Journal of Software Engineering and Applications, **2013, 6, 17-21**
doi:10.4236/jsea.2013.67B004 Published Online July 2013 (http://www.scirp.org/journal/jsea)

Development of a Client-Server System for 3D Scene Change Detection

Haoming Wang, Baowei Lin, Toru Tamaki, Bisser Raytchev, Kazufumi Kaneda, Koji Ichii

Graduate School of Engineering, Hiroshima University, Higashi-Hiroshima, Japan.
Email: lin@eml.hiroshima-u.ac.jp

Received April, 2013.

ABSTRACT

In this paper, we present a client-server system for 3D scene change detection. A 3D scene point cloud which stored on the server is reconstructed by (structure-from-motion) SfM technique in advance. On the other hand, the client system in tablets captures query images and sent them to the server to estimate the change area. In order to find region of change, an existing change detection method has been applied into our system. Then the server sends detection result image back to mobile device and visualize it. The result of system test shows that the system could detect change correctly.

Keywords: 3D Change Detection; Client-Server System; Tablet

1. Introduction

Recent surveillance technology developments motivate a regular observation of places such as coast, slopes and highways, where disasters and accidents happened with high probability. At coast, a lot of wave dissipating blocks are placed to decrease the power of waves. If the configuration of the blocks is altered due to erosion, the blocks might be collapsed and the waves would not be dissipated. Hence, any small changes of the face must be alerted. So a 3D scene change detection system will be helpful to defend disasters and accidents.

In addition, people need a practical application to observe 3D scene change. Since a variety of mobile devices are much cheaper and more powerful than before, a 3D scene change detection system on a mobile platform is possible to be implemented. We present a system that is capable of detecting 3D scene change based on query image provided by a mobile platform. Image acquisition is performed on a mobile device, and only selected key frames are sent to a powerful server for change detection. Then the detection results are returned to the mobile device. We embedded an existing change detection method into our system, finally we built a client-server system which are easy to be used for 3D scene change detection.

The rest of the paper is organized as follows. Related work on registration is reviewed in section 2. In section 3, we first describe the change detection algorithm briefly, and then discuss about basic concept in our system. The system implementation test is given in section 4.

2. Related Work

There are some researches focus on 3D change detection because it has become easy to obtain 3D point clouds by using a range finder or 3D reconstruction techniques [1,7,8].

3D change detection methods with a laser range finder have been proposed for environment monitoring and robot navigation. Goncaluves *et al.* [3] proposed a method to visualized 3D displacement map of two 3D laser-scanned scene point clouds. A method proposed by Pollard *et al.* [4] updates probabilistic voxel models by using a new image and models changes by a probability of seeing observed pixel color. This method can deal with any images with arbitrary (but known) cameras, however, it strongly relays on GMM color models and hence not adequate to objects with similar colors.

Taneja *et al.* [5] proposed a method for computing 2D inconsistency maps combined with MRF and project color difference of images back to 3D surfaces for 3D scene change detection. The irrelevant changes such as walking people, cars and vegetation are excluded by classifying those changes. This method uses color information of images which is not stable for illumination changes.

Lin *et al.* [2,6] proposed an approach which is similar to the work of Taneja *et al.* [5] to utilize 2D images with 3D scene geometry. Their key contribution is to use local feature descriptors for detecting changes instead of color information, because those are well known to be robust to illumination changes, and invariant to many transformations including rotation, translation, and scaling.

Our system chooses Lin's algorithm as our image processing algorithm on the server side, the details will be introduced in next section.

3. System Overview

3.1. Change Detection Algorithm

We use Lin's method to detection change in the server side. In their algorithm, temporal changes are detected by using 3D scene geometry at time A (*reference*) and the image of the same scene at time B (*query*). To quickly detect changes and visualize them for operators at a regular observation, the method uses 3D-2D matching between a 3D reference scene and 2D query images followed by 2D-2D feature comparison between 2D training and query images, which is superior to either of 3D-3D or 2D-2D matching.

In the case of 3D-3D matching [3,9,10], two 3D scenes are registered for change detection. This may be able to provide a detail comparison between reference and query 3D scenes, however, this approach is not suitable because 1) online and real-time 3D reconstruction is very difficult and still a challenging problem, and 2) a 3D laser range finder is not easy to handle at the places of our task because of its heavy weight and the need of electric power. In the case of 2D-2D matching [11-13], images (or videos) taken by a fixed camera are compared to detect changes, or more commonly, moving objects. This approach is also not appropriate for the detection purpose: a number of fixed cameras are needed to cover the whole scene of the observed place, hence, it is impractical if in particular the scene is wide and large. If the scene is observed by a hand-held camera for taking a number of 2D images, the problem above may not arise. However, reference and query 2D images are very difficult to register unless they have a large overlapping area in each pair of the images.

In contrast, Lin's approach uses the combination of 3D-3D and 2D-2D matching. Assume that 3D scene geometry is given but only at time A as a reference 3D scene, hence no 3D range finder is necessary. At time B (query), a hand-held camera is used to take images of the same scene, and the reference 3D geometry is used for 3D-2D matching for camera pose estimation. This enables us to perform a robust 3D-2D matching.

3.2. Model Overview

In order to develop a client-server 3D scene change detection system, we choose an android based tablet device as a client for image acquisition and result visualization in our system. We implement a client application which can communicate with a server through network connection. Here we give an example of how a user uses our client-server system. The user takes a tablet device to a coast where he wants to do change detection, then, take photos of the wave dissipating blocks. The photos will be sent to the server for change detection calculation. Then the detected change points coordinates' information will be returned back to the client and the client application draw these points on the query photos. User can immediately observe the changed area. **Figure 1** shows the general modules of our system.

3.3. Combination with Change Detection Algorithm

In our development, we want to combine Lin's change detection algorithm into our detection system. So, to make the embedded algorithm low computational cost and good matching integration is very important. Because the client system and the server system are separated to each other, the connection which we considered about is just via image because of follows:

- Change detection algorithm is only in charge of processing image.
- The server system is in charge of communication with the clients. It only receives query images and sends result images.

The basic concept of the interface design is shown in **Figure 2**. Change detection program monitors query image folder, and server program monitors result image folder. When server receive a query image from clients, it save this image to query image folder, this action triggers the change detection program to read this image and do image processing task, then it saves the result image to result image folder, also this action triggers server program to read the result image and send it to the clients.

Figure 3 shows the processing flow of our system. First, when a query image is received from a client. Server program saves this image into the query image folder. At the same time, change detection program is

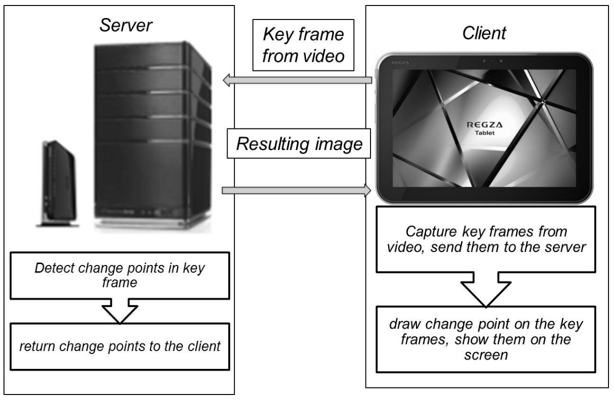

Server
Key frame from video
Client
Resulting image
Capture key frames from video, send them to the server
Detect change points in key frame
return change points to the client
draw change point on the key frames, show them on the screen

Figure 1. Module overview of the client-server system.

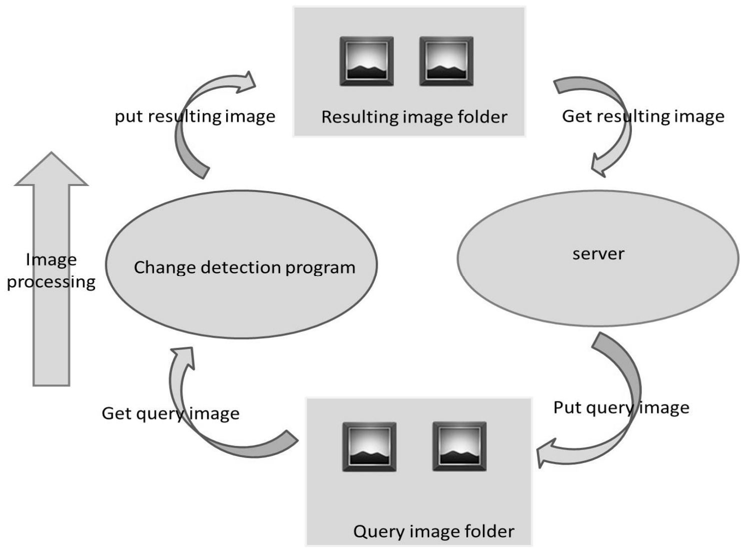
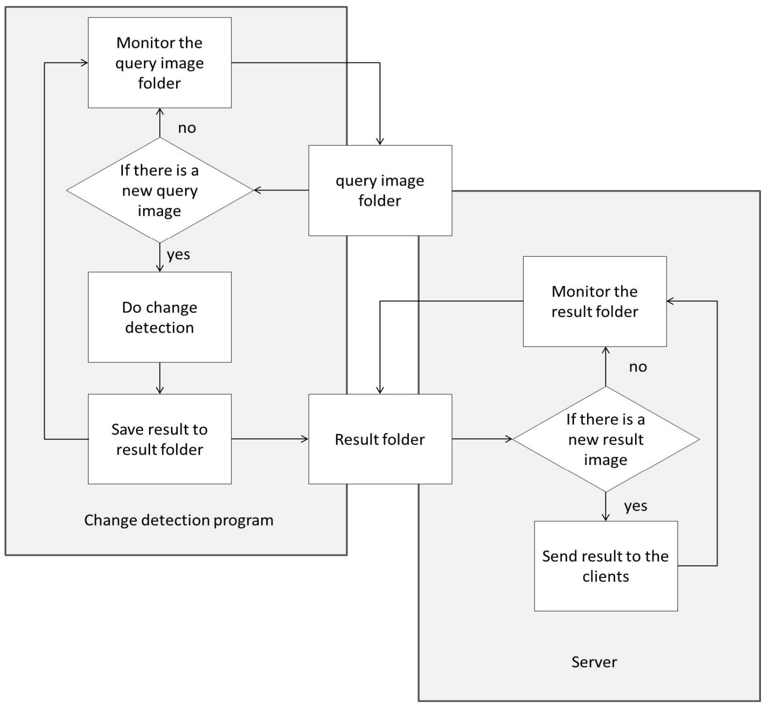


Figure 2. Concept map of interface design.



Monitor the query image folder
no
If there is a new query image
query image folder
yes
Do change detection
Save result to result folder
Result folder
Monitor the result folder
no
If there is a new result image
yes
Send result to the clients
Change detection program
Server

Figure 3. Processing flow of combined systems.

monitoring the query image folder, detecting and saving the result to the result folder. In this processing flow, both server program and change detection program just need to implement a file monitor module for integrating, which is loosely coupled.

4. System Implementation

4.1. System Environment

On server side, we used Dell Latitude E6520 as our server computer, the configuration information of the laptop are as follows:

- Operation System: Fedora 16
- Processor: Intel(R) Core i7-2760QM CPU @ 2.40GH
- Memory (RAM): 8GB

On the client side, we used REGZA Tablet AT700 and SONY XPERIA SGPT12 as the client tablets. REGZA Tablet AT700 is based on Android 3.2.1 and SONY XPERIA SGPT12 is based on Android 4.0.3. The reason we used two tablets is that we want to test the performance of our system under different version of Android platform. Also we tried to use them to test the multi-client supporting ability of our server program. We used Wi-Fi as the network environment. The test implementation details are shown in next section.

4.2. Implementation

4.2.1. Server Implementation

Before 3D change detection, we need to prepare the 3D model dataset of time A. The point cloud of small artificial blocks was generated by 3D reconstruction with Bundler [1] followed by Patch-based Multi-view Stereo (PMVS2) [7,8]. **Figure 4** shows one example image for 3D reconstruction. The 3D point cloud is shown in **Figure 5**.

Then we ran the server program of our system to wait for images sent from clients. Once the system received the images, the change detection program was implemented, and then sent the detection result back to clients.

4.2.2. Client Implementation

After the preparation of server side, we ran a client, and the main interface is shown in **Figure 6**. In this main interface, the left side is the photo preview panel; the right side is the directory list of result images of other clients. Here we named each client by their device serial number. Users can check the results of other clients from different folders. After we pressed the capture button, an image of the same scene as what prepared in sever side was captured, and sent to the server waiting for the change detection result. Once there was detection result back to client, it was visualized on the right panel of the main interface as shown in **Figure 7**.

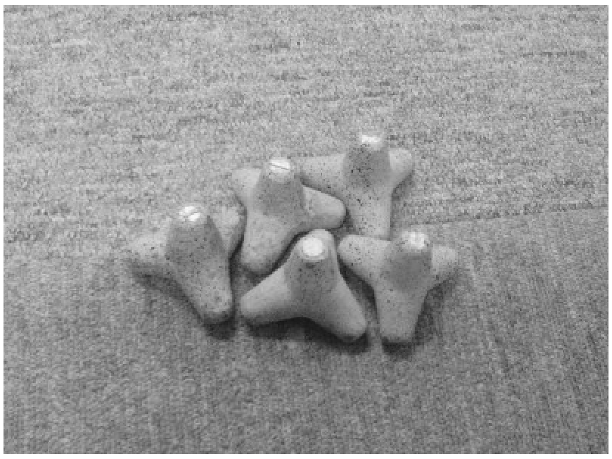


Figure 4. An image for 3D reconstruction.

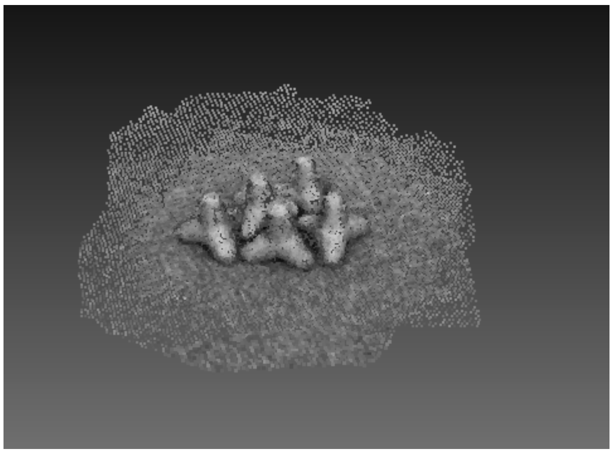


Figure 5. Reconstructed 3D point clouds of small blocks.

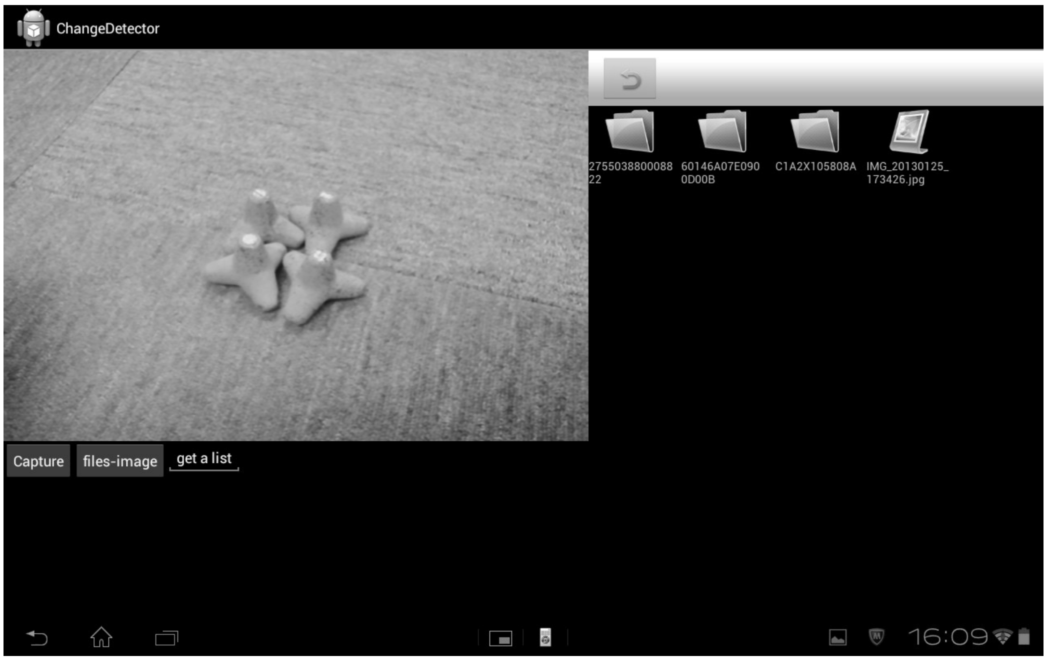
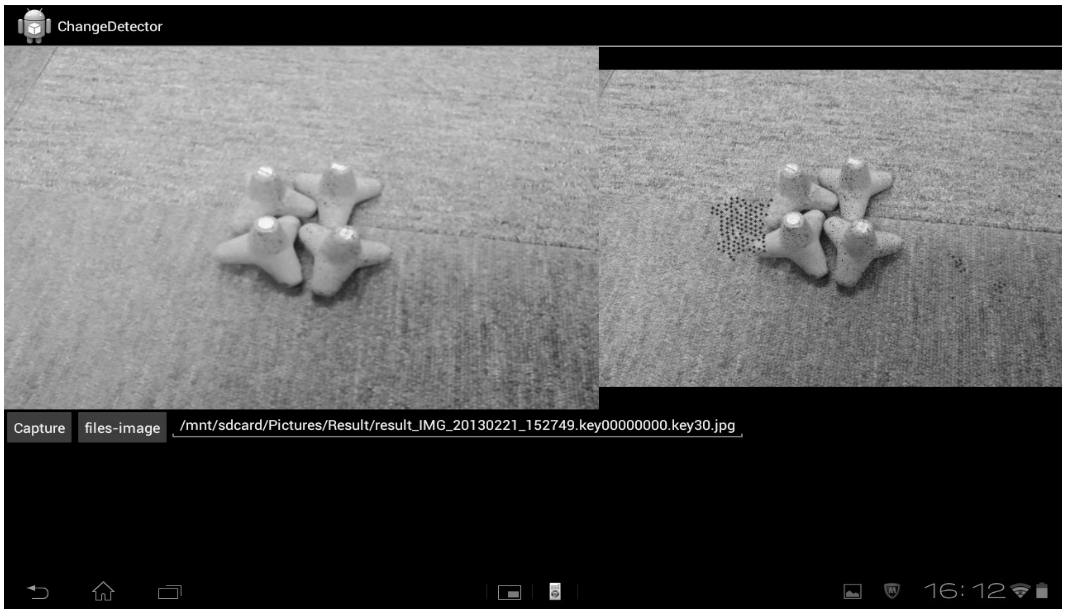


Figure 6. Main interface of client for capturing query image.



/mnt/sdcard/Pictures/Result/result_IMG_20130221_152749.key00000000.key30.jpg

Figure 7. Main interface of client for visualizing change detection result.

On the result image, red points specify an area that change happened. Comparing to **Figure 4**, it is obvious that the red point area is absolutely the change area (a block removed). Although there are some noises on the result image because of the limitation of the change detection algorithm, the result is reasonable acceptable.

5. Conclusions

In this paper, a client-server system for 3D change detection has been introduced and preliminary test has been implemented. In our future work, we will accelerate the implementation speed, improve the detection accurate and make better UI design.

REFERENCES

[1] N. Snavely, S. M. Seitz and R. Szeliski, "Modeling the World from Internet Photo Collections," *International Journal of Computer Vision*, Vol. 80, No. 2, 2008, pp. 189-210. doi:10.1007/s11263-007-0107-3

[2] Y. Ueno, B. Lin, K. Sakai, T. Tamaki, B. Raytchev and K. Kaneda, "Camera Position Estimation for Detecting 3D Scene Change," *18th Korea-Japan Joint Workshop on Frontiers of Computer Vision*, 2012.

[3] J. G. M. Goncaluves, "3D Laser-Based Scene Change Detection for Basic Technical Characteristics/Design Information Verification," *ESARDA Bulletin*, No. 45, 2010, pp. 66-72.

[4] T. Pollard and J. Mundy, "Change Detection in a 3D World," *Computer Vision and Pattern Recognition*, 2007.

[5] A. Taneja, L. Ballon and M. Pollefeys, "Image Based Detection of Geometric Changes in Urban Environments," *International Conference on Computer Vision*, 2001.

[6] B. Lin, Y. Ueno, K. Sakai, T. Tamaki, B. Raytchve, K. Kaneda and K. Ichii, "Image Based Detection of 3D Scene Change," *IEEJ Transactions on Electronics, Information and Systems*, Vol. 133, No. 1, 2013, pp. 103-110. doi:10.1541/ieejeiss.133.103

[7] Y. Furukawa and J. Ponce, "Accurate, Dense and Robust Multi-View Stereopsis," *Computer Vision and Pattern Recognition*, 2007.

[8] Y. Furukawa and J. Ponce, "Accurate, Dense and Robust Multi-View Stereopsis," *Pattern Analysis and Machine*, Vol. 32, No. 8, 2010, pp. 1362-1376. doi:10.1109/TPAMI.2009.161

[9] J. Ryle and N. Hillier, "Alignment and 3D Scene Change Detection for Segmentation in Autonomous Earth Moving," *International Conference on Robotics and Automation*, 2011.

[10] B. Neuman, B.Sofman, A. Stentz and J. A. Bagnell, "Segmentation-based online Change Detection for Mobile Robots," *International Conference on Robotics and Automation*, 2011.

[11] B. Micusik, "Trajectory Reconstruction from Non-Overlapping Surveillance Cameras with Relative Depth ordering Constraints," *ICCV*, 2011.

[12] Y. Kameda, T. Takemasa and Y. Ohta, "Outdoor See-Through Vision Utilizing Surveillance Cameras," *International Conference on Computer Vision*, 2004.

[13] J. Melo, A. Naftel, A. Bernardino and J. Santos-Victor, "Viewpoint Independent Detection of Vehicle Trajectories and Lane Geometry form Uncalibrated Traffic Surveillance Cameras," *International Conference on Image Analysis and Recognition*, 2004