AN EXPLICIT GROWTH MODEL OF THE STEREO REGION GROWING ALGORITHM FOR PARALLEL PROCESSING

Dongjoe Shin*, and Jan-Peter Muller

Imaging Group, Mullard Space Science Laboratory, Department of Space and Climate Physics,
University College London, Holmbury St. Mary, Dorking Surrey, RH5 6NT
(ds2, jpm)@mssl.ucl.ac.uk

Commission V, WG V/4

KEY WORDS: Dense reconstruction, Stereo Region Growing, Parallel Processing, Adaptive Least Square Correlation, GOTCHA

ABSTRACT:

GOTCHA is a well-tried and tested stereo region growing algorithm, which iteratively applies Adaptive Least Square Correlation (ALSC) matching to the adjacent neighbours of a seed point in order to achieve a dense reconstruction with sub-pixel precision. It is, however, a computationally expensive algorithm as every seed point collected by the ALSC matching produces quadrants or octants of new matching candidates. Accordingly, the computational complexity increases exponentially as the stereo matching region grows. To expedite the matching process of a traditional GOTCHA, this paper proposes a parallelised stereo region growing algorithm called a MT-GOTCHA. To achieve data parallelism, the proposed method initially divides a stereo image from arbitrary distributed seed points, which are able to employ multiple GOTCHA's. In addition, since it estimates a cluster of neighbours using a non-linear diffusion equation and performs multiple ALSC processes in parallel to verify local matching candidates, more tiepoints are obtained within less processing time. Experimental results demonstrate the proposed method can reduce the processing time of a dense reconstruction at a reasonable cost of memory consumption.

1. INTRODUCTION

A major goal of "shape from stereo" is generally concluded to obtain accurate and complete reconstruction from an overlapping image pair. However, it is often difficult to satisfy both conditions at the same time, particularly when dealing with images containing a matching ambiguity (e.g., significant scene distortion due to a wide baseline separation or containing either a repetitive pattern or homogeneous texture). Thus, matching accuracy tends to trade off against matching completeness and vice versa depending on their application.

For example, a traditional matching approach for close-range developed for dense reconstruction in the computer vision community exploits the epipolar constraint (i.e., a correct correspondence should only be found within an epipolar line in the other image). Consequently, Dynamic Programming (DP) has served as a standard method for dense reconstruction but it often includes outliers as the epipolar constraint is not sufficient to define a tiepoint at times (i.e., DP results only satisfy the global matching constraint defined within an epipolar line and not for each and every potential pixel pair). Moreover, matching consistency between epipolar lines is not guaranteed without a proper post-process (Scharstein and Szeliski, 2002; Ohta and Kanade, 1985).

On the other hand, the remote sensing (far-range) community has developed a stereo region-growing algorithm to achieve a dense reconstruction from different reasons, such that some matching difficulties noticed in close range images are not generally found but the linear epipolar constraint no longer holds. For instance, occlusion and significant geometric (e.g., projective) distortion are hardly observed in orbital stereo pairs. However, it is difficult to define a linear fundamental matrix that maps a point to a line in satellite imagery from a push broom sensor (Otto, 1988; Kim, 2000).

One good example of a stereo region growing algorithm is the Gruen-OTto-CHAu (GOTCHA) algorithm (Otto and Chau, 1989), which is based on the Adaptive Least Square Correlation (ALSC) (Gruen, 1985) combined with a 2D region growing algorithm. For example, given initial matching results, referred to as "seed points" in (Kim and Muller, 1996), GOTCHA can increase the number of matching pairs considerably as it does not limit growing directions within an epipolar line as a seed point can grow in both the *x* and *y* directions if ALSC confirms that the accuracy of a matching candidate is acceptable.

However, there is one major disadvantage which is that the computational complexity of the GOTCHA algorithm is significant compared with its counterpart DP matching. This is partly because ALSC needs to be performed sequentially at every neighbouring pixel of a new growing position. Also, growing results need to be ordered with respect to matching similarity in order for GOTCHA to decide the next "best" growing point. Consequently these two procedures (i.e., ALSC followed by sorting) become a major bottleneck of the GOTCHA process as the number of seeds increases. To address this problem, this paper proposes a modified GOTCHA algorithm called MT-GOTCHA, which expedites the growing process by employing multiple processes in parallel.

To realise a parallel process, the proposed method divides an image into multiple growing regions so that each region has its own independent process. Additionally, multiple ALSC processes associated with a single seed point are also parallelised. This means that the proposed method could achieve micro and macro level parallelism, e.g., multiple

* Corresponding author.

GOTCHA's can operate at a macro level with multiple ALSC processes at a micro level. To achieve this, MT-GOTCHA models the growth of a seed point by two boundaries called inter and intra boundary respectively, i.e., the intersection of two boundaries defines the final growing region of a seed point.

This paper is organised as follows. Section 2 reviews more details about the ALSC and GOTCHA algorithm and identifies potential problems. In Section 3, the proposed stereo region growing method is explained particularly regarding how the growing boundaries are defined in MT-GOTCHA. Finally, experimental results and conclusions are presented in Section 4 and 5, respectively.

2. STEREO REGION GROWING

2.1 ALSC

Adaptive Least Square Correlation is the core-matching algorithm used in GOTCHA. It can estimate a true correspondence from an approximated tiepoint by changing the shape of a matching window iteratively. Thus, if a given seed point is located close enough to the true correspondence, ALSC can move the initial position and converge at the true correspondence.

Suppose $I_l(\dot{x})$ is a functional which maps a 2D positional vector $\dot{x} = [x \quad y]^T$ to an intensity scalar value of a left image, and also assume that it is continuous and differentiable at any point within a left image domain. Then, a tiepoint can be defined as a point pair $(\dot{x}_i, \dot{x}_j)$ which satisfies

$$I_l(\dot{x}_i) = I_r(\dot{x}_j) + N(\dot{x}_j) \tag{1}$$

where $N(\dot{x})$ is a Gaussian functional that represents image noise at a point $\dot{x}$ and similarly $I_r(\dot{x})$ is an intensity functional defined in a right image.

If the right position of an initial tiepoint is given around the true position and assume the error between the true and the initial position is introduced by a local affine distortion, then the error vector $\dot{d}_j$ can be found by minimising a cost function,

$$\min_{\dot{d}_j} \left| I_l(\vec{x}_i) - I_r(\vec{x}_j + \dot{d}_j) \right| \tag{2}$$

where $\vec{x}_j$ denotes an approximate right position given initially and $\dot{d}_j$ is the offset between the true position and its approximation such that $\vec{x}_j = \vec{x}_j + \dot{d}_j$ and a point pair $(\dot{x}_i, \dot{x}_j)$ defined in a tiepoint is related by a local affine transform, i.e.,

$$\vec{x}_l = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \vec{x}_r + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \tag{3}$$

where a_{ij} denotes affine distortion parameters and t_x and t_y represents a translation in x and y direction.
By employing Taylor's theorem up to the first order term, the second term of the cost function (2) can be expanded with respect to the approximated right position, $\vec{x}_j$, i.e.,

$$I_r(\vec{x}_j + \vec{d}_j) = I(\vec{x}_j) + \frac{\partial I(\vec{x}_j)}{\partial x} dx + \frac{\partial I(\vec{x}_j)}{\partial y} dy \tag{4}$$

where dx and dy is delivered from (3). Thus, a closed form solution of (2) is found in the least squares sense and ALSC recursively applies (2) with the updated solution until it converges at a certain point.

It is also important to mention that the cost function shown in (2) appears to find a 2D vector $\dot{d}_j$ but it is, in fact, parameterised by (3) so that at least three points pairs (i.e., each pair provides two conditions for an affine transform) are required to avoid a singular system of equations. These point pairs are found from a matching window and (2) is therefore normally over-determined.

Fig. 1 demonstrates the performance of ALSC, which started from an incorrect right position (see a green cross in the right image) but found the correct position after 15 iterations. One can notice that ALSC rapidly converges at the true point and a square box shown in both images represents an initial matching window and multiple rectangles shown in the right image visualises how ALSC distorts the square shape iteratively from the initial matching window.

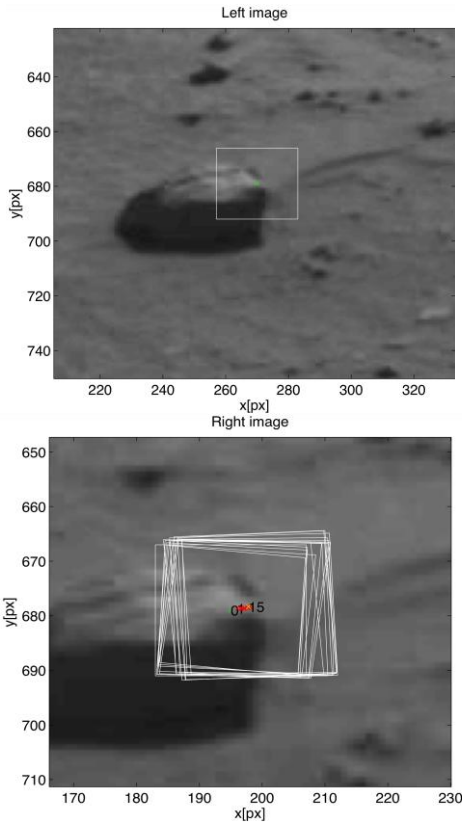


Figure 1 Example of ALSC, where an initial point marked as a green cross in both images. The right image shows an updated corresponding point after 15 iterations as a yellow cross whilst intermediate right positions are highlighted as a red colour.

2.2 GOTCHA

GOTCHA can be seen as an extended version of a general 2D region growing algorithm used in image segmentation. Similar to a traditional binary segmentation, GOTCHA grows a seed

point by adding adjacent neighbours to a seed point recursively. A difference is it uses neighbour point pairs to deal with a stereo image and avoids a false neighbour pair which fails to produce a refined correspondence within predefined ALSC matching error. Suppose that a set of initial tiepoints is given as

$$T = \{t_k = (\mathbf{x}_i^k, \mathbf{x}_j^k) : | I_l(\mathbf{x}_i^k) - I_r(\mathbf{x}_j^k) | < \tau\} \quad (5)$$

where τ is a matching threshold, then the 8-connected neighbours of the k^{th} tiepoint are defined as

$$N_8(k) = \{n_l = (\mathbf{x}_m, \mathbf{x}_n) : | \mathbf{x}_m - \mathbf{x}_i^k | = 1, \ | \mathbf{x}_n - \mathbf{x}_j^k | = 1\} \quad (6)$$

In this case, initial seed point pairs of GOTCHA (i.e., the first place from which the algorithm begins to grow) is defined as all neighbours of an initial tiepoint set, i.e.,

$$S^{(0)} = \bigcup_{k=0}^{|T^{(0)}|} N_8(k) \quad (7)$$

where $S^{(0)}$ represents a set of seed points and the number in the parenthesis written as a superscript denotes the number of iterations that GOTCHA takes to grow. Once an initial seed set is ready, GOTCHA recursively adds a new seed point pair if any point pair in $S^{(i)}$ produces a saturated result from ALSC and this iteration continues until $| S^{(i)} | = 0$.

To explain this more clearly, a pseudo code of GOTCHA algorithm is presented in Fig. 2 where *ALSC*(.) shown on line 7 represents a function that returns the dissimilarity of ALSC matching, τ_a represents a matching threshold of ALSC, s'_k on line 8 represents an updated seed point pair of s_k by ALSC, and *sort*(.) in line 12 is a sorting function which orders seed elements with respect to the matching similarity so that a seed point having the higher similarity is tested earlier in the next growing process. As noticed from lines 7 to 10 in Fig. 2, this algorithm increases a search space exponentially. Also, recursive sorting shown in line 12 makes the growing process slower as $|S|$ increases. Please also note that the pseudo

```
1   i = 0
2   While (|S(i)| > 0)
3         T(i+1) = T(i)
4         S(i+1) = S(i)
5         For k = 1: |S(i)|
6               S(i+1) = S(i+1) – {sk}, S(i) = S(i) – {sk}
7               If ( ALSC(sk) < τa)
8                     T(i+1) = T(i+1) ∨ {sk' }
9                     S(i+1) = S(i+1) ∨ N8(|T(i+1)|) – {N8(|T(i+1)|) ∧ T(i+1)}
10              End If
11        End For
12        sort(S(i+1))
13        i++
14  End While
```

Figure 2 A pseudo code of GOTCHA

code presented in Fig. 2 is used to highlight how data in *T* and *S* in (5) and (7) change incrementally. The more efficient algorithm is therefore possible by minimising the number of temporary variables.

As a dissimilarity measure of ALSC matching, a traditional ALSC uses the largest eigenvalue of the covariance matrix of the estimated dt_x and dt_y. Thus, if an updated solution has a large dissimilarity, this means that the solution is not saturated at the fixed point as the increment of the translation parameters is large. This measure normally performs better than the sum of intensity differences or correlation score when matching windows contain homogeneous or repetitive texture that can also gives small intensity difference at any point within a matching window.

3. PARALLELISED STEREO REGION GRWOING

One of the straightforward approaches to expedite the growing process shown in Fig. 2 is performing multiple ALSC matching processes (see from line 5 to 7 in Fig. 2) simultaneously. Otto and Chau implemented this idea in the Multi-Instruction stream and Multi-Data stream (MIMD) parallel processing architecture, where multiple processors establish a farm of worker processors to which a central control processor (i.e., the main CPU of a Unix system) connects via data pipelines (Otto and Chau, 1989). In this method, the main controller processor owns a global GOTCHA process, operating as a master process such as sending multiple ALSC tasks to each worker processor and managing both the priority of the processing queue and image data required for individual ALSC matching, whilst each worker processor simply performs a single ALSC matching and returns the result to the master.

However, this approach only works if the size of an image is small enough due to the narrow data bandwidth of the data pipelines at that time. To address this, Holden et al. (1993) proposed a Geometrically Parallel Streo Matcher approach (also referred to as GPSM), which divides an input image into multiple overlapping rectangular tiles and initiates multiple GOTCHAs after providing each worker processor with relevant tiles for matching (Holden et al., 1993). Accordingly it can also achieve data parallelism and this minimises data transfer rate. For example, since each worker processor has an image tile to which a seed point belongs as well as the overlapping regions from adjacent tiles, each worker process is independent and inter-processor communications are significantly minimised. However, GPSM requires a tool that can regenerate uniformly distributed seed points to initiate every GOTCHA process successfully.

Unlike these previous efforts, the proposed method is more concentrated on task parallelism as the hardware constraint studied in the earlier parallel computing methods has been changed significantly, e.g., the amount of memory and data transfer bandwidth are even ready for the cloud computing. Therefore, a hardware-related issue of parallel computing (e.g., balancing computing load of the multi-processing cores and managing communications between processors) is not investigated in this paper. Instead we focus on identifying independent sub-tasks from the growing algorithm and how to perform them simultaneously without sharing any input data. Therefore, the fundamental idea of the proposed method is to define independent processing tasks with as many as we can. For example, the proposed method is designed to have multiple GOTCHA processes, which also use multiple ALSC matchers when testing neighbours. Thus, to some extent, this can be considered as a way to combine two previous efforts (i.e., one of which uses multi-ALSC and the other of which uses multi-GOTCHA).

As a preliminary requirement of the proposed method, an input stereo image needs to be divided appropriately according to the distribution of initial seed points, i.e., a larger growing area is

assigned to a seed point having higher matching similarity, and we call the boundary between seed points an inter-boundary. In addition, we define another boundary (called an intra-boundary) within the inter-boundary. This is used to define a cluster of local neighbours from a seed point according to a matching confidence. Consequently, intra-boundary can provide more than 8 neighbours if a seed is surrounded by strong matches. Otherwise, it only produces small number of neighbours, which are tested by multiple ALSCs in parallel.

3.1 Inter-boundary estimation

The inter-boundary is the boundary between multiple seed points and it is estimated by a modified Voronoi tessellation that provides a larger area to a seed point having a higher matching similarity. To realise this idea, the initial positions of seed points move according to the sum of gravity pulls between seed points before applying Voronoi tessellation and an initial matching dissimilarity is used as the mass of a seed in the computation of a gravity pull.

Voronoi tessellation, which is also known as a dual of Delaunay triangulation, is a method partitioning a plane into convex sub-regions based on the nearest neighbourhood rule (Aurenhammer, 1991) from input points (called sites). For example, a Voronoi region of the i^{th} point $\dot{x}_i$ of a site set V is defined as

$$R_i = \{\dot{p} : |\dot{x}_i - \dot{p}| \leq |\dot{x}_j - \dot{p}|\} \quad (8)$$

where $\dot{p}$ is a point in an image and $\dot{x}_j \in V - \{\dot{x}_i\}$. Although there are sophisticated implementations of Voronoi tessellation which can improve the computational efficiency considerably such as Fortune's algorithm (Fortune, 1986) in which the computational complexity is known as $O(|V|\ln|V|)$, we adopt the simplest algorithm estimating a Voronoi region by intersecting half planes constructed between sites. For example, a half plane used to estimate R_i from two sites $\dot{x}_i$ and $\dot{x}_j$ is a half infinite plane that contains $\dot{x}_i$ and limited by an orthogonal line intersecting the middle of the line ($\dot{x}_i$, $\dot{x}_j$). Thus, the construction of a half plane is affected by the geometrical distribution of input sites.

In our case, a site set V is replaced with a set of initial seed point pairs $T^{(0)}$, so that the positions of $T^{(0)}$ need to be modified according to their matching dissimilarity values before tessellation. We model the motion of the initial point as a result of the interconnected gravity pull. For example, if a point $\dot{x}_i$ has a higher dissimilarity then it creates a strong pulling force resulting in that other points move toward the point. Consequently, the middle points that define half planes move closer to the point $\dot{x}_i$ and R_i estimated by intersecting every half-plane containing $\dot{x}_i$ becomes smaller.

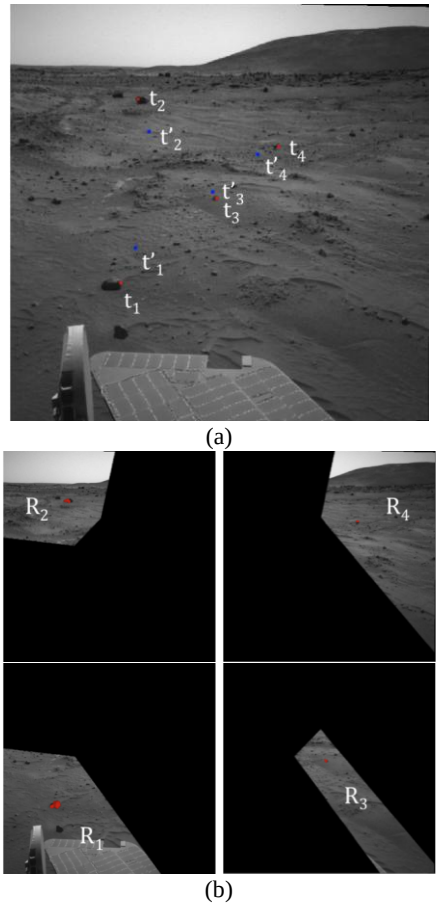
Suppose that there are two tiepoints t_k and $t_l \in T^{(0)}$, where $t_k = (\dot{x}_a, \dot{x}_b)$ and $t_l = (\dot{x}_c, \dot{x}_d)$ then a gravity force that pulls from t_k to t_l in a left image is defined as

$$\dot{\mathbf{f}}_{left}(t_k, t_l) = g \frac{ALSC(t_k) \cdot ALSC(t_l)}{|\dot{\mathbf{x}}_a - \dot{\mathbf{x}}_b|^2} (\dot{\mathbf{x}}_a - \dot{\mathbf{x}}_b) \quad (9)$$

where g represents a gravity coefficient. Additionally, a gravity map $F(T)$ can be defined as a network of gravity pulls between every pair of tiepoints, e.g., an element of F at the k^{th} row and the l^{th} column, $f_{k,l}$ is $\dot{f}(t_k, t_l)$. Thus, $f_{k,l} = -f_{l,k}$ and its diagonal values should be 0 as self-pulling is avoided, i.e.,

$$F(T) = \begin{bmatrix} 0 & \dot{f}(t_0, t_1) & \cdots & \dot{f}(t_0, t_{|T|}) \\ -\dot{f}(t_0, t_1) & 0 & & \vdots \\ \vdots & & \ddots & \dot{f}(t_{|T|-1}, t_{|T|}) \\ -\dot{f}(t_0, t_{|T|}) & -\dot{f}(t_1, t_{|T|}) & \cdots & 0 \end{bmatrix} \quad (10)$$

Once a gravity map is ready, the total gravity that a single point has can be easily found by summing elements of a row (or column) vector of F.



(a)

(b)

Figure 3 Example of an inter-boundary estimation from four input tiepoints, where initial left positions of tiepoints (t_1-t_4) highlighted as red dots are moved to t'_1-t'_4 marked as blue dots (a) and updated tiepoints creates four partitions from Voronoi tessellation (b).

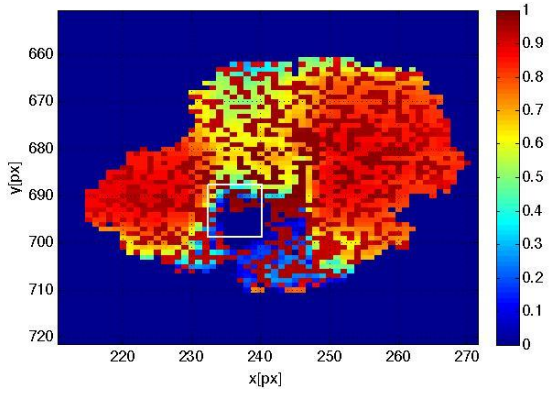
Fig. 3 visualises an example of Voronoi tessellation from four initial tiepoints denoted as t_1-t_4. Fig. 3(a) is the left image of a test stereo data so that dots in Fig. 3(a) represent left tiepoints. Using (10), each tiepoint moves from its initial position from the red to the blue before partitioning the image. Consequently, each tiepoint owns a Voronoi region R_i where GOTCHA is performed later (see Fig. 3 (b)).

3.2 Intra-boundary estimation

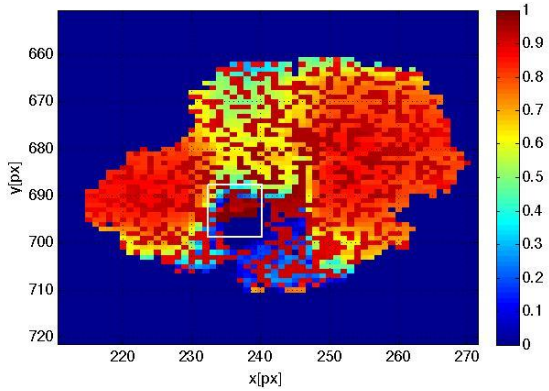
The intra-boundary is the confidence boundary of a seed point, which can estimate how much a seed point possibly grows with

given dissimilarity value after the n^{th} iteration of GOTCHA. A diffusion equation has been adopted to model this non-linear shape of a confidence boundary. Accordingly, the proposed method can include either non-adjacent neighbours of a seed point or a smaller number of neighbours depending on matching dissimilarity of the current state. In fact, an intra-boundary provides a way for GOTCHA to control the number of neighbours and this can be further exploited to balance computing load adaptively. For example, it is potentially possible that GOTCHA optimises the number of ALSC processes depending on the computing load by updating diffusion policy, e.g., a stricter diffusion rule is enforced if less neighbours are required.

An intra-boundary is modelled using a diffusion equation, by assuming that the confidence at current state flows out from sources (i.e., correct matches up to the current state) until it reaches equilibrium. This is a more accurate assumption to



(a)



(b)

Figure 4 Example of a cluster of local neighbours, where a confidence map of R_1 obtained from Fig. 3(b) (a) and local neighbours found by an inter-boundary at point (268, 780) (b).

describe the growth of a seed than a general idea that a matching possibility of a point pair is exponentially reduced as it goes away from known true matches. Suppose $c(\dot{x}, t)$ is a function that estimates a matching confidence at time t and assume it is differentiable, then a diffusion of confidence is given as

$$\frac{\partial c(\dot{x},t)}{\partial t} = \alpha \nabla^2 c(\dot{x},t) \qquad (11)$$

where α is a diffusion coefficient and matching initial confidence at $\dot{x}$ is defined as

$$c(\dot{x},0) = 1 - \frac{ALSC(\dot{x})}{\tau_a} \qquad (12)$$

After sampling continuous time t, (11) can be approximated as

$$c(\dot{x},n+1) = c(\dot{x},n) + \alpha\{L_x c(\dot{x},n) + L_y c(\dot{x},n)\} \qquad (13)$$

where L_x and L_y represent Laplacian operators in terms of x and y and n denotes the n^{th} discrete time. In the proposed method, a 7x7 window centred at a seed point is used to define a supporting area of diffusion, i.e., the previous matching results in this window are also taken into consideration when defining the neighbourhood of a current seed point, and a degree of diffusion is controlled by α and the number of iterations, i.e., n in (13).

Fig. 4 demonstrates a result of an inter-boundary estimation. Fig 4(a) shows a confidence map of 1793 tiepoints obtained from a region R_1 (see Fig. 3(b)) after 618 iterations of a GOTCHA process. The white box shown in both images illustrates a window used to define a supporting area of a seed point at (268, 780). The seed point produces 13 neighbours when n = 5 and they are shown as highest confidence values in Fig. 4(b) for visualisation.

4. EXPERIMENTAL RESULTS

The proposed method has been implemented in JAVA and the algorithm tested in a multi-cored single processor system. Thus, in order to test parallelised tasks with a single processor, the proposed method has been designed to run a single processing task with multiple threads. Also, since we are only focussed on the task parallelism and do not investigate the effect of computational load balancing and the delay from the data transfer, direct comparison with the result from a C implemented parallel algorithm in MIMD architecture is unavailable. However, it is still possible to predict its impact from the normalised performance graph shown in Fig. 5.

To compare a traditional GOTCHA with MT-GOTCHA's having a different number of seed points, we fixed the number of GOTCHA iteration, i = *200* (see Fig. 2) and averaged processing time was measured to count the total processing time of MT-GOTCHA because some of the GOTCHA's may terminate earlier than others in the proposed method. Four performance measures, such as peak memory usage (Max. 823

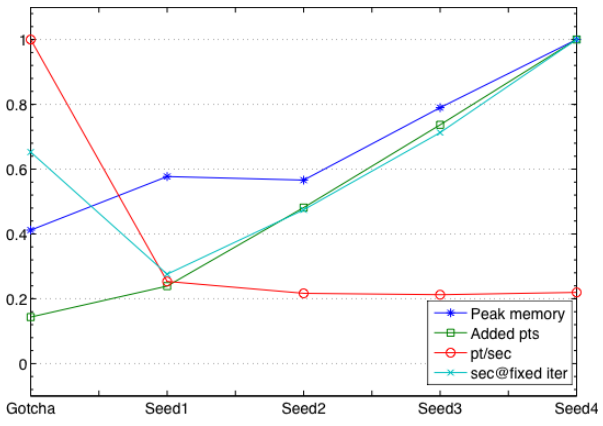
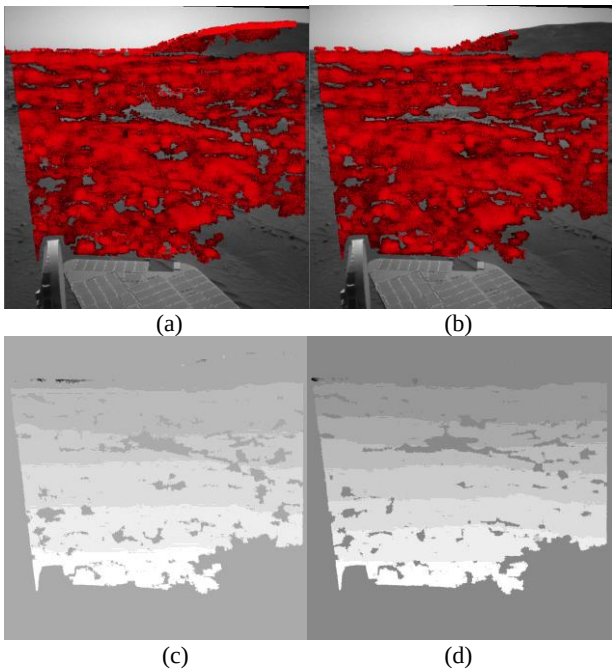


Figure 5 Performance results of the MT-GOTCHA algorithm

[MB]), the number of added points (Max. 2657 [pts]), total processing time (Max. 32.34 [sec]), and a ratio of tiepoint addition (Max. 0.055 [pts/sec]), has been used to evaluate the performance of algorithms and Fig. 5 represents them as *, □ , O, and ×, respectively. From the test results we can conclude that a traditional GOTCHA performs worst in terms of the ratio of the tiepoint addition (381 [pts] / 21.11 [sec]), whilst all MT-GOTCHA's produce similar results. However, GOTCHA shows the smallest peak memory (339 [MB]), which is increased as the number of seeds increases in MT-GOTCHA. For example, MT-GOTCHA with 4 seed points can achieve a 597% increase in the number of added tiepoints at cost of a 143% increase in peak memory usage.

To demonstrate reconstruction completeness, the proposed parallel region-growing algorithm has been applied to stereo images from NASA Mars Exploration Rover mission. Fig. 6 shows that the resulting dissimilarity maps (a)-(b) and disparity maps (c)-(d) from GOTCHA and MT-GOTCHA. In Fig. (a)-(b) the brighter red indicates smaller dissimilarity whilst Fig.6 (a)-(b) uses a brighter colour to represent a larger disparity.



(a) (b)

(c) (d)

Figure 6 An example of dissimilarity map from GOTCHA (a) and MT-GOTCHA (b); a disparity map (c) from (a) and (d) from (b)

The completeness ratios are 52.93% and 52.88%, respectively, so that GOTCHA is slightly better but it includes false tiepoints in the area adjacent to the sky and MT-GOTCHA with a single seed is 3.5 times faster than GOTCHA.

5. CONCLUSIONS

This paper has presented a parallelised stereo region growing algorithm, focussing on task parallelism for GOTCHA. In the proposed method, multiple GOTCHAs are performed after partitioning a stereo image according to the estimated inter-boundary of initial seed points. Also, it has been designed to employ multiple ALSC operations simultaneously in each GOTCHA. The number of ALSCs is controlled intra-boundary of a seed point, which is estimated by the proposed confidence diffusion equation. The experimental results demonstrate the proposed MT-GOTCHA can achieve considerable speedup at reasonable cost of memory consumption.

ACKNOWLEDGEMENT

The research has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 218814 "PRoVisG"

REFERENCES

Aurenhammer, F., 1991. Voronoi diagrams - a survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23(3), pp. 345-405.

Fourtune, S., 1986. A sweepline algorithm for Voronoi diagrams. In: *Proceedings of the second annual symposium on Computational geometry*, pp. 313-322.

Gruen, A. W., 1985. Adaptive least squares correlation: a powerful image matching technique. *South African Journal of Photogrammetry, Remote Sensing and Cartography*, 14, pp. 175-185.

Holden, J., M. J. Zemerly, and J.-P. Muller 1993. *Parallel stereo and motion estimation*. John Wiley & Sons, Inc., New York, Amsterdam, pp. 175-232.

Kim, T. and J.-P. Muller, 1996. Automated urban area building extraction from high resolution stereo imagery. *Image and Vision Computing*, 14(2), pp. 115-130.

Kim, T., 2000. A Study on the epipolarity of linear pushbroom images. *Photogrammetric Engineering & Remote Sensing*, 66(8), pp. 961-966.

Ohta, Y. and Takeo Kanade, 1985. Stereo by intra- and inter-scanline search using dynamic programming. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 7(2), pp. 139-154.

Otto, G. P., 1988. Rectification of SPOT data for stereo image matching. In: *International Archives of Photogrammetry and Remote Sensing*, 27(B3), pp. 635-645.

Otto, G.P. and T.K. W. Chau, 1989. Region-growing algorithm for matching of terrain images. *Image and Vision Computing*, 7(2), pp. 83-94.

Scharstein, Daniel and Richard Szeliski, 2002. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1), pp.7-42.