PANORAMIC VIEWFINDER: PROVIDING A REAL-TIME PREVIEW TO HELP USERS AVOID FLAWS IN PANORAMIC PICTURES

Patrick Baudisch, Desney Tan, Drew Steedly, Eric Rudolph,
Matt Uyttendaele, Chris Pal, and Richard Szeliski
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
{baudisch, desney, steedly, erudolph, mattu, szeliski}@microsoft.com, pal@cs.umass.edu

ABSTRACT

Image stitching allows users to combine multiple regular-sized photographs into a single wide-angle picture, often referred to as a panoramic picture. To create such a panoramic picture, users traditionally first take all the photographs, then upload them to a PC and stitch. During stitching, however, users often discover that the produced panorama contains artifacts or is incomplete. Fixing these flaws requires retaking individual images, which is often difficult by this time. In this paper, we present Panoramic Viewfinder, an interactive system for panorama construction that offers a real-time preview of the panorama while shooting. As the user swipes the camera across the scene, each photo is immediately added to the preview. By making ghosting and stitching failures apparent, the system allows users to immediately retake necessary images. The system also provides a preview of the cropped panorama. When this preview includes all desired scene elements, users know that the panorama will be complete. Unlike earlier work in the field of real-time stitching, this paper focuses on the user interface aspects of real-time stitching. We describe our prototype, individual shooting modes, and an implementation overview.

KEYWORDS: *Panorama, user interface, interactive, stitching, real-time, preview.*

1. INTRODUCTION

To allow digital camera users to take pictures with a viewing angle wider than the one supported by the built-in lens, researchers have proposed techniques for *stitching* multiple photographs or a stream of video (Teodosio and Bender, 1993; Irani and Anandan, 1998) into a single contiguous panoramic picture, or *panorama* (Mann and Picard, 1994; Szeliski, 1996). Recent full-view stitching methods even allow automatic stitching of photos taken in arbitrary order and spatial arrangement (Szeliski and Shum, 1997; Bergen et al., 1992]. While some cameras offer simple *stitch assist* modes (see related work section), stitching is still generally performed as a post-hoc step to picture taking.

Making flawless panoramas using a post-hoc stitcher can be challenging. We conducted an informal survey on an internal company mailing list for stitching users and received 26 responses out of 163 mailing list members. Each respondent had taken between 4 and 180 panoramas (median 20). All except one participant who had shot all his panoramas using a tripod had discovered at least one of the following flaws when stitching their pictures:

- *Ghosting (88% respondents)*: Objects that move between frames end up appearing translucent when the stitcher blends frames together. Since cameras typically show only one picture at a time, ghosting can be hard to detect.
- *Missing content (65% respondents):* Users may miss relevant areas or relevant areas may disappear when cropping the panorama to its final rectangular shape, as illustrated in Figure 1. Even if users are aware of the danger, the difficulty of understanding perspective projections (Edwards, 1999) in com-

bination with the particular geometry of panoramas (cylindrical or spherical projection) can make it hard for users to estimate what space needs to be covered.

- *Stitching failed (38% respondents):* The stitcher was unable to connect one or more frames to the rest of the panorama. Reasons include lack of overlap between pictures, lack of the texture, or flaws in the pictures, such as poor focus or motion blur.

Figure 1:Common mishap when taking panoramas: (a) The user is trying to cover the Seattle Space Needle, but does not cover enough of the sky around it. During cropping, the user has the choice between (b) including white fringes and (c) cropping relevant content.

Fixing these flaws requires retaking images, but when users discover the errors during stitching, it is often too late.

While other researchers have looked at real-time previews generated using real-time stitching from a technical perspective (e.g. Peleg and Herman, 1997), we address the above issues by providing users with a novel user interface that allows users to utilize these previews in order to verify which areas of the scene have been successfully covered. The user interface of our system, called Panoramic-Viewfinder is designed to match the interaction model of existing digital cameras. In fact, we have constrained our designs to depend only a mode selector and a shutter button so that it may be ported to utilize existing camera interaction mechanisms.

We begin with a walkthrough of the system. We then go over related work and describe the implementation of our prototype. We conclude with a summary of our findings and a description of future work.

2. PANORAMIC VIEWFINDER

Panoramic Viewfinder is an interactive system for panorama construction that provides users with a real-time stitched preview of the panorama *while shooting*.

Figure 2 shows the user interface of our prototype system, here running on a Sony U50 device with an attached webcam. Panoramic Viewfinder offers three main user interface elements designed to help complete the desired panorama, i.e., (1) *the preview* shows the panorama in its current state of completion, (2) *the viewfinder* shows what the camera sees at this moment, integrated at the correct location into the panorama, and (3) *the real-time cropping frame* shows the extent the panorama will have *after cropping* if the user stopped shooting at this instant.

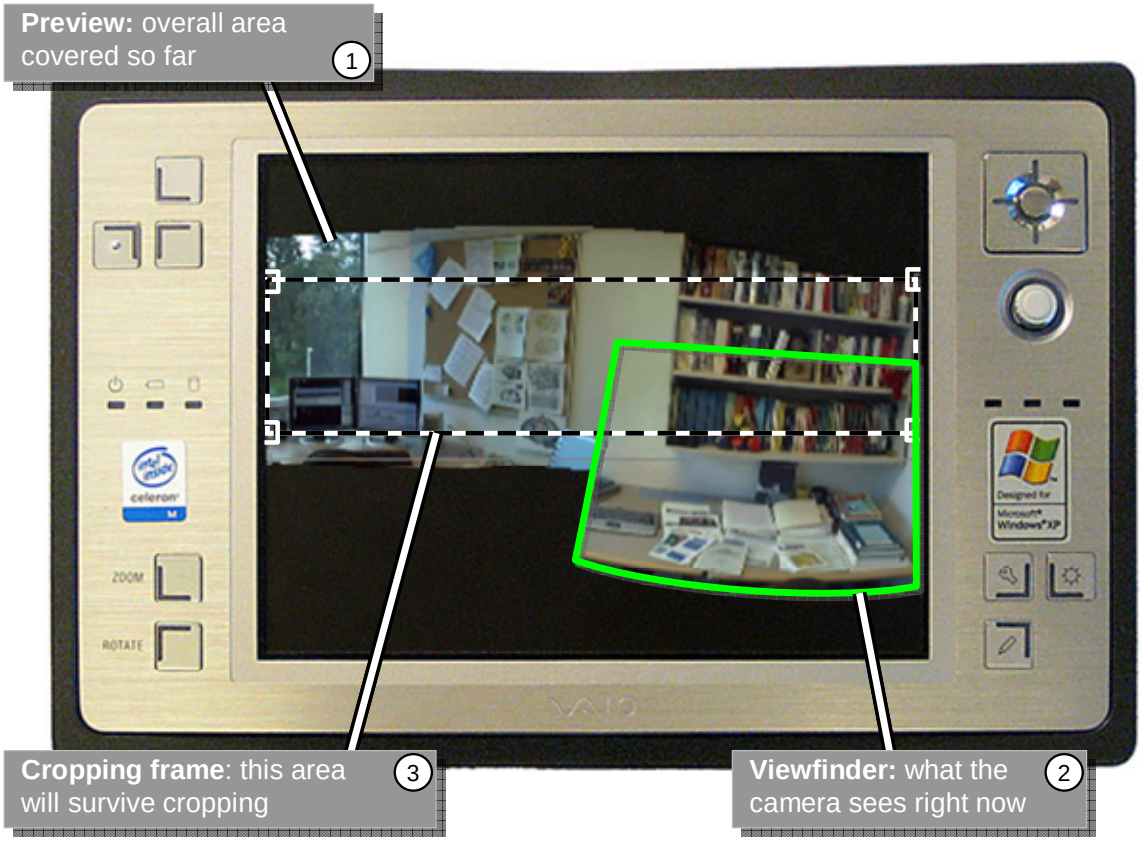

Figure 2: The Panoramic Viewfinder user interface on a Sony U50 ultra portable PC

2.1. Walkthrough

Figure 3 illustrates the user's interaction with the Panoramic Viewfinder using an example scenario. The user's task is to take a panoramic picture of the front of a building complex[1]. This example features our system's *manual* mode, in which the user manually triggers each photo. We explain the automated modes later in the paper.

(a) The camera is ready, but the user has not triggered the shutter button. The LCD shows what the camera sees, just as in any other camera.

[1] For a demo video of our system see http://www.patrickbaudisch.com/projects/panoramicviewfinder

(b) The user has pressed the shutter button and the camera has taken the first picture. From now on, the viewfinder is displayed in its relative location with respect to the panorama. This is made possible by continuously matching the viewfinder contents with the panorama (real-time stitching, see section "Implementation").

(c) The user pans the camera and takes additional pictures. Since these are automatically added to the panorama, the panorama grows. Whenever necessary, the panorama preview is automatically scaled to fit the screen. The cropping frame is also updated continuously. It always represents the largest rectangle that can be inscribed into the current panorama. Before stopping, the user checks the cropping frame and sees that stopping now would cause part of the building to be removed during cropping!

(d) The user takes an additional shot of the top left corner. This completes the bounding box and the cropping frame now contains the entire building complex.

(e) When uploading the photo to the PC, the panorama is restitched using our high-quality offline stitcher (Brown et al., 2005) to obtain maximum image quality. When cropping the panorama to rectangular shape now, all desired content is presented. The user already knew that this would be successful, because the cropping frame had indicated that all the required material had been taken. While we expect that most users will choose to keep the default cropping frame, as displayed in the preview, we do permit users to modify the final cropping if they so desire. In this way, they can perform non-standard crops (e.g. non-rectangular). Whatever the intent, they know from the preview that they have captured the desired material for it.

In summary, Panoramic Viewfinder allows users to take pictures by first shooting the desired scene elements, then filling the bounding box around these elements. The key element in this process is the real-time cropping frame, which serves as a visual reference. The real-time cropping frame inverts the traditional cropping model: Traditional post-hoc cropping requires users to *narrow* the image down to what is desired; using the real-time cropping frame users keep adding content until the cropping frame has *grown* to cover the desired area.

This approach has two main benefits. First, it avoids the need for providing the cropping frame with explicit controls, such as resize handles. This supports our goal of creating an interaction model that will transfer seamlessly into a standard consumer camera; operating a set of resize handles using the set of built-in buttons available on such cameras can be inefficient and frustrating. Second, growing the cropping frame rather than narrowing it down increases users' efficiency by helping them avoid taking undesired content in the first place.

While the default action during upload is to crop each picture automatically as previewed using the real-time cropping frame, users can always override that choice by accessing the uncropped panorama and cropping it manually with resize handles. This works well in the offline mode because users have access to a mouse rather than having to use the camera as their input device.

In addition to helping users take care of missing content, Panoramic Viewfinder can also help users detect ghosting: ghosting typically becomes visible as a disconnect between overlapping frames. And finally, Panoramic Viewfinder provides support for the third problem, i.e., failure to stitch, by immediately notifying users whenever such a failure occurs. We describe this is the following section.

2.2. Dealing with stitching failure

In order to place the viewfinder into the panorama preview, the system matches every new image received from the camera with the preview. In some situations, for example if the user moved to camera so quickly that the new frame cannot be matched, the system is unable to properly place the received image. If this happens, the viewfinder continues to update, but remains stationary and informs the user of the situation by changing the viewfinder frame from green to red and by playing a continuous error sound.

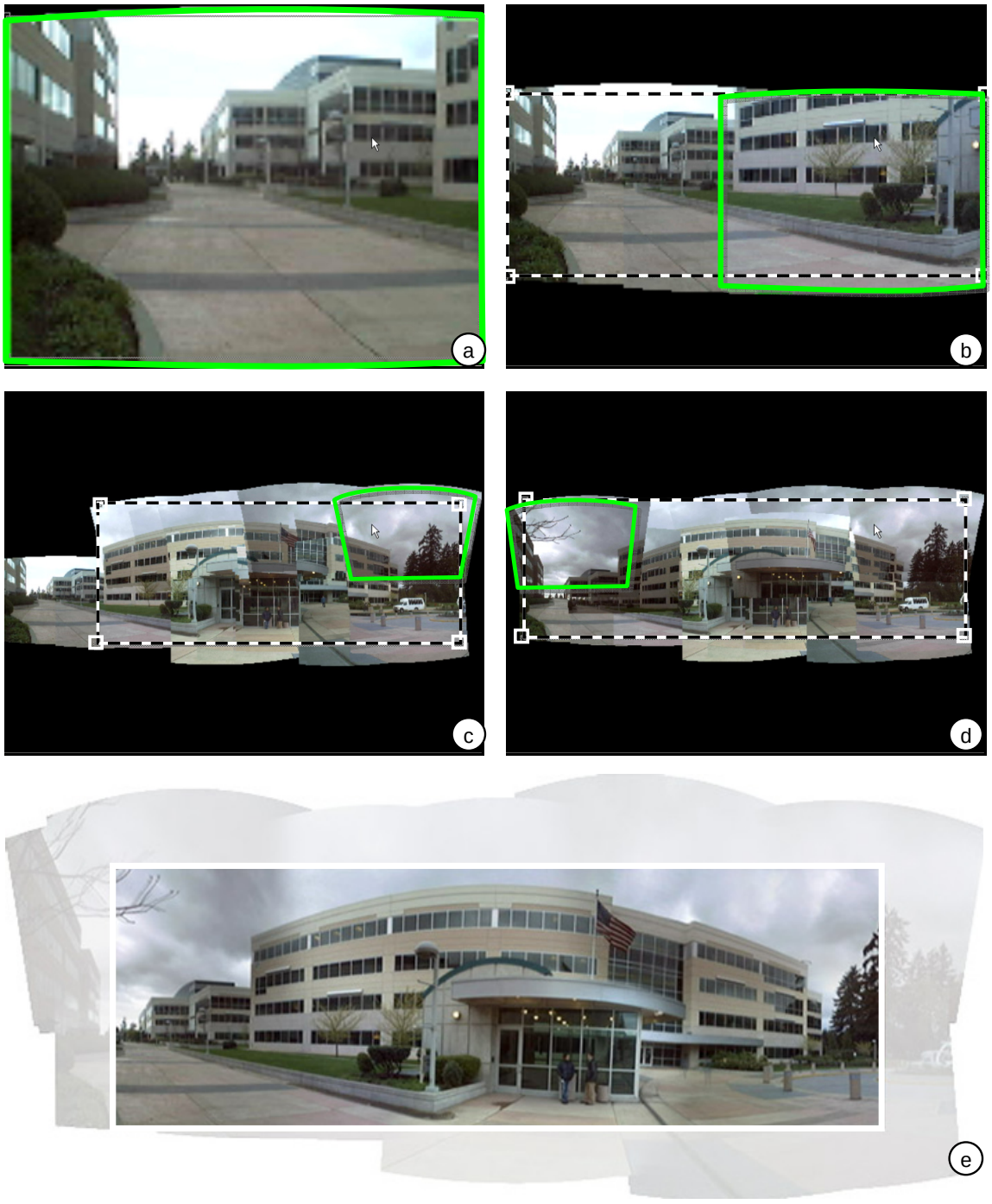

Figure 3: With Panoramic Viewfinder, panning the camera corresponds to a painting operation with a textured rectangular (actual screen captures with viewfinder and cropping frame emphasized to improve readability on b/w printouts)

To fix the problem, users may pan the camera and take an additional shot that offers greater overlap with the existing panoramic image. Alternatively, users may choose to zoom out to cover more of the image, and then zoom back in to fill in with higher resolution. As soon as the next match is found the alert ends and the viewfinder continues to track within the preview.

In addition to alerting users, Panoramic Viewfinder also provides a mechanism for reducing stitching errors in the first place. We first experimented with a *video* mode, i.e., adding each and every frame to the panorama. The implementation of such a video mode was simple, as every new frame is stitched tempo-

rarily anyway in order to properly place the viewfinder. While video mode eliminated most stitching errors, the large number of stored frames affected the frame rate and increased calibration error (Sawney et al., 1998). We therefore created an *automatic* mode instead. In this mode, the system does not capture frames until the stitcher loses track. When it does, the system automatically adds the last frame that it was able to successfully stitch. It then retries stitching the current frame which will typically now work, due to the extra frame in between. Compared to video mode, automatic mode creates less data, is therefore faster, and leads to less accumulated calibration error.

3. RELATED WORK

Panoramic Viewfinder touches three areas of related work: stitching, user interfaces designed to support stitching, and user interface research on small screen devices and interaction.

We have already mentioned some approaches to stitching in the introduction. Panoramic Viewfinder uses a feature-based approach (Brown et al., 1992; Brown et al., 2003). Unlike image-based techniques (Peleg and Herman, 1997; Szeliski 1996; Szeliski and Shum 1997) that attempt to match bitmaps, feature-based techniques reduce the complexity of the matching problem by using only a few salient image features, such as corners. These algorithms tend to be more robust to scene motion and parallax induced by camera translations.

The concept of real-time stitching was first explored by the *VideoBrush* system (Peleg and Herman 1997; Sawhney et al., 1998; Sawhney, Hsu, and Kumar, 1998). While VideoBrush offered most of the features of the real-time stitcher underlying Panoramic Viewfinder, it did not offer any particular user interface beyond displaying images on a computer screen. In this paper, we propose a user interface concept to be added *on top of* real-time stitchers, such as VideoBrush. Novel features include scaling preview, real-time cropping frame, warning when losing track, and the automatic mode.

Some digital consumer cameras, such as the Canon Powershot S230 (www.powershot.com), offer a *Stitch Assist* mode. To help users line up frames and to obtain the right amount of overlap between frames, the LCD shows the current viewfinder next to part of the last frame taken (Figure 4a). In contrast, Panoramic Viewfinder offers users more freedom by allowing them to take content in random order. This also simplifies adding post-hoc content.

In order to test the validity of the Stitch Assist concept for large panoramas we implemented a 2D version of Stitch Assist (Figure 4b). By showing cropped previous frames in all four directions it allows users to add pictures in any of the four main directions. While we found this to work well for smaller panoramas, it does not provide support for filling in the interior of larger 2D panoramas.

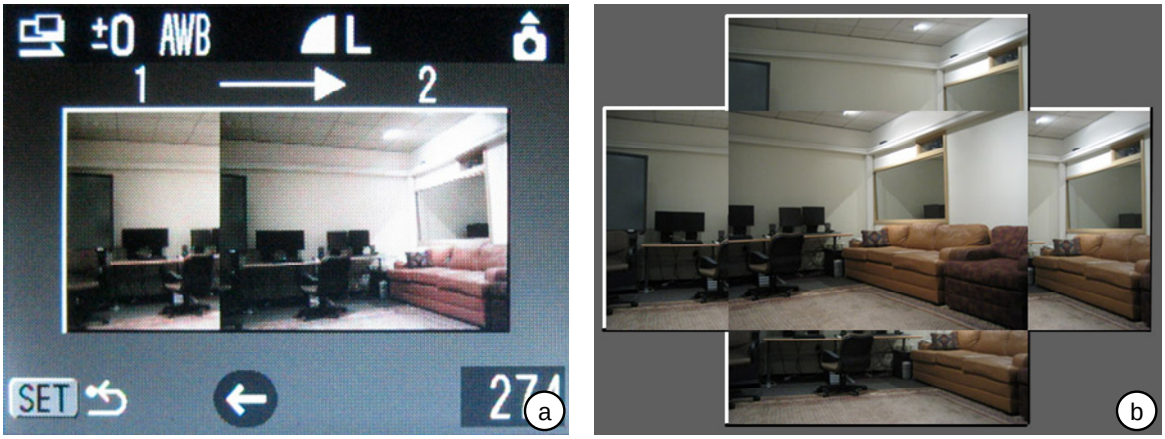

Figure 4: (a) The Stitch Assist mode, here from a Canon Powershot camera tells users where to take the next picture. The user pans the camera until the next picture on the right lines up with the piece of the previous picture on the left. (b) Our 2D adaptation of Stitch Assist allows users to continue a panoramic picture in any of the four main directions.

The HP Photosmart R707 camera allows users to review panoramas on the camera's LCD screen in playback mode. This allows users to see flaws while still on-site. Fixing flaws, however, still requires retaking the entire panorama.

The interaction types embodied by Panoramic Viewfinder have been peripherally explored in small-screen device interaction research. For example, Rekimoto (1996) explored how users can interact with devices by tilting them. Peep-hole displays (Yee, 2003) and the Chameleon system (Fitzmaurice, 1993) allow users to explore digital and physical spaces by moving a display around. Panoramic Viewfinder allows users to "sweep" the camera over the desired areas; similar interactions have been explored in the context of paintable interfaces (Baudisch, 1998).

Displaying a panorama and a viewfinder on the same small screen could be accomplished in a variety of ways. We chose the focus-in-context approach, which was shown to be cognitively less demanding than approaches that require users to switch attention between multiple views (Baudisch et al., 2002).

4. IMPLEMENTATION

Our Panoramic Viewfinder prototype is written in C and runs under Windows XP. We have two implementations of the front end, one using the GDI+ graphics library and one based on DirectX9.

While we focus this paper on describing the interface innovation in this system, the technical details of our stitching engine can be found in (Steedly et al., 2005). It is loosely based on the traditional offline stitcher described in (Brown et al., 2005) and shares some libraries with it. Similar to the offline stitcher, the real-time stitcher in Panoramic Viewfinder performs stitching in four steps: extracting multi-scale oriented patches from each new image, matching them with previous images, estimating the orientation of the camera when the new image was taken, and warping the new image onto the composite. Details can be found in (Steedly et al., 2005).

Panoramic Viewfinder is different from the offline stitcher in that it allows adding an image without reprocessing the images already stitched. To make this possible, Panoramic Viewfinder interleaves the four steps listed above. It finds the best location for each new image, but keeps all existing relationships fixed. While this leads to more error accumulation than the offline approach which minimizes errors between any pair of images, it allows new frames to be added in roughly constant rather than quadratic time.

Using the optimizations described above, our prototype reaches frame rates of up to 4 frames/sec on a 2GHz tablet PC with 1GB of RAM. We obtain slightly slower performance on an ultra portable 1 GHz Sony U50 (Figure 2).

4.1. Computing the cropping frame in real-time

The user interface concept underlying Panoramic Viewfinder relies heavily on its ability to update the cropping frame in real-time. Here we describe our algorithm that makes this possible.

The real-time stitcher outputs the panorama as an alpha-matted composite bitmap image. In order to place the cropping frame, the system computes the largest interior rectangle that contains only fully opaque pixels. The problem of computing the largest inscribed rectangle in an arbitrary shaped polygon has been studied in the computational geometry community. However, existing algorithms (e.g. Alt et al., 1995) require the geometry to be described as a polygon, which does not cover cases involving interior transparent regions. Also, converting the rounded outlines of a panorama to a polygon can be computationally expensive. We therefore created an algorithm based on a pixel-based representation of the panorama.

At the highest level, our algorithm tests all possible combinations of top-left and bottom-right corners thereby testing all possible rectangles that can be inscribed within a given region. The algorithm obtains real-time performance by eliminating top-left corners of boxes with expected areas lower than a moving upper bound and by limiting the search for the bottom right corners to a linear number of possible candi-

dates. By doing this, we obtain a complexity of roughly $O(n^2)$, with n being the width of the picture, which is suitable for real-time use. The detailed steps in this process are:

Step 1—down-sampling: For faster performance, our algorithm starts by creating a version of the panorama bitmap that is down-sampled by a factor of four. All further computation takes place on this down-sampled version.

Step 2—pre-compute span lengths: When computing rectangle surfaces, our algorithm uses information about the length of spans. Since the same span information is used repeatedly, it is cached as follows. The algorithm creates two arrays called *width image* and *height image*. For each pixel, *width image* stores the length of the horizontal span right of this pixel, *height image* stores the length of vertical span below this pixel. Both are created simultaneously by traversing the panorama from last row to first row and each of these rows from last column to first column. Since each span value is generated based on its right or bottom neighbor, this step requires $O(n^2)$ operations.

Step 3—compute rectangle surfaces: Now the algorithm searches for the largest interior rectangle. It initializes the largest found rectangle to area 0. Then it traverses the image top-down and left to right. At each pixel location (a) the algorithm computes an upper bound for the size of the largest possible rectangle whose top-left corner is at that pixel location. The upper bound is computed as the product of the horizontal span times the vertical span—both are looked up in width image and height image. (b) If the upper bound is smaller than the current best then this pixel cannot be the top-left corner of the sought rectangle and the algorithm moves on to the next pixel. (c) If the upper bound is larger than the current best, then the algorithm computes the true largest rectangle with this particular top-left corner. In order to do so, it traverses down the current column of the width image, looking for the bottom left corner that results in the largest rectangle surface. During this traversal, it keeps track of the largest rectangle created by any bottom left corner so far. The algorithm also keeps track of the length of the shortest horizontal span seen so far in this column. The width of a rectangle with a given bottom left corner is the shortest span seen above. For each pixel in that column, the algorithm computes the surface of the rectangle that has the respective pixel as its bottom right corner as the product or that width times its distance from the top left corner. It keeps updating the best value until all pixels are traversed.

In the worst case, *step 3* of our algorithm takes $O(n^3)$ operations. However due its ability to terminate early we find that in practice it is closer to $O(n^2)$ and is suitable for our real-time requirements. Interestingly, performance analysis shows that the algorithm spends 80% of its time in *Step 2*. As a future optimization we therefore plan to improve performance further by computing width image and height image iteratively as new input images are added.

5. CONCLUSIONS

In this paper, we have presented Panoramic Viewfinder, an interactive system for panorama construction that offers a real-time preview of the panorama while shooting. Panoramic Viewfinder makes ghosting, missing content, and stitching failures visible, thereby allowing users to fix them by immediately retaking necessary pictures.

Additionally, showing users the result of their work right away offers “instant gratification”, which we believe is one reason why many users today prefer digital photography to traditional photography. By automating the remaining steps in the process, i.e., high-quality restitching and auto cropping, Panoramic Viewfinder brings panoramic pictures one step closer to the user experience that users expect from regular digital photography, but not yet from panorama photography.

As future work, we are planning to optimize our algorithms to increase the frame rate and to reduce accumulated error. We also plan on running a user study evaluating user’s performance and experience with Panoramic Viewfinder.

6. REFERENCES

Alt, H., Hsu, D., and Snoeyink, J. (1995). Computing the Largest Inscribed Isothetic Rectangle. *Proceedings of 1995 Canadian Conference on Computational Geometry (CCCG)*, 67–72.

Baudisch, P. (1998). Don't Click, Paint! Using Toggle Maps to Manipulate Sets of Toggle Switches. *Proceedings of 1998 ACM Symposium on User Interface and Software Technology (UIST)*, 65-66.

Baudisch, P., Good, N., Bellotti, V., and Schraedley, P. (2002). Keeping Things in Context: A Comparative Evaluation of Focus Plus Context Screens, Overviews, and Zooming. *Proceedings of CHI 2002 Conference on Human Factors in Computing Systems*, 259-266.

Bergen, J. R., Anandan, P., Hanna, K. J., and Hingorani, R. (1992). Hierarchical model-based motion estimation. *Second European Conference on Computer Vision (ECCV)*, 237–252.

Brown, M. and Lowe, D.G. (2003). Recognising Panoramas. Proceedings of the 9th International Conference on Computer Vision (ICCV), 1218-1225.

Brown, M., Szeliski, R., and Winder, S. Multi-image matching using multi-scale oriented patches. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2005)*, Volume I, 510-517.

Edwards, B. (1999). *The New Drawing on the Right Side of the Brain*. Jeremy Tarcher Publishing, Expanded Edition.

Fitzmaurice, G. (1993). Situated Information Spaces and Spatially Aware Palmtop Computers. *Communications of the ACM, 36(7)*, 38–49.

Irani, M. and Anandan, P. (1998). Video indexing based on mosaic representations. *IEEE Transactions on Pattern Analysis and machine Intelligence, 86(5)*, 905–921.

Mann, S. and Picard, R. W. (1994). Virtual bellows: Constructing high-quality images from video. *Proceedings of First IEEE International Conference on Image Processing (ICIP-94)*, 363–367.

Peleg, S., and Herman, J. Panoramic Mosaics by manifold projection. (1997). *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR)*, 338.

Rekimoto, J. Tilting Operations for Small Screen Interfaces. Proceedings of 1996 ACM Symposium on User Interface and Software Technology (UIST), 167–168.

Sawhney, H.S., Kumar, R., Gendel, G., Bergen, J., Dixon, D., Paragano, V. (1998). VideoBrush™: Experiences with consumer video mosaicing. *Proceedings of IEEE Workshop on Applications of Computer Vision (WACV)*, 56–62.

Sawhney, H.S., Hsu, S., and Kumar, R. Robust Video Mosaicing through Topology Inference and Local to Global Alignment. (1998). In *Proc. European Conf. On Computer Vision*, 103-119.

Steedly, D., Pal, C., and Szeliski, R. Efficiently Registering Video into Panoramic Mosaics. To appear in *Proc. Tenth International Conference on Computer Vision (ICCV 2005)*.

Szeliski, R. (1996) Video mosaics for virtual environments. *IEEE Computer Graphics and Applications, 16(2)*, 22–30.

Szeliski, R., and Shum, H. (1997). Creating full view panoramic image mosaics and environment maps. *Proceedings of SIGGRAPH 1997 31st International Conference on Computer Graphics and interactive Techniques*, 251–258.

Teodosio, L. and Bender, W. (1993). Salient video stills: Content and context preserved. *Proceedings of ACM Multimedia 1993*, 39–46.

Yee, K.P. (2003). Peephole displays: Pen interaction on spatially aware handheld computers. *Proceedings of CHI 2003 Conference on Human Factors in Computing Systems*, 1-8.

7. ACKNOWLEDGEMENTS

We would like to thank the members of the Interactive Visual Media Group and the Visualization and Interaction Group at Microsoft Research for their feedback on an earlier version of this paper.