB was chosen for implementation. Experimental result of Visual SLAM positioning is presented together with state convergence analysis.**

Keywords—indoor positioning; smartphone positioning; Visual SLAM

I. INTRODUCTION

In robot navigation systems it is often to integrate sensor measurements to determine the robot and landmark locations using SLAM (Simultaneous Localization and Mapping) technique. Conventional implementations often require expensive sensors and hardware with powerful computing capacity. Recently, smartphones have become popular with multiple built-in sensors and fast processors. A smartphone equipped with accelerometer, gyroscope, camera, GPS (Global Positioning System) and WiFi transceiver could fit the requirement for the robot navigation computing platform.

Using smartphones in navigation problem has been considered in recent researches. In [1] a fast and robust method for scale initialization that exploits basic geometric properties of the learned local map was proposed. As the work is targeting mobile phones, monocular SLAM was employed to jointly estimate a local map and the device's trajectory, but the results strongly rely on good initialization and takes time to converge. An indoor positioning method that allowed for a fine-grained detection of the position and orientation of a user was proposed in [2]. The research combined an image recognition system with a distance estimation algorithm to gain a high-quality positioning service independent from any infrastructure using the camera of a mobile device. The usage of a WLAN (Wireless Local Area Network) positioning system was also proposed to reduce the size of the candidate set of image recognition and hence speed up the system.

For visual positioning, good feature detection and tracking algorithm is required and SIFT (Scale-Invariant Feature Transform) feature detector and descriptor [3] has been proven useful in a number of applications. However, it requires heavy computational capability which is a difficulty for real-time implementation in low cost smartphones. A comparison of SIFT feature detector implementations for Android mobile devices was presented in [4]. Other feature detection algorithms with lower computation cost such as SURF (Speeded Up Robust Features) [5] are also developed. A computationally-efficient feature detector called ORB (Oriented Binary Robust Independent Elementary) that has similar matching performance, less affected by image noise, and is capable of being used for real-time performance was proposed in [6]. On a 1GHz ARM based smartphone with 512MB RAM the processing time for a 640×480 pixel (VGA) frames was around 150ms.

II. SMARTPHONE SENSORS

Modern Smartphones include many sensors suitable for positioning applications, such as Global Positioning System (GPS) receivers, WiFi transceivers, Bluetooth, RFID, NFC, inertial sensors, and CMOS image sensors. GPS provide phone location capability globally but in general will only work well outdoors. As this research focus general smartphone location applications for both indoors and outdoors, the phone sensor consideration is limited to WiFi, inertial sensors, and camera. The positioning capabilities and limitations of different sensors are studied in more detail in this section. The smartphone models used in this study are LG Optimus Black P970 (a low end dual-core smartphone by today's standard) and HTC OneX+ (a high end quad-core smartphone).

A. WiFi

Since Android smartphones provide the Received Signal Strength (RSS) from nearby WiFi access points (Aps), an experiment was conducted to analyze the performance of using WiFi signal strength for assisting indoor navigation and positioning. A grid consisting of 16 nodes was assigned to a room. Distances between nodes and the origin are measured beforehand. Three APs were installed at different locations in the room. The coordinate of grid nodes and APs are described in figure 1 (unit in mm).

Weighted K-Nearest Neighbor (W-KNN) method was used for positioning as below process:

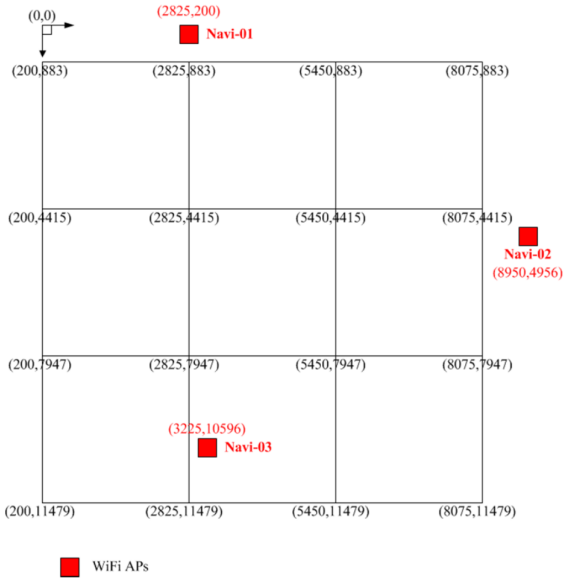


Fig. 1. Coordinate of WiFi grid nodes and WiFi APs

Step 1: Build up RSS map for nodes in the grid by standing at each node and gather about 100 signal strength sample data from 3 WiFi APs. The average RSS value for each node will be used.

Step 2: Pick a randomly position (e.g. point [1950,5298]) within the grid and calculate the weighted-KNN position from around 1000 samples.

Step 3: Repeat Step 2 again to get another result for that position.

The experimental result is shown in figure 2 below. It can be seen that positioning using WiFi signal strength creates errors in the range of several meters. For many indoor robot navigation applications this accuracy level is inadequate. As a result we do not use WiFi RSSI signal for our positioning system.

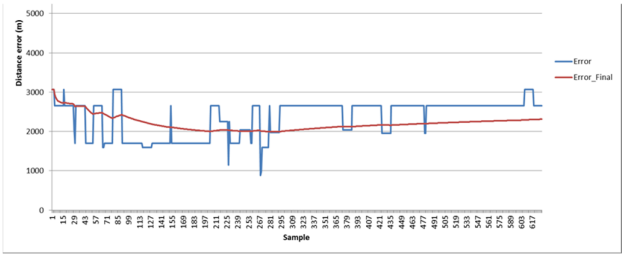


Fig. 2. Position error of one sample point

B. *Inertial Sensors*

An experiment was conducted by collecting the accelerometer and gyro data and the robot wheel encoder data at the same time when the phone is placed on a wheel robot and moving. From the data collected, the positions of the robot were calculated in two ways: one using accelerometer and one using encoder data. The results are compared in figure 3. It is clear that the low cost accelerometers (G-sensor) in smartphone produce large position drift and may be inadequate for many applications. In common robot applications the motions are generally slow with small accelerations and angular speed. Low cost inertial sensors with large measurement noises may not be able to measure the motions due to poor signal to noise ratio in the measurements.

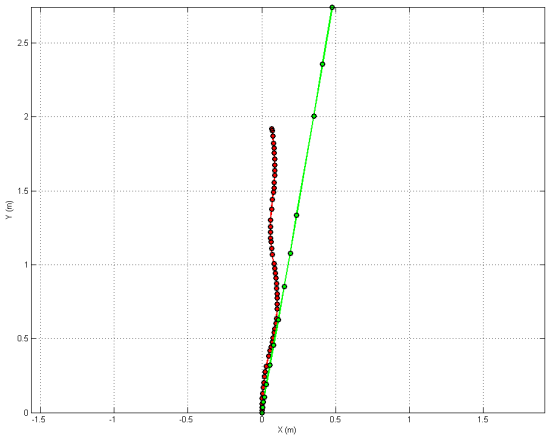


Fig. 3. Positions result from accelerometer (green) and encoder (red)

C. *Smartphone's camera*

The image sensor (camera) in smartphone differs in three ways from the cameras commonly used for navigation problems (especially SLAM problems): it operates at a low (<15Hz) frame-rate; it has a rolling shutter; and most importantly, it has a narrow field-of-view. Each of these effects individually would be problematic affecting the navigation and positioning algorithms result.

An experiment was conducted to test the processing speed of SIFT, SURF, and ORB feature tracking with OpenCV. The samples images are 320x240 pixels (QVGA) frames captured by the smartphone. The testing program was coded in C++ with OpenCV library [7], first running on a laptop with Core i5-480 processor (2.67GHz). For every two continuous frames, feature keypoints were detected and their descriptors were extracted and compared to each other to find the matches. The testing program was also developed in Android and run on 2 smartphones: LG P970 (1GHz, 512MB RAM) and HTC OneX+ (Quadcore 1.7GHz, 1GB RAM). Figure 4 shows the ORB, SIFT, and SURF feature tracking algorithms speed test results on a desktop computer. It can be seen that ORB algorithm is significantly faster than others.

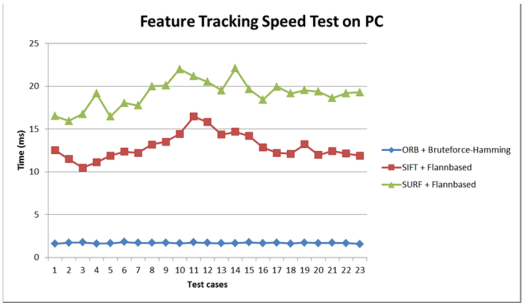


Fig. 4. Feature tracking speed test on PC result

Figure 5 presents the testing result of ORB in 2 different smartphones. The processing time for ORB is about 200 ms for high end and <1 sec for low end smartphones.

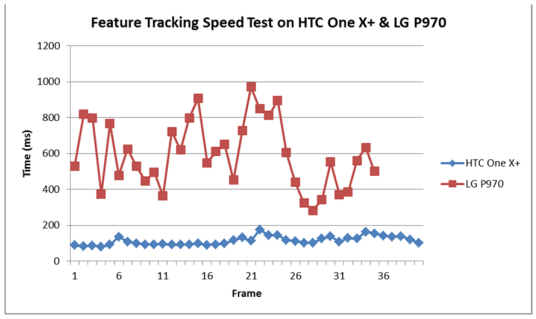


Fig. 5. ORB speed test on HTC One X+ & LG P970

III. VISUAL SLAM ON SMARTPHONE

Visual SLAM is the SLAM method which uses primarily visual (camera) sensors. Cameras provide unique advantages over other sensors used for SLAM. Cameras are cheap, low power consuming, compact and increasingly popular in mobile devices. Andrew Davison is one of the famous researchers working on Visual SLAM. His MonoSLAM [8] is an outstanding starter for SLAM using monocular camera. In his SLAM model:

- Only single camera was used as the sensor for localization and mapping.
- The experiment requires 4 known points for better distance estimation.

In the Visual SLAM implementation used in this paper:

- SLAM formulation with unknown environment and a robot kinematics model with motion sensor input.
- A single camera on a smartphone was used as a bearing sensor.
- A wheel-based robot (with encoders) carries the smartphone (with camera) on it.
- Encoder data was combined with feature tracking for localization and mapping.
- Inverse depth re-parameterization[9] was implemented to solve the depth initialization problem.

A. System description

The overall architect for the smartphone-based Visual SLAM wheel robot system is shown in figure 6. Stingray robot connects with the smartphone via Bluetooth communications. The micro-controller on the robot controls the low-level robot motion and sends the encoder data to the phone. The phone performs high level positioning calculation and generates path and control commands to the robot. To provide truth references for positioning results, a laser ranger connected to a laptop PC is used to acquire the real path of the robot motion. In order to synchronize the sampling timestamp, the Android phone connects to this laptop via WiFi. Whenever the phone processes a camera frame, it also sends a request to the laptop to acquire the laser ranger data via TCP connection.

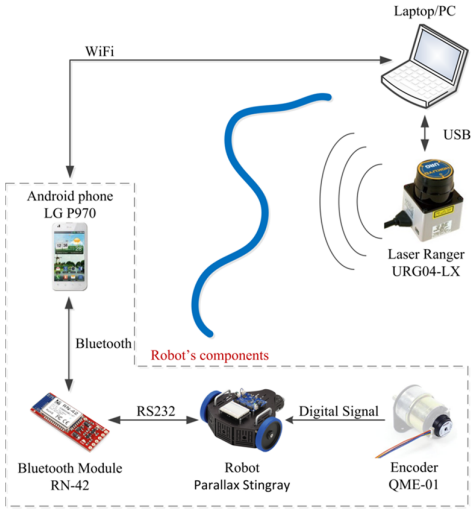


Fig. 6. System connections diagram & experimental setup

The experiment process is listed as follow:

Step 1: On the event of new camera frame captured, the phone logs its inertial sensor data (for future expansion) and make a request for encoder data from the robot.

Step 2: Apply the feature tracking process on the captured camera frame to extract the information about the feature (it is an old feature in previous frame or a new feature detected) and its coordinate on the image (in pixel).

Step 3: Use the feature information from Step 2 to calculate the measurement update for SLAM process.

Step 4: The encoder and gyroscope data acquired from Step 1 are used to calculate the state update for the SLAM process.

The result of Visual SLAM process would be estimations of the current robot poses and tracked feature positions. The phone will store these data and be ready for the next camera frame.

B. Visual SLAM formulation

1) State equation

The robot motion equation can be expresses as the following form in discrete time:

$$\mathrm{x}_v(k+1) = f_v\big(\mathrm{x}_v(k), u(k)\big) + w(k)$$

In this research the robot is assumed to move in 2-dimension with 2 translations and 1 rotation degrees of freedom. Hence,

$$\mathrm{x}_v(k+1) = \begin{bmatrix} x_c(k+1) \\ y_c(k+1) \\ \theta(k+1) \end{bmatrix} = \begin{bmatrix} x_c(k) \\ y_c(k) \\ \theta(k) \end{bmatrix} + \begin{bmatrix} V_k^c \times \cos\big(\theta(k)\big) \times \Delta t \\ V_k^c \times \sin\big(\theta(k)\big) \times \Delta t \\ \omega^c(k) \times \Delta t \end{bmatrix} + w(k)$$

where: $(x_C \ y_C \ 0)^T$ is the 3-D world coordinate position of the robot/camera;

θ is the robot heading angle;

V_k^C, ω^C are derived from the odometer and gyro sensor inputs;

$w(k)$ is the white Gaussian process noise;

Δt is the sampling time.

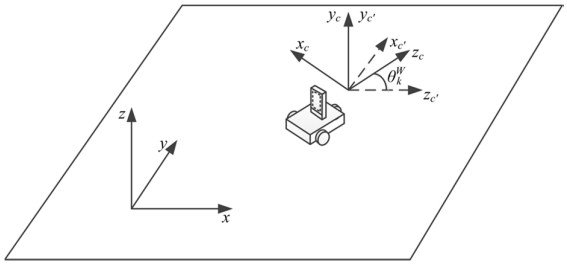


Fig. 7. Camera coordinate and world coordinate

Adding the observed feature positions states, the overall state equation would be:

$$\mathrm{x}(k+1) = f\big(\mathrm{x}(k), u(k)\big) + n(k)$$

$$\mathrm{x}(k+1) = \begin{bmatrix} \mathrm{x}_v(k+1) \\ \mathrm{y}_1(k+1) \\ \mathrm{y}_2(k+1) \\ \vdots \\ \mathrm{y}_n(k+1) \end{bmatrix}_{(5n+3)\times 1} = \begin{bmatrix} f_v\big(\mathrm{x}_v(k), u(k)\big) \\ \mathrm{y}_1(k) \\ \mathrm{y}_2(k) \\ \vdots \\ \mathrm{y}_n(k) \end{bmatrix}_{(5n+3)\times 1} + \begin{bmatrix} w(k) \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}_{(5n+3)\times 1}$$

where: y_i is the 5-D state vector of feature i $(i = 1..n)$ follow the inverse depth re-parametrization convention,

$$\mathrm{y}_i = [x_i \quad y_i \quad \theta_i \quad \phi_i \quad \rho_i]^T$$

and it encodes the ray from the first position of the camera optical center in world coordinate $(x_i \ \ y_i \ \ 0)^T$ from which the feature was observed.

The i-th feature position in world coordinate can be derived from the inverse depth parameters as following:

$$\mathrm{x}_i = \begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \\ 0 \end{pmatrix} + \frac{1}{\rho_i} m(\theta_i, \phi_i),$$

where: m is the unit directional vector from the first camera position to the feature position (in world coordinate),

$$m(\theta_i, \phi_i) = \begin{bmatrix} \cos\theta_i \cos\phi_i \\ \sin\theta_i \cos\phi_i \\ \sin\phi_i \end{bmatrix};$$

θ_i, ϕ_i is the azimuth and elevation (coded in world coordinate); and $\rho_i = 1/d_i$ is the inverse of d_i - the depth along the ray.

2) Measurement equation

For each observed features, its position in the image can be expressed as:

$$\mathrm{z}_i(k) = h\big(\mathrm{x}_v(k), \mathrm{y}_i(k)\big) + v(k)$$

$$h = \begin{bmatrix} u_0 - f_x \dfrac{h_x}{h_z} \\ v_0 - f_x \dfrac{h_y}{h_z} \end{bmatrix}$$

From Fig. 8, it can be verified that:

$$h^C = \begin{bmatrix} h_x \\ h_y \\ h_z \end{bmatrix} = R^{CW} \left(\left(\begin{bmatrix} x_i \\ y_i \\ 0 \end{bmatrix} - \begin{bmatrix} x_C \\ y_C \\ 0 \end{bmatrix} \right) + \frac{1}{\rho_i} m(\theta_i, \phi_i) \right).$$

where R^{CW} is the rotation matrix to convert from world coordinate to camera coordinate.

$$R^{CW} = \begin{bmatrix} -\sin\theta_k & \cos\theta_k & 0 \\ 0 & 0 & 1 \\ \cos\theta_k & \sin\theta_k & 0 \end{bmatrix}.$$

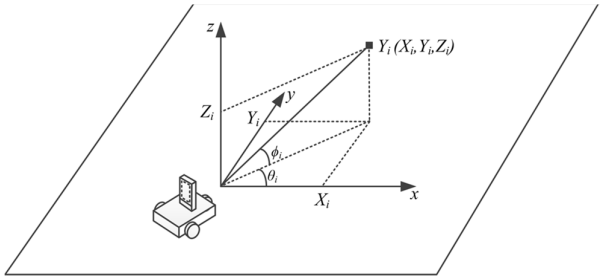


Fig. 8. Feature description in camera coordinate

With the state and measurement equations, Extend Kalman Filter (EKF) can be used to estimate the states.

C. Experiment Description

The Stingray robot with LG970 smartphone on it moves along a specific path in a room (lab room). A set of image frames was captured and the corresponding encoder values were also acquired and save to the SD card on the smartphone.

To automate the feature tracking process (for measurement equation in Visual SLAM), the result of feature detection by ORB detector would need to be processed through a filter to assure the feature availability and stability throughout the motion frames. This filter takes time to develop and is still not available in this time. Instead, the features in image frames were extracted manually and were saved into a file for off-line processing in Matlab. Features with different depth are extracted to provide better positioning accuracy.

The true features position in the world coordinate (which has the origin at the robot starting position) were measured manually for later comparison.

The Visual SLAM process needs a first guess for the initial state, which are:

- The robot starting position
- The robot starting heading angle
- First observed position
- Azimuth angle
- Elevation angle
- Inverse depth

The initial state errors affect the convergence of the SLAM process, which will be presented in the next section.

D. Experimental results

Due to the nonlinear nature of the SLAM problem, the state convergence is affect by the choice of initial states and different state will vary in convergence rate. In our experiment, the true initial states were first used in the Visual SLAM to verify its calculation and result. Figure 9 shows that the estimated robot/camera positions do not deviate from truth and does not diverge over long period of time (state errors are less than 3cm)

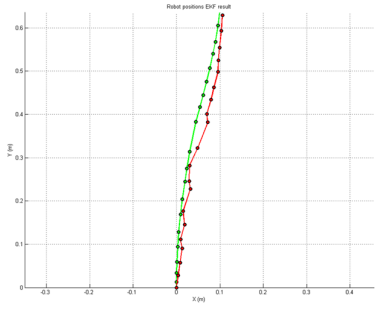
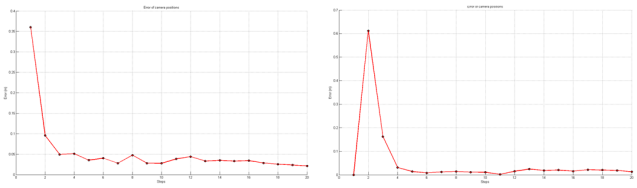


Fig. 9. Case 1 – robot positions from EKF result

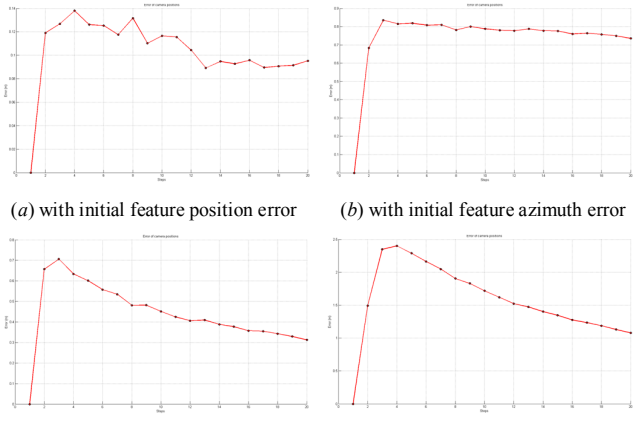
Camera position convergence was studied by introducing initial robot position errors. Figure 10 shows the camera positions convergence with initial position and heading angle errors (position error converge from 35-50cm to less than 3cm)



(*a*) with initial position error (*b*) with initial heading error

Fig. 10. Error convergence with initial robot position errors

Figure 11 shows the robot position convergence with initial feature position and angle errors. Since the convergence speed is slow, the errors are still high (>30cm) but they are decreasing.



(*a*) with initial feature position error (*b*) with initial feature azimuth error

(*c*) with initial feature elevation error (*d*) with multiple feature elevation errors

Fig. 11. Robot position error with initial feature position errors

Figure 12 shows the state convergence with all initial state errors.

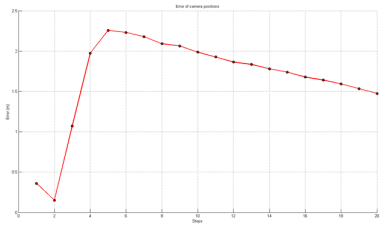


Fig. 12. Robot position error with all initial state errors

IV. Discussion

A. Practical limitations of the system

1) Images resolution and computational speed on smartphones

Due to the limited processing power of current day smartphones, the image resolution is limited in this research. With the QVGA resolution in the experiment, it took about 200ms for a basic image processing by the LG P970 smartphone. If VGA resolution was chosen, the speed decreases to 500ms per frame. This could affect the efficiency of the image processing algorithms, making it harder to detect and track good features. Hopefully this problem will be solved with more powerful smartphones in the future (with quad-core or even 8-cores processors which are now being available to the market). The development of GPU processing in mobile CPU will also bring a better chance to increase the efficiency of image processing using smartphone in future.

2) Dependency on lighting condition

The lighting condition is an important factor which affects the image quality. This limitation applies for most of the computer vision algorithms. With the current platform and software framework, additional image processing algorithms which produce better result could be implemented in the future to deal with such problem.

3) Camera autofocus speed

The autofocus speed of the camera on smartphones is rather slow. Some blurred image frame will be captured during the movement of the robot. The solution for this problem is turning off the autofocus function on the camera or keeping the robot's moving speed low enough (especially during turning motions) to avoid the blurred frames.

B. Research contributions

1) Analyzing of smartphone sensors for navigation problem

In this research, we evaluated several smartphone sensors for their usefulness in navigation. The accelerometer did not help much for most of the navigation problems due to its poor signal to noise ratio. For the gyroscope and the compass, their data could help in some applications to provide both the relative and absolute angle direction. The Visual SLAM experiment in this research did not use either of them because of the better accuracy from the wheel-robot encoder data.

From the result of weighted-KNN Wi-Fi RSS positioning experiment, the position error was about 2-4 meters and was not suitable for an indoor navigation application in small rooms

but it may help in large indoor buildings or outdoor environment when GPS is unavailable.

With the development of operating system for smartphones (Android, iOS, Windows Mobile), access of the phone camera and other sensors becomes more easily. The OpenCV image processing library for the Android does provide an excellent aid for researchers to test image processing algorithms directly on the smartphones. The camera on the smartphones nowadays still has some drawbacks in the navigation applications with its small viewing angle. Hopefully in future, the smartphone manufacturers will improve their camera and make it more suitable for navigation problems.

2) A system framework for testing image related navigation algorithms

The system proposed in this research provides the following capabilities:

- Many robot control algorithms (such as motion control, path planning, obstacle avoidance, etc.) can be implemented on the Bluetooth-connected wheel-robot, whether on the robot micro-controller or on the smartphone with more powerful processing capability.
- For navigation problems, especially with the SLAM problems, the smartphone provides a platform with multiple positioning sensors including gyroscope, digital compass, WiFi transceiver, and image sensor.
- With the aid of OpenCV library, most of the image processing algorithms could be easily implemented in the system.

3) Choice of suitable smartphone feature tracking algorithms

Several feature detectors, descriptors and matchers such as SIFT, SURF, and ORB were compared when implementing on the smartphone with low processing capability. The ORB detector and descriptor provides promising capability for real-time computer vision applications on a smartphone.

4) State convergence analysis of Visual SLAM

Real-time experiments were conducted using the system and a set of images along with the encoder data was captured and input to the Visual SLAM algorithm coded in Matlab.

Many tests were done to evaluate the convergence properties of different state elements in the SLAM algorithms. Wrong guess of the robot starting position or heading angle states do not affect much on the EKF result. However the convergence speeds with initial features states error are much slower, especially with the azimuth and elevation angle. These elements need multiple iterations with large measurement variations to converge faster.

As the robot movement is mostly in 2 dimensions space, the changes of the feature position in the camera frame are small (especially the vertical changes). The turning motions of the robot and the camera movement in vertical direction could help to speed up the convergence when bigger changes are created than those in straight motion.

C. Future works

1) Good filter for feature detection and tracking

The feature detection and tracking algorithm was tested on the computer for the speed but the features in the experiment were extracted manually. In order to automate the process and implement the whole system on the smartphone, a good filter needs to be created to assure the feature availability and stability throughout the motion frames. For example, several continuous frames can be taken into consideration to set the weight of trust for every features detected. The higher the feature is weighted the more the feature can be trusted.

2) Complete implementation on the smartphone

Implementation of the OpenCV C++ code on the Android smartphone was already tested. For EKF calculations where the matrix calculations on the phone are required, Eigen library is a good choice as a replacement for Matlab in the experiment. The estimated processing time for the LGP970 to finish the image processing and SLAM calculation for a step interval is around 400ms which is still good for a slow robot motion. This number could be further decreased using a more powerful smartphone in the future.

With the increasing popularity of the smartphones and also the advancement of powerful processors integrated, the smartphones-based computing platforms have high potential in future indoor navigation applications.

ACKNOWLEDGMENT

The authors acknowledge the support from National Science Council (project NSC 101-2221-E-011 -055) which makes this research possible.

REFERENCES

[1] S. Hilsenbeck, A. Moller, R. Huitl, G. Schroth, M. Kranz, and E. Steinbach, "Scale-preserving long-term visual odometry for indoor navigation," in 2012 International Conference on Indoor Positioning and Indoor Navigation, IPIN 2012, November 13, 2012..

[2] M. Werner, M. Kessel, and C. Marouane, "Indoor positioning using smartphone camera," in 2011 International Conference on Indoor Positioning and Indoor Navigation, IPIN 2011, September 21, 2011.

[3] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," International Journal of Computer Vision, vol. 60, pp. 91-110, 2004.

[4] K. Matusiak and P. Skulimowski, "Comparison of key point detectors in SIFT implementation for mobile devices," in International Conference on Computer Vision and Graphics, ICCVG 2012, September 24, 2012.

[5] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," in 9th European Conference on Computer Vision, ECCV 2006.

[6] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in 2011 IEEE International Conference on Computer Vision, ICCV 2011, November 6, 2011..

[7] OpenCV Developers Team. (2013). Open Source Computer Vision Library. Available: http://opencv.org/

[8] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 29, pp. 1052-1067, 2007.

[9] J. Civera, A. J. Davison, and J. M. M. Montiel, "Inverse depth parametrization for monocular SLAM," IEEE Transactions on Robotics, vol. 24, pp. 932-945, 2008.