Real-time Stereo-Image Stitching using GPU-based Belief Propagation

Michael Adam[1], Christoph Jung[1], Stefan Roth[2], Guido Brunnett[3]

[1]Fraunhofer-Institut für Graphische Datenverarbeitung IGD
[2]TU Darmstadt, Department of Computer Science
[3]Chemnitz University of Technology, Faculty of Computer Science
Email: {michael.adam, christoph.jung}@igd.fraunhofer.de
sroth@cs.tu-darmstadt.de
guido.brunnett@informatik.tu-chemnitz.de

Abstract

Image stitching or mosaicing is a challenging vision problem, especially when considering aspects like high definition content, real-time, and proper compensation of the parallax error of objects at different distances to the camera system. Today many approaches to image stitching exist, most of them deal with medium resolution images, offline processing, or are restricted to objects at similar distance. Our approach is based on calculation of disparities between corresponding points in stereo images by employing standard belief propagation. Instead of computing depth-maps like in previous approaches, we compute *stitch-maps* modifying cost functions in the underlying Markov random field model, which makes further projection steps dispensable and thereby overall computation more efficient. Our GPU-based implementation is real-time capable, allowing to stitch high definition images at constant frame rates. We show exemplary results from our system to demonstrate the quality of the merged images.

1 Introduction

Image stitching considers the problem of finding a suitable way of merging two or more images, depicting either the same or overlapping parts of a scene, into one single seamless view. Image stitching (or image mosaicing) techniques are required in many areas of application, for example in aviation photography [22], immersive telepresence or video conferencing [19], for creating cylindrical 360^o panoramas [6], or even spherical [20] video views of the environment.

The key challenge in image stitching is the displacement of objects in two different views of the same scene (parallax); moreover the displacement is different for objects at different depth levels (for cameras that do not share a common optical centre). Hence, while a simple image registration can work quite well in a single depth region of the image (for example for the distant background), where the parallax is small, there can be an obvious mismatch in other regions (for example the foreground regions), where the displacement of the objects is large. If now a simple rigid registration and blending is applied to the images, this effect is visible as "ghosting" of objects, due to the overlap of the mismatched regions. Most solutions to image stitching comprise (1) the problem of finding correspondences between the two input images, (2) solving some optimization problem to compensate for the parallax error, and (3) finding an appropriate transformation to merge the two images afterwards. For (2) especially the handling of multiple objects at different distances from the camera has to be addressed. In approaches that involve per pixel operations, the overall complexity is very sensitive to spatial resolution, especially for high definition (HD) content, which is the focus of our application. However, this aspect at the same time offers possibilities for parallelisation of sub processes, since the per-pixel operations can be parallelised efficiently, for example among GPU processing cores.

Our paper describes a novel approach to HD video image stitching, whereas we consider images acquired by a pair of digital cameras. We focus on the margin regions of HD stereo images for stitching, in order to achieve a nearly double-sized panorama. Consequently, processing is not required to be applied to the full image. The approach is related to that of [5], where standard belief propagation

in Markov random fields (MRF) is employed to calculate depth-maps in order to merge images to large panoramas. Compared to this and other previous approaches we are able to perform HD image stitching in real-time, as well as for larger image sizes. The novelty of our approach is a modification to the depth-map calculation, which, however, is still based on standard belief propagation. By directly computing *stitch-maps* our approach avoids the explicit computation of depth values, which makes additional projection steps dispensable and thereby improves the overall computation performance. Driven by our application in the area of camera arrays and telepresence, our solution particularly focuses on cases where near and far objects are assumed to appear close to each other in the image, which therefore requires a robust compensation of the objects' parallax error. Our implementation is realized on GPU hardware exploiting the nVIDIA CUDA Framework.

2 Related Work

Our approach to image stitching is based on Markov random fields (MRFs) and employs the widely used max-product belief propagation (BP) algorithm [9] for approximate inference. MRFs in combination with BP can be used to estimate variety local properties of a captured scene, including disparities between pairs of images [4]. Techniques for more efficient message calculations are introduced in [3], which are also exploited in our implementation. Beyond this improvement of efficiency, parallel hardware architectures like GPU can be used to increase processing speed [8, 16]. Even though it has been pointed out, that graph cut (GC) based approaches [17] may lead to lower-energy solutions than BP [13], graph cuts are not as parallelisable on the GPU as BP [18], which makes it inappropriate for our processing platform. Recently continuous formulations of the problem [27] demonstrated remarkable improvements on GPU in both computation time and memory consumption compared to GC frameworks. Since depth information could be calculated by belief propagation based techniques, the usage of this information for stitching has been investigated in [23, 5]. Both approaches are related to ours, but require additional projection steps for image merging. Unlike the GPU-based implementation in [5], our approach (including image pre-

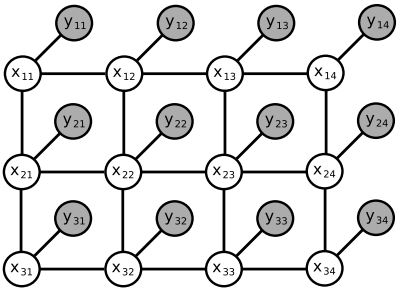

Figure 1: Example of a Markov Random Field with hidden (white) and observable (grey) nodes.

processing) is fully implemented using the nVidia CUDA Framework.

Early approaches to panoramic video stitching have been described for example in [26], already providing real-time operation for low-resolution images. Later publications propose methodologies for higher resolutions, based on motion estimation [24, 25]. Recent approaches also focus low-cost webcams [7], where an adaptive blending function is used to reduce ghosting effects. Since no local alignment is performed in the overlapping region, objects moving from one image to the other can produce clearly visible "jump" effects, and large objects in the foreground show recognizable stitching artefacts.

3 Our Approach

In our approach we use MRFs combined with BP for depth-map calculation [4], and extended the basic approach to more efficiently handle the conditions in image stitching applications. Furthermore the implementation is realized on a GPU subsystem (based on nVIDIA CUDA) in order to meet the real-time requirements even for HD video content at constant frame rates. In the following we give a short overview of the basic approach and derive the corresponding modifications. There are a variety of approaches for using stereo image pairs for calculating depth maps, which encode scene depth by providing disparity values between corresponding scene-points in both input images. As we will point out later, the direct usage of this disparity information for stitching can be problematic. Our approach requires two input images: the left I_l and the

right one I_r, both of width w_I pixels. The images are depicting the same scene, whereas parts of the scene (objects) appear in both images. The region where the images contain the same part of the scene is called overlapping region, which is of width w pixels. The width of the overlapping region is chosen, so that the nearest objects appearing in both images, which have to be stitched correctly, fit together.

To generate one large merged image from the two input images, a joint region I_s for the overlapping region has to be created, which in the end removes the visual characteristics of the parallax errors and replaces the original overlapping image regions of I_l and I_r. Therefore the joint region is chosen to have the same width w as the overlapping region.

For calculation of the joint region a Markov Random Field (MRF) is defined (see Figure 1 for a generic example). A MRF is an undirected graph $G = (V, E)$, where the nodes in V represent random variables and the edges in E represent the factorization structure, for example among pairs of these variables. The nodes V can be divided in two sets: an observable node Y, which represents the observed random variables, here the stereo image pair, and hidden nodes $X = \{x_i\}$, which represent the hidden quantities x_i of the scene. In the discrete case, each node in X can take one of L possible values, so called labels. In the case of depth estimation these labels refer to disparities.

For every pixel within the joint region there is a hidden node. These nodes are connected to the observations by an edge $(x_i, y) \in E$. The hidden nodes themselves are connected in a lattice way: $(x_i, x_j) \in E$ if x_i and x_j are non-diagonal neighbours.

The unary potential function $\Phi(x_i, y)$ denotes the compatibility of a labelling x_i with the observation y. Between neighbouring nodes x_i and x_j within the MRF the compatibility of their labelling is modelled by the pairwise potential $\Psi(x_i, x_j)$.

The posterior distribution of the labeling can then be written as:

$$P(X|Y) = \frac{1}{Z(Y)} \prod_{(x_i,x_j)\in E} \Psi(x_i, x_j) \prod_{i \in V} \Phi(x_i, y) \quad (1)$$

where $Z(Y)$ is a normalisation constant.

To improve numerical stability we have applied a logarithmic transformation [17]: $D_i(f) \propto -ln(\Phi_i(f, y))$ and $U_{ij}(f, g) \propto -ln(\Psi_{ij}(f, g))$. The unary potential Φ_i becomes the data cost D_i and the pairwise potential Ψ_{ij} becomes the discontinuity costs U_{ij}. With these definitions the following energy function can be derived:

$$E(\mathbf{F}) = \sum_{(x_i,x_j)\in E} U_{ij}(f_i, f_j) + \sum_{i \in V} D_i(f_i) \quad (2)$$

where $\mathbf{F}$ is a labelling for all nodes and f_i is the label of node x_i.

We chose the discontinuity cost function as being truncated linear because of the possibility of calculation the associated messages in linear time [10], which results in a reduction of overall computation time in comparison to the quadratic execution time of a generic cost function:

$$U_{ij}(f, g) = min(\alpha_U \ |f - g|, \tau_U) \quad (3)$$

The data cost function is defined as truncated quadratic:

$$D_i(f) = min(\alpha_D \ s_i(f)2, \tau_D), \quad (4)$$

where α_U, α_D are scaling values, τ_U, τ_D truncation thresholds and $s_i(f)$ is a weighting function for compatibility of the images at node x_i. For depth estimation for example, the data cost weighting function s_i is usually defined as:

$$s_i(f) = C\left(I_l\left(x, y\right), I_r\left(x + f, y\right)\right) \quad (5)$$

where (x, y) is the position of node x_i within the MRF and C a function for comparing pixel values. In our case, a colour matching approach is used. The cost function in depth estimation (5) leads to a disparity-map where for each pixel in the left image the distance to the corresponding location in the right image is determined. If these depth values are subsequently used during stitching to transform the left image, a good transition between the images is achieved only at left edge, but the right one gets cleaved, and vice versa. This cleaving can be prevented by scaling the disparity values by one to zero from the left to the right edge of the overlapping area. This leads to acceptable transitions at both edges, but at the same time stitching artefacts appear in the overlapping area, due to accessing false corresponding pixels in the both images. To obviate these problems the data cost function can be adapted. A cost function that is more appropriate for stitching calculates a disparity-map containing disparities from the left to the right image on

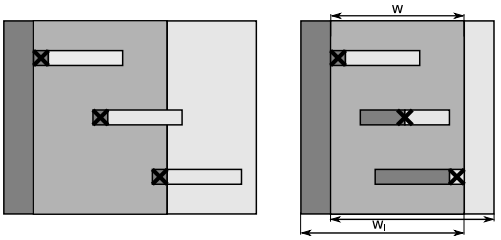

Figure 2: Illustration of searching for corresponding pixels in depth estimation (left image) and stitching (right image). Search in stitching considers disparities both relative to the right and the left image. The left/right camera image is highlighted in dark/bright grey, the overlapping region in grey.

the left edge of the overlapping area and disparities from the right to the left image on the right edge of the disparity-map. In the region between these two edges an appropriate transition has to be defined. To distinguish these modified disparity-maps from depth-maps, we call them *stitch-maps* because they contain values customized to disparity-based stitching.

Taking the above mentioned aspects into account, we define the data cost weighting function s_i as:

$$s_i(f) = C\left(I_l\left(w_I - w + x - f\frac{x}{w}, y\right),\right. \\ \left. I_r\left(x + f\frac{w-x}{w}, y\right)\right) \tag{6}$$

Instead of selecting one pixel in the left image and search for the corresponding pixel in the right image, we select the node within the MRF and search, depending on the position of this node, in both images (see figure 2). On the left edge of the overlapping region the value of x is 0. Therefore the search performed is equivalent to depth estimation at the leftmost nodes. On the other side of the overlapping area ($x = w$) the term $w - x$ of the x-position in I_r becomes 0. The search is then performed within the left image by comparison to static pixels in the right image.

Taking these disparities $B(x, y)$, for a given position (x, y) within a stitching area of width w, the horizontal displacement of the corresponding pixels in the images is then defined as:

$$Disp_l(x, y) = x - B(x, y)\frac{x}{w} \tag{7}$$

for the left image and

$$Disp_r(x, y) = B(x, y)\frac{w - x}{w}. \tag{8}$$

This definition of the displacements enables the construction of the joint region I_s of the parts of the left and right image that do not overlap:

$$I_s(x, y) = \\ \left(1 - \frac{x}{w}\right) I_l\left(x + w_I - w + Disp_l(x, y), y\right) \\ + \frac{x}{w} I_r\left(x + Disp_r(x, y), y\right) \tag{9}$$

The output image is then assembled from the unmodified parts of I_l, I_r, and the joint region:

$$O(x, y) = \begin{cases} I_l(x, y) & \mid x \in r_1 \\ I_s(x - (w_I - w), y) & \mid x \in r_2 \\ I_r(x - (w_I - w), y) & \mid x \in r_3 \end{cases} \tag{10}$$

where $r_1 = [0, ..., w_I - w)$, $r_2 = [w_I - w, ..., w_I]$, $r_3 = (w_I, ..., 2w_I - w]$ and (x, y) are the coordinates of a pixel in the output image.

4 Implementation

Our test setup consists of two DALSA Pantera SA 2M30 cameras mounted on a tripod (see Figure 3). The alignment is chosen to represent the later setup in a circular camera array configuration, with an angle of $45°$. With these constraints the width of the overlapping region is limited to 20 percent of the full image width of 1600 pixels.

The processing pipeline of our real-time stitching approach is depicted in Figure 4. Once the camera image pair is captured and accessible in CPU-RAM, the pictures are transferred to the graphics card memory. The first step on the GPU is the demosaicing of the Bayer-coded images to obtain the corresponding colour images. To compensate for the vignetting effect, a brightness correction is applied. These images are then projected to a common coordinate system (corresponding to rectification in depth estimation) using a cylindric projection. This projection transforms the images in order to make

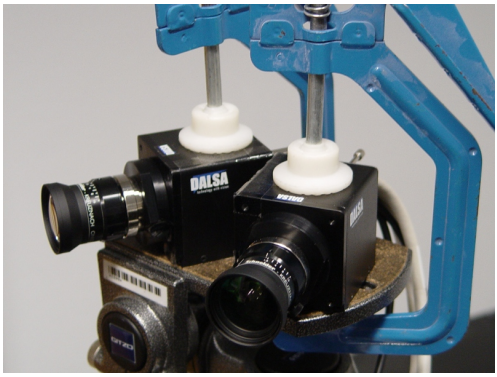

Figure 3: Camera setup: Two cameras fixed on a tripod, representing alignment in a circular camera array.

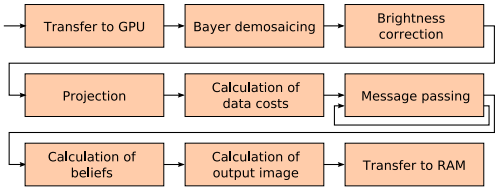

Figure 4: Pipeline of our stitching approach.

the epipolar lines nearly horizontal in the overlapping region. In the data cost calculation step, the images are scaled down to remove unnecessary data extrapolated in Bayer demosaicing and to reduce the overall memory usage. Based on this downscaled data, the data costs (see expression 4) are calculated. The costs are used for message passing in a multi-scale belief propagation scheme [3]. After a certain number of iterations message passing is stopped and the disparities are calculated. Now the stitched image can be directly calculated from the projected, brightness-corrected, normal-scaled colour images. Finally, the stitched image is transferred back to CPU-RAM.

The stitching assumes a global alignment of the image pair. For our approach, we use a visual calibration to obtain camera-related parameters like cylindrical projection factor, size of the overlapping region, and displacement. This is necessary because standard image registration approaches turned out to be inapplicable due to the small image overlap.

All steps of our stitching pipeline (Figure 4), except for the transfer operations, are implemented as CUDA kernels based on version 2.0 of the nVIDIA CUDA SDK [12]. The kernel execution is configured, so that a kernel instance is created for every output element. Beyond the usage of CUDA, we use common algorithmic improvements to speed up the calculations [3]. These techniques are the above-mentioned linearisation of the message calculation [10], a multi-scale approach for fast convergence [11], and a chequerboard-style [3] message update algorithm. Due to the nature of the message passing algorithm, the performance of the implementation is mainly limited by the GPU platform. Since cached memory access is optimized for local operations, global access as it is required for calculation is inefficient. The messages for each direction and the data-costs are stored in a one-dimensional array each. The access is performed using by texture-fetch operations to exploit the built-in cache.

5 Results

Our evaluations mainly concentrate on the processing speed of the image stitching and the visual results, as the application constraints involve real-time operation on HD video content. All tests are performed on a single Opteron 2216 CPU core (Windows XP) at 2.4 GHz and a GeForce GTX 260 with 192 processing cores. The main part of the processing is carried out on the GPU, whereas the CPU is only occupied by image acquisition, transfer to GPU memory and visualisation.

As a reference, we provide performance measurements of the BP implementation for calculation the depth-map of the Tsukuba image pair from [1]. Since the processing time is independent from the content of the images, but depends on the image size, number of iterations and choice of cost/compatibility functions, the measurement can be considered representative for other image pairs with the same parameters. Table 1 shows, that the performance of BP is massively improved by exploiting GPU hardware. The CUDA-based BP is about 40-50 times faster than the not optimised, single core CPU based implementation [3]. Our tests indicated that 40 iterations for single scale and 4 iterations per scale for the multi-scale approach lead to acceptable convergence and therefore good

visual results. Choosing a maximum disparity of 96 pixels (corresponding to the same number of labels), we are able to create stitched panoramas with barely noticeable stretching artefacts and concurrent avoidance of ghosting and blurring artefacts.

Approach	Execution Time
CPU	71881 ms
CPU, ms	9663 ms
GPU	1374 ms
GPU, ms	236 ms
GPU, ms, lmc	45 ms

Table 1: Execution time of BP depth estimation for Tsukuba image-pair (384x288, maximum disparity: 16). ms stands for an implementation of a multi-scale approach of BP, and lmc denotes the algorithm based on message calculation in linear time.

Since there are to our knowledge neither data sets nor standards for evaluation of the quality of stitching available, we decided to evaluate our results by visual comparison to a linear blending and a commercially available stitching application. In contrast to other previous related publications, for example [23], which use Autostitch for evaluation, we decided to use PanoramaStudio [2] because of its capability to adjust the projection parameters and perform a fair comparison to our results. To exclude the effect of global misalignment, we use the well known stereo datasets for depth estimation [1] for evaluation of our stitching, which provide already aligned views. We crop these images at the right side of the left image and on the left side of the right image, in order to reduce the size of the overlapping region. Figure 5 shows sections of stitching results using stitch-maps in comparison to a linear blending approach and the results of PanoramaStudio (using optimal parameters). Figure 9 shows a comparison of PanoramaStudio and stitch maps on a different, real-world image pair.

As is apparent, stitching improves the visual quality in comparison to blending. While both stitching approaches provide reasonable visual quality, stitching using stitch-maps increases the quality in areas with large disparity jumps, observable in the area around the cone in front of Figure 5b, or the arm in Figure 9-2. The parameters for stitch-map generation of the stereo datasets can be chosen similar to the parameters of depth estimation. While using a truncated quadratic cost function for the data costs and a truncated linear for the discontinuity costs, the truncation value for the data costs τ_D is set to 1.7, the scaling factor α_D is set to 0.07 and the truncation value for the discontinuity costs τ_U is 4.0 ($\alpha_U = 1$). The parameters for the real camera setup differ with respect to sensor noise and calibration mismatches: τ_U remains at 4.0, it further turned out that a α_D of 0.0002 and a τ_D of 0.8 provide best results.

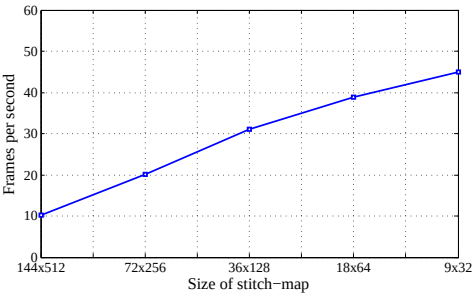

Figure 7: Frames per second for different stitch-map resolutions.

To achieve high frame rates, we also investigated the use of low resolution stitch-maps. This can be done by directly using the intermediate results of multi-scale BP and interpolating bi-linearly (Figure 6). We came to the conclusion that low resolution stitch-maps lead to a good trade-off between visual quality and high frame rates (Figure 7). By using a low resolution stitch-map (18x64) we are able to stitch images at 37 frames per second, merging two images of 1600x1200 pixels to one image of size 2832x1200 pixels. We also achieve good results for objects at different distances from the camera and an improvement of the overall image quality, as depicted in Figure 8.

6 Conclusion

In this paper we presented a novel approach to local alignment of images for real-time video stitching applications, with the main focus on merging image margin regions in order to create large HD panoramas. We used disparity information calculated by an adapted MRF model and efficient belief propagation. This technique provides a high stitching quality because of its pixel-wise alignment

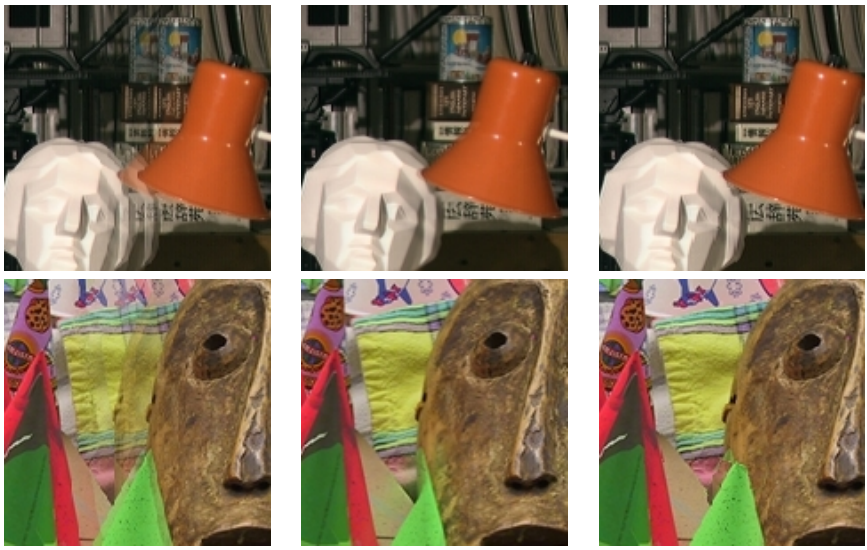

Figure 5: Comparison of blending (column 1), stitching with PanoramaStudio (column 2) and stitching with our approach (column 3): blending produces ghosting artefacts and PanoramaStudio stitching artefacts at the cone in b2. Our reduces these artefacts massively.

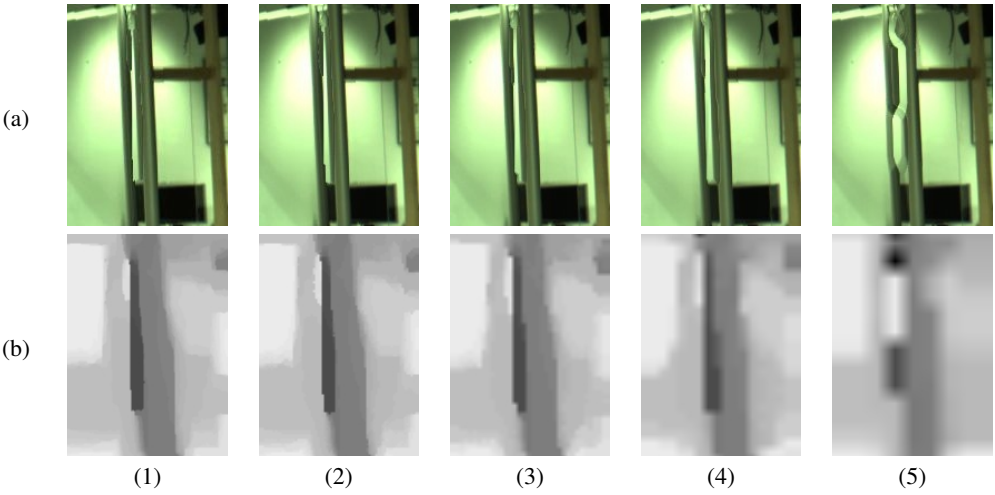

Figure 6: Section of stitching (upper row a) using low-resolution stitch-maps (lower row b). Resolutions are: 144x512 (1), 72x256 (2), 36x128 (3), 18x64 (4) and 9x32 (5).

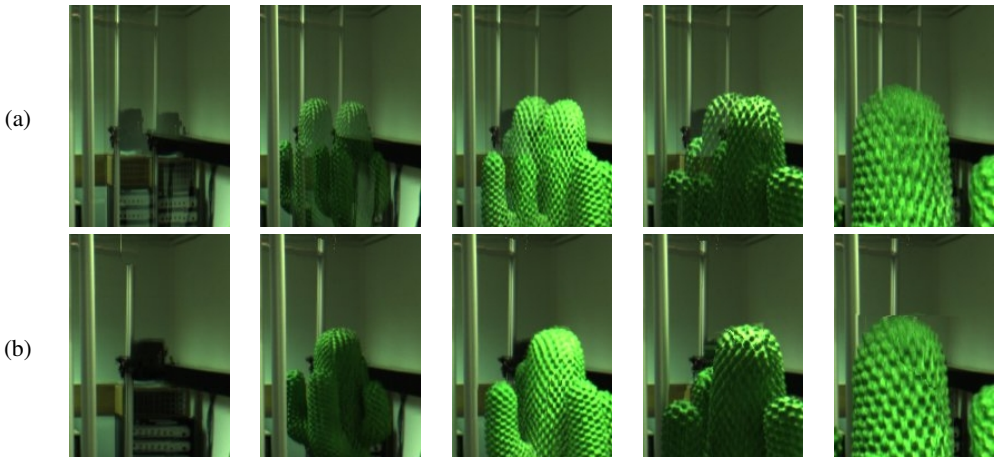

Figure 8: Sequence of merged images (detail) depicting objects at different distances, using blending (upper row a) and stitching (lower row b): the stitching produces good results for objects at different distances within the same image (see green cactus in the foreground and grey poles in the background).

of the images and implicit handling of large disparity jumps between neighbouring pixels. Furthermore the approach introduced the novel principle of *stitch-maps*, which makes subsequent projection steps dispensable and thereby improves performance. Several example images showed the appropriateness and quality of our approach. The algorithm is executed in real-time on standard PC hardware by exploiting the parallelism of a GPU platform.

Our future work will mainly lead into two directions. First we aim to improve our methodology for real-time disparity estimation. Here the main focus is on the quality of the computed stitch-maps, where for example approaches based on pre-segmentation and plane fitting can improve the results [28]. Furthermore it is planned to exploit the image stitching system for more than two cameras in circular camera arrays.

7 Acknowledgements

This work was funded by the European project hArtes under the Seventh Framework Programme of the European Commission.

References

[1] Daniel Scharstein and Richard Szeliski. High-Accuracy Stereo Depth Maps Using Structured Light. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on,* 195, 2003.

[2] PanoramaStudio. http://www.tshsoft.de/en/panoramastudio/index.html.

[3] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient belief propagation for early vision. *In CVPR*, 261-268, 2004.

[4] Jian Sun, Nan-Ning Zheng and Heung-Yeung Shum. Stereo Matching Using Belief Propagation. *IEEE Trans. Pattern Anal. Mach. Intell*, 787-800, 2003.

[5] Alan Brunton and Chang Shu. Belief Propagation for Panorama Generation. *3D Data Processing Visualization and Transmission, International Symposium on*, 885-892, 2006.

[6] Richard Szeliski and Heung-Yeung Shum. Creating full view panoramic image mosaics and environment maps. *SIGGRAPH '97: Proceedings of the 24th annual conference on computer graphics and interactive techniques*, 251-258, 1997.

[7] Mai Zheng, Xiaolin Chen and Li Guo. Stitching Video from Webcams. *ISVC '08: Proceed-*

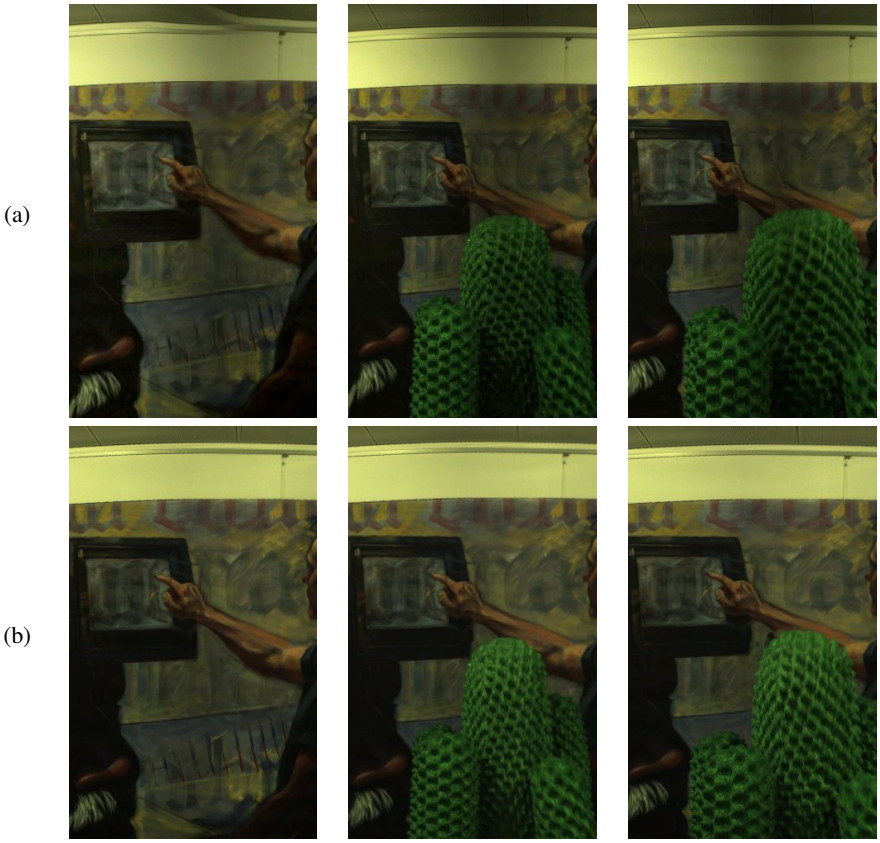

Figure 9: Comparison of stitching using PanoramaStudio (row a) and stitching using stitch maps (row b) without an object (1) and with an object in different distances (2,3): PanoramaStudio produces noticeable artifacts (edge of ceiling in (1) and the arm for example in (2) and (3)) while the stitching using stitch maps achieves a plausible output image.

ings of the 4th International Symposium on Advances in Visual Computing, Part II, 420-429, 2008.

[8] Alan Brunton, Chang Shu and Gerhard Roth. Belief Propagation on the GPU for Stereo Vision. *Computer and Robot Vision, Canadian Conference*, 76, 2006.

[9] Jonathan S. Yedidia, William T. Freeman and Yair Weiss. Understanding belief propagation and its generalizations. *Exploring artificial intelligence in the new millennium*, 239-269, 2003.

[10] G. Borgefors. Distance Transformations in Digital Images. *Computer Vision, Graphics, and Image Processing*, 344-371, 1986.

[11] E. H. Adelson, C. H. Anderson, J. R. Bergen, P.J. Burt and J.M. Ogden. Pyramid methods in image processing. *RCA Engineer*, 33-41, 1984.

[12] NVIDIA. CUDA Programming Guide 2.0. 2008.

[13] Marshall F. Tappen and William T. Freeman. Comparison of graph cuts with belief propagation for stereo, using identical mrf parameters. *In ICCV*, 900-907, 2003.

[14] William T. Freeman, Egon C. Pasztor and Owen T. Carmichael Y. Learning low-level vision. *International Journal of Computer Vision*, 2000.

[15] Yair Weiss. Interpreting images by propagating Bayesian beliefs. *Advances in Neural Information Processing Systems 9*, 908-915, 1997.

[16] S. Grauer-Gray, C. Kambhamettu and K. Palaniappan. GPU Implementation of Belief Propagation Using CUDA for Cloud Tracking and Reconstruction. *5th IAPR Workshop on Pattern Recognition in Remote Sensing*, 2008.

[17] Y. Boykov, O. Veksler and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1222-1239, 2001.

[18] A. Bhusnurmath and C.J. Taylor. Graph Cuts via l_1 Norm Minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1866-1871, 2008.

[19] G. Heinrich, M. Leitner, C. Jung, F. Logemann and V. Hahn. A platform for audiovisual telepresence using model- and data-based wave-field synthesis. *125th AES Convention San Francisco*, 2008.

[20] (James) Elliott Coleshill, Alexander Ferworn. Panoramic Spherical Video - The Space Ball. *ICCSA*, 2003.

[21] Johannes Kopf, Matt Uyttendaele, Oliver Deussen and Michael F. Cohen. Capturing and Viewing Gigapixel Images. *ACM Transactions on Graphics 26*, 2007.

[22] J. Majumdar, S. Vinay and S. Selvi. Registration and mosaicing for images obtained from UAV. *International Conference on Signal Processing and Communications*, 198-203, 2004.

[23] Zhi Qi and J.R. Cooperstock. Depth-based image mosaicing for both static and dynamic scenes. *International Conference on Pattern Recognition*, 1-4, 2008.

[24] Masakatsu Kourogi, Takeshi Kurata, Junichi Hoshino and Yoichi Muraoka. Real-time Image Mosaicing from a Video Sequence. *International Conference on Image Processing*, 1999.

[25] Tomokazu Satoa, Sei Ikeda, Masayuki Kanbaraa, Akihiko Iketani, Noboru Nakajima, Naokazu Yokoya and Keiji Yamada. High-resolution Video Mosaicing for Documents and Photos by Estimating Camera Motion. *Computational imaging. Conference No 2*, 2004.

[26] Ding-Yun Chen, Murphy Chien-Chang Ho, Ming Ouhyoung. VideoVR: A Real-Time System for Automatically Constructing Panoramic Images from Video Clips. *CAPTECH '98. International workshop*, 1998.

[27] T. Pock, T. Schoenemann, G. Graber, H. Bischof and D. Cremers. A Convex Formulation of Continuous Multi-Label Problems. *European Conference on Computer Vision (ECCV)*, 2008.

[28] A. Klaus, M. Sormann and K. Karner Segment-Based Stereo Matching Using Belief Propagation and a Self-Adapting Dissimilarity Measure *International Conference on Pattern Recognition (ICPR)*, 2006.