Reconstruction of a three-dimensional structure from a sequence of views using SIFT detector and GPU parallelism

Diego Aracena-Pizarro
Universidad de Tarapacá, School Industrial, Informatics and System, Computer Eng. Area, Arica, Chile
daracena@uta.cl

Nicolas Daneri-Alvarado
Universidad de Tarapacá, School Industrial, Informatics and System, Computer Eng. Area, Arica, Chile
ndaneria@gmail.com

ABSTRACT

This work focuses on perform 3D reconstruction from real images captured by a camera in properly controlled positions, with the aim of model an environment through successive views, in order to extract existing features on the environment, in this case, points of interest and reconstruct 3D surfaces.

The main contribution of this paper consist in offering a new way to perform virtual reconstruction based on the combination of different existing vision techniques, adapting it to the needs of the environment, focusing on the SIFT method as the axis of the process, besides better integration between the epipolar geometry techniques and robust methods, and use parallel programming techniques specifically the GPU parallelism. Initial tests are performed in C + +, CUDA, TLC and VRML.

Keywords

SIFT, dense matching, Delaunay triangulation, 3D reconstruction, parallel programming, CUDA

1. INTRODUCTION

In literature different proposals already exists about the 3D object reconstruction using 2D images sequence, which presents different applications based on vision techniques. Previous work developed, such as Gaffar [Gaf07a], Remondino and El Hakim. [Rem06a], Grandon et al [Gra07a] has shown that the initial process of feature extraction is critical for successful outcome, which results in a near-real reconstruction. It is for this reason that this research focuses on using techniques of detection and matching based on SIFT operator, instead of Harris and Stephan, as in previous work.

This work focuses on using digital cameras to capture 2D images and generate a 3D reconstruction model because it allows a sequence of images, extraction of relevant information of features (points) to reconstruct 3D surface and shape captured images in a visual environment.

Hayden and Pollefeys [Hey04a] presents a 3D reconstruction process, which consists of a sequence of input images and a 3D surface model output. The stages are as follows: List of images, structure and motion recovery, dense correspondence model building, you get a 3D photo-realistic result in a visual environment. The same idea is taken by Grandon et al. [Gra07a] except that the process is divided into two major modules, the first responsible for obtaining the depth map and the seconds 3D model.

This work arises from the proposed [Hey04a] with modifications and upgrades necessary to achieve and perform 3D reconstruction

The architecture of the system for the reconstruction step is described in the second section. The next section will describe the tests and analyze the results obtained with the methods used. Finally, conclusions are mentioned, as well as contributions and future work that emerges from this proposal

2. SYSTEM ARCHITECTURE

In architecture (Figure 1) is considered as the main entrance of the system in a sequence of two views which are processed to obtain data from which can be reconstructed the 3D model and texturing.

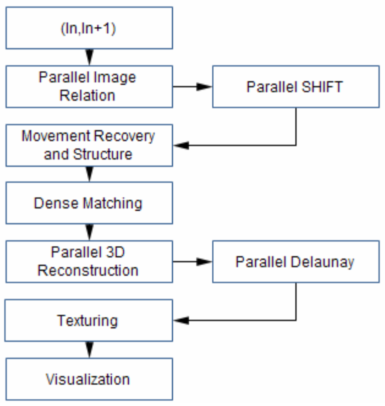

Figure 1. System Architecture

It makes the process of camera calibration in order to obtain the intrinsic parameters, applying the method of Zhang [Zha00a]. We have favored this method for its simplicity and good results compared with others[Ara05a].

Image Value

This phase takes as input a pair of images. Initially minutiae are detected in both images using the SIFT method [Bei97a] but oriented towards parallel programming so that the search of characteristic points is accomplished by dividing the image into equal parts called subpicture to be analyzed individually for each core of the GPU to get the points, by special treatment of the edges of each subimage.

The characteristic point matching is performed by a correlation around the points using the algorithm first searched by Best-bin(BBF) [Bei97a]. It also achieves a cloud of points using RANSAC stronger for optimization.

Structure and Motion Recovery

From the set of points for 2D, 3D depth is determined with respect to the second view. The goal is to create the depth map, which is determined by triangulation of corresponding pairs of points. To map deep, it must first perform the rectification of the image pair.

The rectification process is performed by proposing Fusiello reported in [Fus08a] for calibrated images is why the extrinsic parameters are calculated within the same algorithm from the minutiae delivered. This method was chosen to be compact and easy to modify for the purposes of this paper.

Calculated parameters R and T from Fusiello [Fus08a] are applied to these images to obtain the correct orientation to achieve this correction and compute the disparity map. With this result we can compute the dense correspondence.

Construction of model

First you get a cloud of points through triangulation and then apply Delaunay [Del34a] but take advantage of parallel programming, for the surface formed by a mesh of triangles.

With the triangulation done to bring the points of interest matching 3D coordinates, note that not all points are taken. The most important points for the reconstruction by a variant of the RANSAC algorithm, using as a criterion the minimizing of the normal distance to the straight edge defined as part of the structure to reconstruct, in relation to a preset threshold, allows you to select the best spotsn in the 3D environment. In addition, combined with an algorithm of Canny edge detection [Can86a] this is done so that the figure accurately retains its original form. The steps are:

- Apply RANSAC to the 3D coordinates
- Apply the edge detector to any view with which they work and then eliminate all 2D points which are outside the region formed at the edges detected
- Remove the 3D coordinates related to the 2D point removed.

Finally we come to make the formation of the surface, so it uses the Delaunay method [Del34a] but so it is oriented with parallel programming, so that the construction of the triangles needed for the creation of the mesh is done by dividing the total area in which the model is built into cubes and each GPU core of the construction works of triangles in each bucket, making a special treatment for points near the edges of the cube, for when we have cases of triangles formed by dots of different cubes.

Model Texturing

We carried out the mesh texturing, for it is exported to AutoCAD DXF format [Aut09a], to texture element meshes, that are used to make a guidance algorithm using DLT (Direct Linear Transformation).

3. PARALLELIZATION

It has been decided to perform parallelization using GPU using CUDA throwing as many threads as have the image pixels. Each thread will perform the operations necessary for the implementation; after several tests it was concluded that the GPU better handled one-dimensional blocks size than that of two or three.

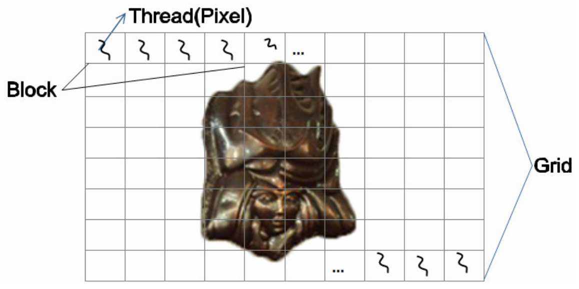

Figura 2. Distribution of the work of the threads.

According to [Nvi09a] it is recommended to launch multiple blocks of 64 threads to make more efficient memory management and therefore the times are lower. Once you reach this point these are tested for block sizes different thread such as 64, 128, 256 and 512 threads. And the results obtained were very similar in all cases. In the case of 512 wires, although the GPU can be used to launch up to 512 threads simultaneously, because the memory required by the wire, this solution is not viable and errors were running. It is necessary to clarify that the size of the grid is matched to image and treated on the size of it.

Upon completion of this section and in view of the few differences between the different sizes of block and grid, it was proposed that the program was more functional and self-adjusts to any size image. To do this, we defined a variable called Ancho_Bloque, which contains the x dimension of the block of threads to launch, and launch grid blocks needed to cover the size of the image processing.

it was decided that CUDA also applies the Delaunay triangulation (same way SIFT parallelization), the structure of Grid. This requires extra calculations to determine the coordinates of a point unit in 3D space from the variables inside the *thread.*Considering that our points have three dimensions, which are determined according to the following equations:

$$thread_X = Pos_X * \underline{(blockID_{XX} * \mathrm{mod}(blocks_X))} + threadID_X$$

$$thread_Y = Pos_Y * \underline{(blockID_{XX} / \mathrm{mod}(blocks_X))} + threadID_X$$

$$thread_Z = Pos_Z * blockID_{XX} * blocks_X + threadID_Z$$

Blocksx is the number of triangles needed in the X axis in 3D space, and calculations that must be done are highlighted because the grid structure can only be 2D. Also, these are calculations that minimize the execution time, because they do not vary with the *thread,* but do so only on the basis of the triangle to which they belong.Therefore, all *threads* of one the triangle have the same value for such calculations as they indicate, in short, the index of triangle in the X, Y, Z in 3D space.

Following this reasoning, it generates a first *kernel* that generates the calculations for each of the triangles that are used later, and after its implementation. Thereby performing a similar thought to the SIFT parallelization.

4. TESTS

The images were captured with a digital camera Sony DSC-S930, 10.1 megapixels and 3X optical zoom, and worked on a Dell Vostro 1500 Notebook, Core 2 Duo 2.2GHz processor, 3GB of RAM and a 512MB GeForce 8600GT VGA.

A comparison was conducted to estimate the improvement in the speed of the process when using CUDA, both SIFT as Delaunay. Three tests were conducted for each variable in which the window size (Tv) implies a larger number of triangles generated in the process of triangulation, a larger size, greater amount of data generated and SIFT for varying the image size (Ti) to generate different amounts of image partitions.

The first test images captured were of the Amerindian Mural at the Universidad de Tarapacá, with the cloud of points obtained necessary to generate the desired surfaces. Finally, the texturing process took place for parts or sections of the texture of the object or figure in the study of images. It generated the 3D model in DXF format and correspondence between 3D points and 2D model of the image obtaining the result (Figure 3).
For the second test an external environment was considered called "Presencias Tutelares" (Figures 4-5), sculptures located 27 miles south of the city of Arica, which represent an Andean worldview. This Followed the same process leads to model the 3D sculptures (Figures 6-7).

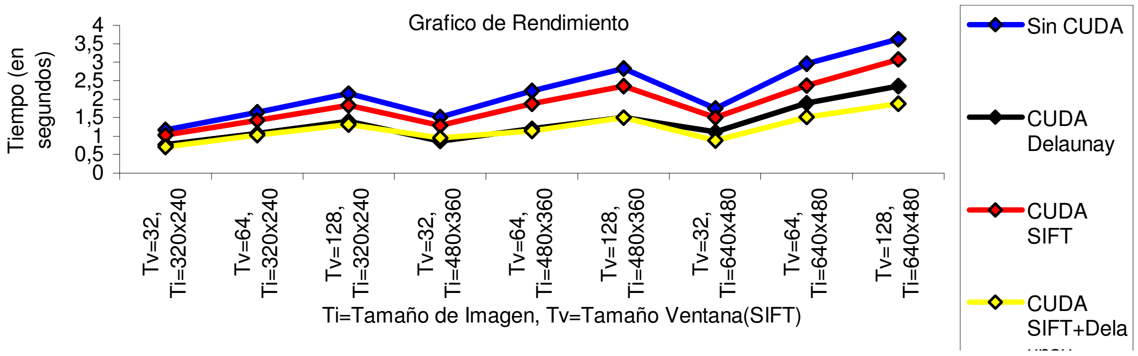

Figure 3. Face's Results

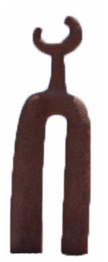
Figure 4.Sculpture 1

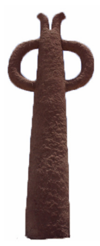
Figure 5.Sculpture 2

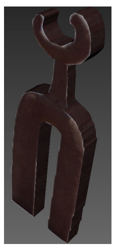
Figure 6. Sculpture's 3D Model

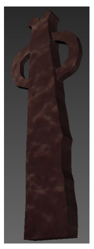
Figure 7. Sculpture's 3D Model

Finally the second scenario we stood almost in an ideal case, since the reconstruction centers practically at the edges of the figure, since within it presents a uniform appearance generating few minutiae therefore the results are closer to reality. The result execution time is show in Figure 8.

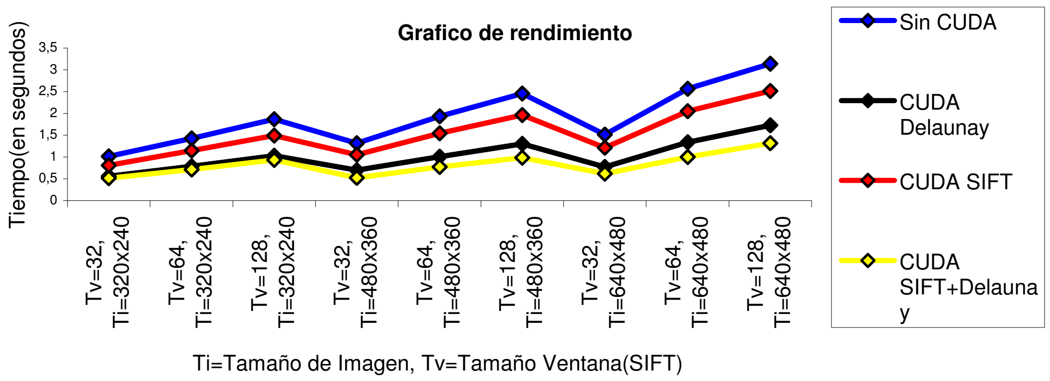

Figure 8. Results "Presencias Tutelares"

As shown in the pictures, the difference is more or less significant, in fact it implies a change in execution time of 40%, with a 1.66 speed-up on cases of less data, more data for the performance starts increased to an exponential trend achieving 3.34 speed-up in the best which is a significant improvement, therefore we can conclude that a greater amount of data will be a much better process performance. Furthermore, it is remarkable in all tests.

5. CONCLUSION

A system made purely with computer vision allows automation of the reconstruction process and its use can be generalized to other areas that require three-dimensional reconstruction from multiple views.

Regarding the parallelization in light of these results, we can say, but are not generalizing these algorithms to greatly improve its performance to the surveyor and run it on a graphics card.This allows us, therefore, to run on a GPU 3D reconstruction system from the time of the acquisition and to display, especially considering that the card with in use is one quarter of the potential of current GPU

Extrapolating the results to a more general algorithmic profile, one can say that as long as the facility is to parallelize the algorithm, ie, no dependence between different *threads,* this does not require excessive resource use, and there is sufficient volume of data. In all probability this been an improvement in adapting the algorithm to run on GPUs *multi-thread.*

6. FUTURE WORK

Regarding the parallelism, this should be focused on testing various distribution strategies and clustering, which will improve performance.

7. REFERENCES

[Ara05a] Aracena, D.,Campos, P., and Tozzi, C., "Comparison of digital camera calibration techniques". Revista de la Facultad de Ingeniería - Universidad de Tarapacá. Vol. 13 Nº 1. 2005.

[Aut09a] Autodesk, DXF Reference, 2009

[Bei97a] Beis, J., and Lowe, D.G "Shape indexing using approximate nearest-neighbour search in high-dimensional spaces", Conference on Computer Vision and Pattern Recognition,Puerto Rico, pp. 1000–1006, 1997.

[Can86a] Canny, J.F., A computational approach to edge detection. IEEE Trans Pattern Analysis and Machine Intelligence, 8(6): 679-698, Nov 1986.

[Del34a] Delaunay B., "Sur la sphere vide". Bull. Acad. Sci. USSR VII 7, pp. 793 - 800. 1934.

[Fus08a] Fusiello A, Quasi-Euclidean Uncalibrated Epipolar Rectification, Pattern Recognition, 2008. ICPR 2008. 19th International Conference , pp 1-4, 2008

[Gaf07a] Gaffar A, 3D Reconstruction of Objects from Image Series, Publisher: Institut of Numeric Mathematic, University of Göttingen, 2007

[Gra07a] Grandon, N., Aracena. D., Tozzi, C., Reconstrucción De Objeto 3D A Partir De Imágenes Calibradas. Ingeniare. Rev. Chil. Ing. [Online]. 2007, Vol.15, N.2 [Citado 2009-07-11], Pp. 158-168

[Hey04a] Heyden, A., and Pollefeys, M, *Multi-view Geometry*, in Emerging Topics in Computer Vision, G. Medioni and S. B. Kang (Eds.), Prentice-Hall, capitulo 3, 2004

[Nvi09a] NVIDIA CUDA Programming Guide (version 1.0). Chap. 5, pp. 52-53, 2009.

[Rem06a] Remondino, F., El-hakim, S., Image-based 3d modelling: a review, Photogrammetric Record, 21(115), pp. 269-291 2006

[Zha00a] Zhang A., "A Flexible New Technique for Camera Calibration", IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 22 Nº 11, pp. 1330-1334. 2000.