The Journal of GIS Association of Korea, Vol. 16, No. 4, pp. 455-465, December 2008

Invited Papers

ARVisualizer : A Markerless Augmented Reality Approach for Indoor Building Information Visualization System

Albert Heekwan Kim* · Hyeondal Cho**

ABSTRACT

Augmented reality (AR) has tremendous potential in visualizing geospatial information, especially on the actual physical scenes. However, to utilize augmented reality in mobile system, many researches have undergone with GPS or ubiquitous marker based approaches. Although there are several papers written with vision based markerless tracking, previous approaches provide fairly good results only in largely under "controlled environments." Localization and tracking of current position become more complex problem when it is used in indoor environments. Many proposed Radio Frequency (RF) based tracking and localization. However, it does cause deployment problems of large RF-based sensors and readers. In this paper, we present a noble markerless AR approach for indoor (possible outdoor, too) navigation system only using monoSLAM (Monocular Simultaneous Localization and Map building) algorithm to full-fill our grand effort to develop mobile seamless indoor/outdoor u-GIS system. The paper briefly explains the basic SLAM algorithm, then the implementation of our system.

Keywords : Augmented Reality (AR), Mobile, indoor localization and tracking, Hand-held, u-GIS, SLAM (Simultaneous Localization and Map building), USB camera, UMPC, GIS

요 약

증강현실은 지리정보의 가시화 특히 현장에서의 직접적인 가시화에 있어 매우 높은 잠재력이 있다. 하지만 현재까지의 대부분의 이동형 증강현실 시스템은 사용자의 정확한 위치를 파악하기 위해 GPS 또는 범용적으로 쓰이는 마커를 현장에 붙이는 등의 방식을 사용되었다. 물론 최근의 연구에서 마커없는 환경을 지향하고 있으나 대부분 연구실 또는 제어 환경으로 사용이 제한되어있다. 특히 실내의 경우 GPS를 사용할 수 없기 때문에 새로운 위치파악기술이 더욱 절실 하다. 최근 활발히 활용되고 있는 무선(RF)기반의 실내 위치확인 및 내비게이션 기

* Cheif Executive Officer (albertk@zenitum.com)
** Principal System Architect (nukbridge@zenitum.com)

술 역시 대량의 센서와 인식기를 설치한다는 점에서 그 실용성이 의문이다. 본 연구에서는 단일카메라기반의 SLAM 알고리듬을 이용하여 특수한 하드웨어 없이 카메라만으로 실내 위치 확인 및 내비게이션이 가능한 알고리듬을 제시하였으며, 동시에 확인된 위치에서 증강현실을 통한 정보의 가시화가 가능하도록 구현 하였다. 향후 본 연구가 목표하고 있는 실내외 seamless 연동형 u-GIS 시스템의 기본 기능으로 활용 될 것이다.

주요어 : 모바일, 증강현실, 복합현실, 실내 내비게이션, u-GIS, 동시 위치인식 및 지도작성 알고리듬

1. Background

Even with the advent of the civilian GPS technologies recently, the indoor location tracking has been a "grey-area" for the research community. Since there is no possible, at least currently, method to access GPS from indoor space, some researches have been focusing on different methods, mostly with Radio Frequency (RF) based and few with motion sensor based approaches.

Systems like Active-Bat [1] developed by AT&T and Cricket [2] use ultrasound time of flight measurements, others like Active-Badge [3] use small infrared tags that provide symbolic information, like a name of a room. Infrared has limited range hence it has not become very popular. PinPoint 3D [4] uses radio frequency lateration to provide an accuracy of three meters and requires a special developed infrastructure. SpotON [5] and UOID [6] projects developed a 3D and 2D location, respectively, system using RFID tags. Kang *et al.* also recently attempted to guide automobiles' 2D parking path based on RFID [7]. SmartFloor [8] uses a sensor grid installed on a floor and the accuracy depends of the sensor spacing distance.

These systems require a special infrastructure that needs to be deployed. This is an obvious disadvantage. Systems that do not require a special developed infrastructure take advantage of an existing and well defined telecommunication network, like cellular or WiFi networks. Cellular networks, like GSM and 3G/UMTS, can provide, with minimal cost, the user current cell identification. The accuracy of this solution depends mostly on the size and density of cells. In urban environments, there is a much better accuracy since more base stations are installed. Signal triangulation is also possible, using the different radio signals from the different base stations [9].

Knowing on which base station the client is connected can also be applied to wireless LANs. This kind of information can be used to locate a wireless client within the access point coverage area.

Implementing a location system using WLAN communication infrastructure has been for some time target of intensive research. Various appro-

aches have been proposed, mostly based on the received signal strength. One of the first systems to be introduced was RADAR, developed by Microsoft Research Group [10].

RADAR uses received radio strength to map the user current location and it uses an empirical model (K-Nearest Neighbor) as well as a simple signal propagation model. The accuracy of this system is about four meters for 75% of the time and it uses special developed access points. R. Battiti *et al.* describe a system capable of deriving the location using neural networks [11]. In this work it is described the training phase, always present in a RF fingerprinting based system, and the neural network architecture used [120].

Despite the accuracy (about 2.3 meters), the results are only compared to test data and information about software developed, radio strength reading method and system architecture is in existent. M. D. Rodriguez *et al.* [13] also describe a location system for hospital services based on a neural network. In this work the SNR is used to calculate the user position and all the processing is done in the wireless client. Their results show an error below 4 meters 90% of the time.

Bayesian models are also used to calculate a wireless device position, D. Madigan *et al.* [14] propose a system with an accuracy of four meters. A. Haeberlen *et al.* [15] describe a probabilistic approach and their location system provides symbolic information, like the office number inside a building.

As explained above, RF based indoor location tracking is favored from many research projects. However, there are some limitations on deploying RF based solutions. 1) The resolution of tracking is quite sparse. The tracking ranges are from 5m to 10m depends on the technology. 2) Since RF based sensors are physically installed in the sites, to achieve more dense resolution, more hardware sensors and reader (active readers) must be in place beforehand. 3) When they localize the tracking targets, it is only possible to track 2 dimensional location. It's quite difficult to determine the users (one with active readers or beacons) relative position to the sensors.

Because of various problems mentioned above, we now present a noble way to overcome the problems

2. Introduction

SLAM (Simultaneous Localization and Mapping) is a well defined and used approach in the robotic community for constructing a representation of the environment on the fly and estimating robot motion. SLAM is mainly accomplished by using modern methods of sequential Bayesian inference and normally uses sensors such as laser range-finders and sonar. MonoSLAM was created based on the probabilistic SLAM methodology using a single freely moving wide-angle camera as the only sensor and with a realtime constraint [16].

The MonoSLAM algorithm runs at 30 frames per second (fps), estimates camera pose and creates a sparse map of the environment natural landmarks. It is a very efficient algorithm with a

low level of jitter (1-2 cm) and drift-free, while being robust to handle extreme rotation, occlusion and closed loop. However, it is restricted to indoor environments, smooth camera movement and monochrome camera image.

To initialize the system, a known picture is necessary to be present in the initial frame at an approximated certain distance. The algorithm begins searching for features in the image utilizing the image interest operator of Shi and Tomasi [17] to locate the best candidate within a limited window of 80×60 pixels. This window is randomly positioned in any area that does not contain other features nor will be out of the camera view, based on the current camera and angular velocities.

When good features are selected, the algorithm estimates its depth and associates it with a level of uncertainty. As the feature continues to be tracked in the next frames, the depth estimation is enhanced and the feature is fully initialized and stored as an oriented planar texture of 11× 11 pixels. It utilizes the Davison and Murray's approach, which relies on visual landmarks, as they have more unique signatures than standard corner features [18]. The tracking of each feature occurs within an ellipse region, and shape and position in the image are defined based on its level of uncertainty and on the camera estimated movement, respectively.

The features are inserted in a probabilistic feature based map that is maintained during all the lifetime of the operation and is updated by the Extended Kalman Filter (EKF). The map grows as new features are added, or shrinks when a feature that fails to be detected many times is removed.

The MonoSLAM technique was used in an AR scenario where virtual furniture is added to an image stream captured by a handheld camera.

3. System Design

In there previous research, Davison *et al* has been using a wide-angle IEEE 1394 cameras connected to the computer [19]. The throughput of IEEE 1394 camera data transfer rate is quite steady. Also, since it has a wide-angle optics, the processing software has luxury of seeing wider scenes with more feature points overlapped.

Nevertheless, using IEEE 1394 on most of hand-held devices would be a problem. Most IEEE 1394 cameras are using a 6-circuit Fire-wire 400 (IEEE 1394) alpha connector which draws power from the host computer (Figure 2). Unfortunate, because of its power constraints, most off-the-shelf mobile computers (including most laptop computers) only support IEEE 1394a, 4-circuit alpha connector, developed by Sony and already widely in use. Since the 4-circuit connector does not support the power, additional

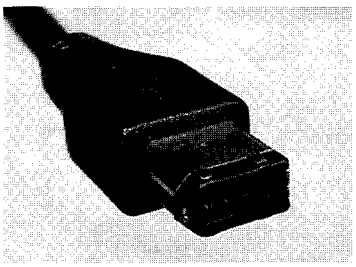

Figure 1: A 6-circuit FireWire 400 Alpha connector

Figure 2. The System: UMPC with USB Camera

power sources must be supplied to the IEEE 1394 cameras. [20]

This limitation forces us to use USB 2.0 cameras connected to a hand-held devices. (or embedded cameras) On the current implementation, Logitech Qucikcam IM (USB 2.0) was used for the image acquisition. Although USB 2.0 nominally runs at a higher signaling rate (480 Mbit/sec) than FireWire 400, typical USB PC- hosts rarely exceed sustained transfers of 280 Mbit/sec. This is likely due to USB's reliance on the host-processor to manage low-level USB protocol, whereas FireWire delegates the same tasks to the interface hardware.

To overcome the limitation, some part of algorithm had to be minimized and optimized for the hand-held device. For the hand-held device, we used SONY VGN-UX58LN UMPC (Ultra Mobile PC). The system runs on Intel Core2 Solo U2200 with 1.2Ghz clock speed. (Figure 2)

3.1 Camera Tracking

As in Davison's original work [21], our system maintains an estimate of the current camera pose/motion parameters and 3D scene points. Their respective means are x_v, y_i, and a covariance matrix, P, encodes Gaussian uncertainty as well as correlation between measurements and camera parameters.

The system is initialized with a set of four known points. Image patches are used as feature descriptors and measurements are made by performing normalized cross correlation between a template image patch and the current image in a search window around the predicted feature position. When no features are present in part of the current image, new features are added at positions that have a high response to the corner detector.

To improve feature measurements over large changes in viewing configuration, we have incorporated Molton's locally planar patch features [22], where each feature is assumed to be planar and it's template is pre-warped based on an estimated normal. Pre-warping patches significantly improves the range of poses over which a template can be matched, even when the normal estimate is a coarse estimate (pointing directly toward the camera when the feature is initialized).

Experimentally, we have found the cost of updating the normal estimate by image- alignment and Kalman-filtering to be too costly to justify for the slight further improvement in tracking. Therefore we don't update the initial normal estimates. To improve performance, instead of performing cross correlation at every position within the search window, we have limited the measurement search to detected fast corners [23]. In experiments, the use of these corners reduces the number of correlations by approximately 85%, at a cost of 12% fewer matches, and a mean measurement error increase of only 0.1 pixels.

3.2 Keyframe Selection and Storage

The system deals with a large amount of raw image data from the video stream, so memory management becomes an issue. Images are stored in three types of memory (texture, main, and disk memory). As images are acquired by the camera they are loaded into main memory. When they are requested for texture mapping they are loaded into texture memory and if they are determined to be novel they are written to disk. When storage exceeds a predefined quota for main or texture memory, images are unloaded based on their distance from the current virtual view. A frame is considered novel if it has a sufficiently different position or orientation from the closest saved frames. The score we use for novelty incorporates the distance between frames relative to the mean distance to the scene and the angular difference between the principle viewing rays of the new frame and nearby saved frames.

First we compute the Euclidean distance between the new camera center, ccnew, and each of the saved camera center, cci, weighted by the inverse mean depth to the scene points, X_j (where j $\in$ 2 [1, m]), relative to the new camera.

$$d_i = \frac{dist(cc_{new}, cc_i)}{\sum_j (depth_{cc_{new}}(X_j))/m)}$$

If the minimum of this distance evaluated for all saved cameras, $\min_i$ (d_i), is above a threshold, a, we consider the frame novel (in our experiments $a = 0.1$ resulting in images approximately 10 cm apart when the mean depth of the scene is $1m$). If the minimum distance is below then for each saved camera with $d_i < a$ we compute the angular distance from it's principle viewing ray to that of the new frame. If the maximum angular distance is above a second threshold, β, then we also consider the frame novel (in our experiments $\beta = \pi/4$ resulting in images approximately 45° apart).

Camera pose parameters, x_v, from the tracking system are stored along with each saved image. Since SLAM only updates an estimate of the current camera pose, over time old camera estimates provided by the tracking system may become invalid. As a result we intermittently improve estimates by resectioning saved cameras using all available 3D points and their 2D measurements.

4. ARVisualizer : Software Design

ARVisualizer applies MonoSLAM algorithm to compute camera's 3D pose and at the same time maintains a sparse map (represented by a small amount of features) on the back stage. After each step of updating the map status, it visualizes the map and camera's pose with OpenGL and some other utilities.

Figure 3 depicts the general workflow of ARVisualizer. The core of the diagram is the spare map, which generally contains ~13 features. The application works in the following way: On the startup, application loads configuration (i.e. default corner pattern model, camera parameters) and initialize image/video grabbers. Before online

Table 1. Major Modules and Descriptions

Main modules are listed in table 1.

Module	Description
Math Library [24]	Matrix, algebra operation, etc
Image Processing library. [25]	Feature matching, detector, etc
3D drawing Utility	Primitives for drawing axis and setting up 3D environment. Also some interfaces are defined in VWGLOW and SceneLib
Image/video acquisition	Interfaces to read images or live stream from capture device
MonoSLAM algorithm	including EKF-based SLAM algorithm and feature matching with active vision
OpenGL utilities	OpenGL extensions
GUI	Button control and display panels

running, initialize the map with fixed patterns at a known distance (e.g. black and white papers in a fixed distance). Some features are automatically detected and initialized with the default corner pattern and the distance in order to get a initial map of the environment.

When online, map is updated when new image data arrives. It follows an EKF-based approach – estimate, observe and update. That is, firstly, estimating the next position of camera based on current map and camera's motion model. Secondly, matching and tracking features in the new arriving image. And, finally, updating map by synthesizing the features locations and camera's position.

Update the GUI to display the movement of features and changes of camera's position. Map information is refreshed with the feature positions as well as their confidence area. Overlay AR model on the map at specified feature.

5. ARVisualizer : Code Structure

5.1 Initialization

To initialize the system, the following steps must be taken. First, *MonoSLAM* class in *monoslam.h* must be loaded for the configuration. *MonoSLAM::Auto Initialize Features in monoslam.h*, then automatically detects features. Finally *nonoverlappingregion.h, SceneImproc (SceneLib\scene_single.h)* Initializes maps by detecting and matching the flow of features

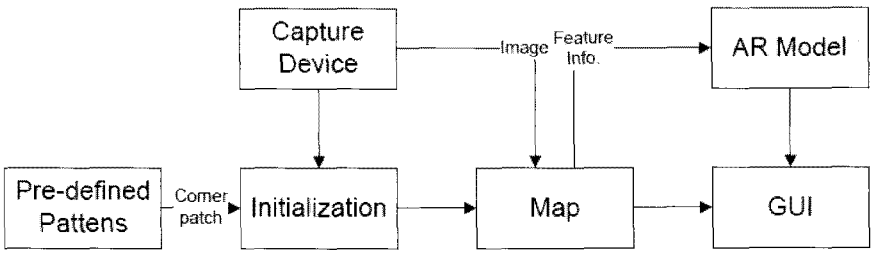

Figure 3. Workflow of the ARVisualizer Software

5.2 Image/video acquisition

As showed in Figure 4.1, The capture device works in a push mode – read data from devices at a fixed time interval controlled by a thread monitor. Capture devices are all derived from *SequeceBase*, which defines control interfaces such as "Next", "Continue", "Stop", and the different behaviors among devices are resort to the specific driver.

Take *USBCamGlow (SequencerUSBCam Glow, SequencerDXCam* in Figure 5, 6) for example, it overrides interfaces in *SequenceBase* by calling customized driver for capturing video stream with DirectShow

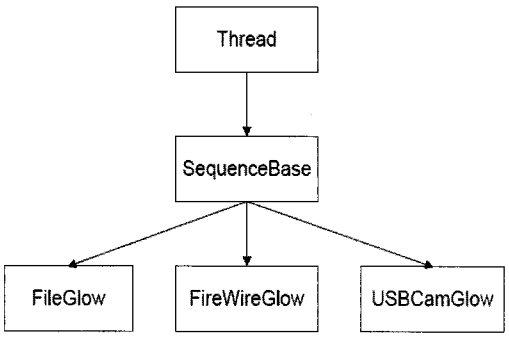

Figure 4. Diagram of Capture Device

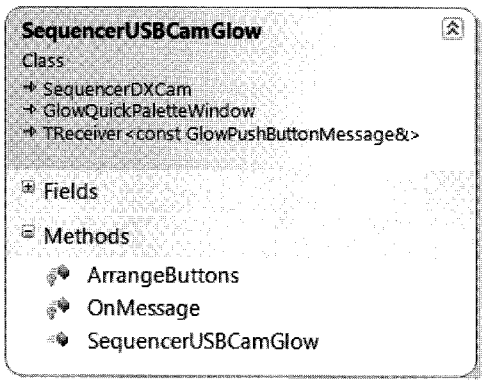

Figure 5. Inheritance tree of *SequencerUSBCam Glow*

5.3 SceneLib

SceneLib implements MonoSLAM algorithm, defining some important classes and structures, such as MonoSLAM, scene_single, Kalman, Feature, etc. ARVisualizer uses Glow to manage window operation and encapsulates all the 3D rendering operations in *ThreedToolGlCommon* and *Threed ToolGl Widget* which manage window system (display, user interaction, etc).

The two display windows in main GUI is inherited from *ThreedToolGlWidget*. Also, AR or other image rendering operations could be done in either *ThreedToolGlCommon* or *Threed-Tool GlWidget*.

5.4 3D Display (in VWGL and VWGLOW)

ARVisualizer uses Glow to manage window operation and encapsulates all the 3D rendering operations in *ThreedToolGlCommon* and *Threed-ToolGl Widget* which manage window system (display, user interaction, etc). The two display windows in main GUI is inherited from *Threed-ToolGlWidget*. Also, AR or other image rendering operations could be done in either *ThreedTool-GlCommon* or *ThreedToolGlWidget*. (Firgure 6)

5.5 Main GUI (in MonoSLAMGLow)

Two classes are implemented to deal with 3D display, user interaction and image capture. *Mono-SLAMControl* is a wrapper for capture device and manages all the buttons in the control panel.

The specified device used could be configured with certain MACRO (_IMGFILE_, _USBCAM_, etc) in the development environment.

In logic, *MonoSLAMControl* is created when

ARVisualizer is loaded, and it keeps a pointer to *MonoSLAMGlow* class. So that, whenever a button is pressed or triggered, *MonoSLAM-Control* will broadcast the button event to *Mono SLAMGlow* to process The specified device used could be configured with certain MACRO (_IMGFILE_, _USBCAM_, etc) in the development environment

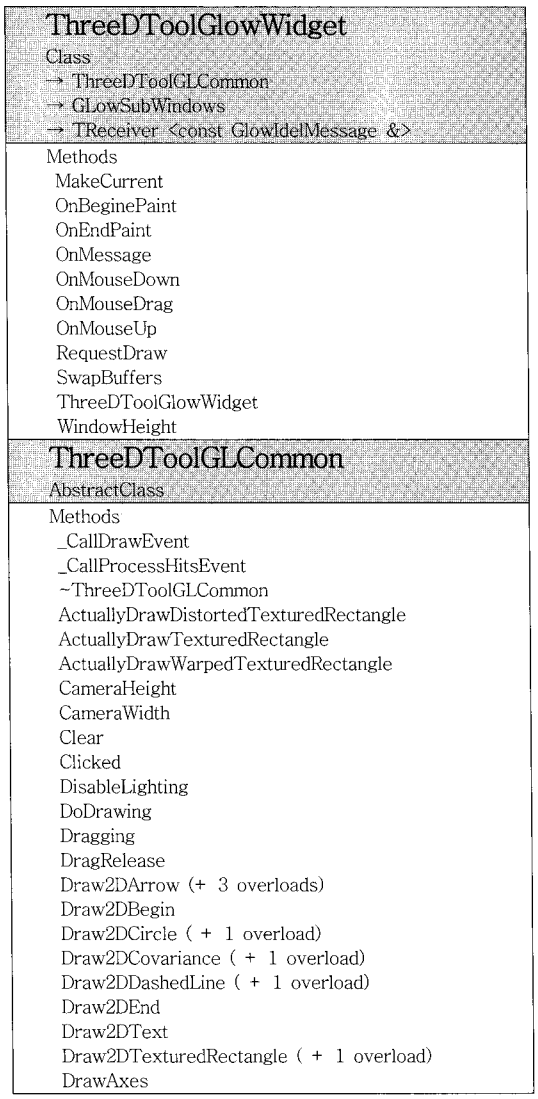

Figure 6. 3D Drawing Widget Classes

In logic, *MonoSLAMControl* is created when ARVisualizer is loaded, and it keeps a pointer to *MonoSLAMGlow* class. So that, whenever a button is pressed or triggered, *MonoSLAM-Control* will broadcast the button event to *MonoSLAMGlow* to process.

On the other hand, *MonoSLAMGlow* is the display window, consisting two panels - 3D map panel and VR (virtual reality) panel. 3D map panel displays the feature and camera's positions in 3D coordinates, while VR panel renders image and AR model.

Important rendering functions such as *Rectified-Internal3DDraw*, *RawInternal3D Draw*, are defined in SceneLib /MonoSLAM/threedDraw. cpp. AR models could be overlayed on specific features by modifying these interfaces.

6. Conclusion

In this paper, we have implemented a Mono-SLAM based Augmented Reality for the computationally limited hardware devices. To accomplish that, we develop a noble method to optimize the algorithm so that an off-the-shelf narrow angle USB camera can be connected on the UMPC. We have demonstrated that merging collada/KML based 3D objects on incoming video scenes, as well as text annotations freely on the 3D space.(Figure 7) Still, however, the implementation is on Intel Core2 Solo which has only a single core.

Finally, we have found that if we can utilize

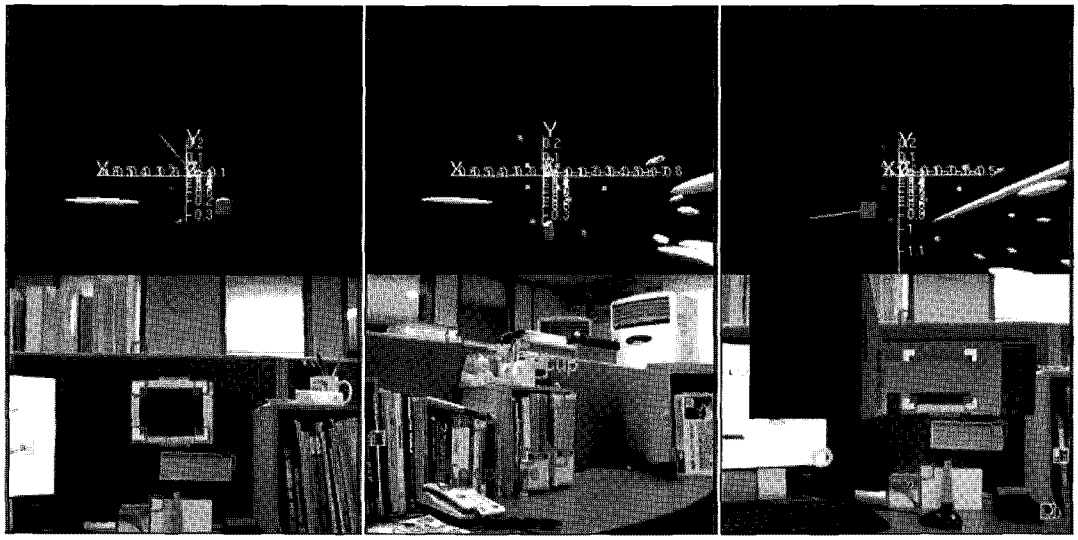

Figure 7. Implementation Results: (From left to right) (1) Initialization of 3D map building, (2) Free space text annotations, (3) 3D Collada Image overplayed on the 3D space.

multi-core architecture; separating tracking and map building on different threads, we can increase the performance of the system significantly by using dense map, instead of using a sparse map in this implementation. [26]

Acknowledgement

This research was supported by a grant (code 07KLSGC05) from Cutting-edge Urban Development- Korean Land Spatialization Research Project funded by Ministry of Construction & Transportation of Korean government.

References

A. Ward, A. Jones, and A. Hopper. A new location technique for the active office. IEEE Personal Communications,4(5):42–47, Oct. 1997.

N. B. Priyantha, et al, The Cricket Location-Support System, Proc. 6th Ann. Int'l Conf. Mobile Computing and Networking (MOBICOM 2000), ACM Press, New York, 2000, pp. 32-43.

R. Want et al., The Active Badge Location System, ACM Trans. Information Systems, Jan 1992, pp 91-102.

Jay Werb and Colin Lanzl. Designing a positioning system for finding things and people indoors. IEEE Spectrum, September 1998 pp 71–78.

J. Hightower, et al. SpotON: An indoor 3D location sensing technology based on RF signal strength. Technical Report UW CSE 00-02-02, Department of Computer Science and Engineering, University of Washington, Seattle, WA, Feb. 2000.

Y. Jang, et al. Management Plan of Urban Object IDentification through Status- Analysis of Existing Object Management Code. The Journal of GIS Association of Korea, Vol. 16, No. 1, pp. 51 -64, April 2008.

K. Kang; J. Kim A System of Guiding Path for

Parking Lots based on RFID to Consider Real-Time Constraints. The Journal of GIS Association of Korea, Vol. 16, No. 1, pp. 65-77, April 2008.

Orr, Robert, et al., The Smart Floor: a Mechanism for Natural User Identification and Tracking. Human Factors in Computing Systems, 2000, Georgia Institute of Technology.

Location Technologies for GSM, GPRS and UMTS Networks. SnapTrack. 2003.

Bahl, P. and Padmanabhan, V.N.. (2000). RADAR: An in-building RF-based user location and tracking system. In: EEE Infocom 2000, Tel Aviv, Israel, March 2000.

Roberto Battiti, et al. Location-aware computing: a neural network model for determining location in wireless lans. Technical Report DIT-5, Universit`a di Trento, Dipartimento di Informatica e Telecomunicazioni, 2002.

Kaemarungsi, Kamol, et al. Modeling of Indoor Positioning Systems Based on Location Fingerprinting. IEEE, 2004, School of Information Science. University of Pittsburgh.

Rodriguez, Marcela D., et al. Location-Aware Access to Hospital. IEEE Transaction on Information Technology in Biomedicine, Dec. 2004.

Madigan, David, et al. Location Estimation in Wireless Networks. A. Rutgers University. 2005.

Haeberlen, Andreas, et al.. Practical Robust Localization Over Large-Scale 802.11. MobiCom'04, 2004, Rice University. Philadelphia: ACM, 2004.

Davison, A. J.; Molton, N. D. MonoSLAM: Real -time single camera slam. IEEE Trans. Pattern Anal. Mach. Intell., IEEE Computer Society, 2007.

HI, J.; TOMASI, C. Good features to track. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94). Seattle: [s.n.], 1994.

DAVISON, A. J.; MURRAY, D. W. Simultaneous localization and mapbuilding using active vision. IEEE Trans. Pattern Anal. Mach. Intell., IEEE Computer Society, Washington, DC, USA, v. 24, n. 7, p. 865-80, 2002. ISSN 0162-8828.

DAVISON, A. J.; Reid, I. D.; Molton N. D.; Stasse O. MonoSLAM: Real-Time Single Camera SLAM. IEEE Trans. Pattern Anal. Mach.d.

IEEE 1394 Interface, Wikipedia: http://en.wikipedia.org/wiki/FireWire.

A. Davison. Real-time simultaneous localization and mapping with a single camera. In Proc. International Conference on Computer Vision, Nice, Oct. 2003.

N. D. Molton, A. J. Davison, and I. D. Reid. Locally planar patch features for real-time structure from motion. In Proc. British Machine Vision Conference. BMVC, Sept. 2004.

E. Rosten and T. Drummond. Machine learning for high speed corner detection. In European Conference on Computer Vision, volume 1, pages 430-443, May 2006.

VXL Project: http://vxl.sourceforge.net/

NASA Vision Workbench Project: http://ti.arc.nasa.gov/projects/visionworkbench/

Klein G,; Murray D. Parallel Tracking and Mapping for Small AR Workspaces. In Proc. International Symposium on Mixed and Augmented Reality (ISMAR'07, Nara).

Received	(September 17, 2008)
Revised	(December 8, 2008)
Accepted	(December 9, 2008)