Automatic Head-size Equalization in Panorama Images for Video Conferencing

Ya Chang, Ross Cutler, Zicheng Liu, Zhengyou Zhang, Alex Acero, Matthew Turk

yachang@cs.ucsb.edu
University of California
Santa Barbara, CA 93106

{rcutler,zliu,zhang,alexac}@microsoft.com
Microsoft Research
Redmond, WA 98052

mturk@cs.ucsb.edu
University of California
Santa Barbara, CA 93106

Abstract

In panorama images captured by omni-directional cameras during video conferencing, the image sizes of the people around the conference table are not uniform due to the varying distances to the camera. Spatially-varying-uniform (SVU) scaling functions have been proposed to warp a panorama image smoothly such that the participants have similar sizes on the image. To generate the SVU function, one needs to segment the table boundaries, which was generated manually in the previous work. In this paper, we propose a robust algorithm to automatically segment the table boundaries. To ensure the robustness, we apply a symmetry voting scheme to filter out noisy points on the edge map. Trigonometry and quadratic fitting methods are developed to fit a continuous curve to the remaining edge points. We report experimental results on both synthetic and real images.

1. Introduction

In the past a few years, there has been a lot of interest in the use of omni-directional cameras for video conferencing and meeting recording [1,3,4,5]. While a panoramic view is capable of capturing every participant's face, one drawback is that the image sizes of the people around the meeting table are not uniform in size due to the varying distances to the camera. Figure 1 shows a 360 degree panorama image of a meeting room. The table size is 10x5 feet. The person in the middle of the image appears very small compared to the other two people because he is further away from the camera.

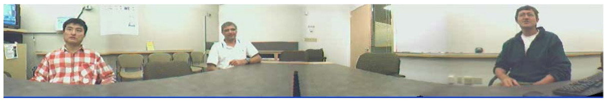

Fig. 1: An image captured by an omni-directional camera

This has two consequences. First, it is difficult for the remote participants to see some faces, thus negatively affecting the video conferencing experience. Second, it is a waste of the screen space and network bandwidth because a lot of the pixels are used on the background instead of on the meeting participants. As image sensor technology rapidly advances, it is possible to design inexpensive high-resolution (more than 2000 horizontal pixels) omni-directional video cameras [1]. But due to network bandwidth and user's screen space, only a smaller-sized image can be sent to the clients. Therefore how to effectively use the pixels has become a critical problem in improving the video conferencing experience.

Spatially-varying-uniform (SVU) scaling functions have been proposed [2] to address this problem. A SVU scaling function warps a panorama image to equalize people's head sizes without creating discontinuities. Fig. 2 shows the result after head-size equalization.

The generation of a SVU function, as described in [2], requires two curves: the bottom curve specifies the table boundaries, and the top curve along people's head top positions. In the previous work, the two curves were created manually. The problem with the manual segmentation is that whenever the camera is moved or rotated, the user has to manually mark an image, thus making it difficult to use. In this paper, we describe a technique to automatically segment the table boundaries and estimate the two curves. As a result, the SVU function can be generated automatically.

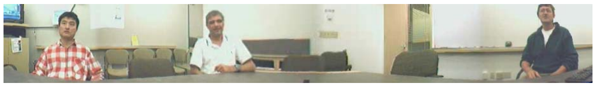

Fig. 2: Result after head-size equalization

2. Cylindrical projection of an omni-directional camera

Fig. 3 shows a mathematical model of the cylindrical projection. The camera is at the center of a rectangular table of size $2W*2L$. The projection center is $(0,0,h)$. The radius of the cylindrical film is r. The projection of the table edge from $(W,L,0)$ to $(W,-L,0)$ on the cylindrical film is

$$v = h(1 - r\cos\theta / W) \qquad (2\text{-}1)$$

where $\theta \in [-\arctan(L/W), \arctan(L/W)]$. The projections of the other three table edges can be obtained similarly.

0-7803-9332-5/05/$20.00 ©2005 IEEE

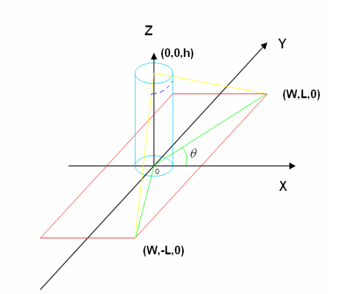

Fig. 3: Cylindrical projection of the table edges

Consider a more general case where the camera may not be at the center of table, and the camera may be tilted. Let $(\Delta w, \Delta l)$ denote the camera shift with respect to the table center. Let (ω, α) denote the camera tilt where ω specifies the direction in which the camera tilts, and α is the tilt angle. The projected curve of the table edge is

$$v = h(1 - r\cos\theta/(W + \Delta w)) + r\cos(\theta - \omega)\sin\alpha \qquad (2\text{-}2)$$
$$\theta \in [-\arctan((L - \Delta l)/(W - \Delta w)), \arctan((L - \Delta l)/(W + \Delta w))]$$

(v, θ) represents a point on the cylindrical film. Assume we cut the film at $\theta = hs$, and flatten it. Let (v, φ) represent a point on the flattened film. Then $\theta = \varphi + hs$. If we substitute it into equations (2-2), we will obtain the equations of the projected curves in the unfolded film coordinates.

In general, there is a one-to-one mapping from the points on the cylindrical film and the points on the meeting table through the projection center $(0,0,h)$. Due to space constraints, we will omit the derivations and formula for the mapping functions. We will use F_1 and F_2 to denote the mapping from the table to the cylindrical film, and F_1^{-1} and F_2^{-1} to represent the mapping from cylindrical film to the table. That is,

$$\begin{aligned} v &= F_1(x, y, z, h, r, \omega, \alpha, \Delta w, \Delta l, hs); \\ \varphi &= F_2(x, y, z, h, r, \omega, \alpha, \Delta w, \Delta l, hs); \qquad (2\text{-}3) \\ x &= F_1^{-1}(v, \varphi, h, r, \omega, \alpha, \Delta w, \Delta l, hs); \\ y &= F_2^{-1}(v, \varphi, h, r, \omega, \alpha, \Delta w, \Delta l, hs); \end{aligned}$$

3. Symmetry Voting

If we apply a general image segmentation algorithm such as EDISON [8], the result is quite noisy. Figs. 10 and 11 show an example. We can see the edge map in Fig. 11 contain a lot of noises in addition to the table boundaries. To filter out the edge map, we observe that most conference tables are bilaterally symmetric. Our idea is to take advantage of the symmetry property to filter out the noise.

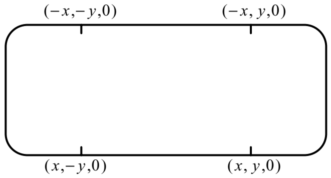

Fig. 4: The top down view of the table.

As shown in Fig. 4, each 3D point $(x, y, 0)$ on the table boundary has three symmetric points: $(-x, y, 0)$, $(x, -y, 0)$, $(-x, -y, 0)$. If a 2D point p on the edge map corresponds to the 3D point $(x, y, 0)$, then the other three points (we call them p's *symmetric projections*) which correspond to 3D points $(-x, y, 0)$, $(x, -y, 0)$, and $(-x, -y, 0)$ respectively, must be on the edge map as well.

The symmetry voting works by enumerating all the possible values of $\Delta w, \Delta l, hs, \omega, \alpha$. For each hypothesis, it checks each point on the edge map to see how many of its symmetric projections are also on the edge map, and increments the accumulated weight accordingly. Finally the hypothesis corresponding to the largest accumulated weight is selected as the solution.

To accelerate computation, we perform the symmetry voting method in two steps: we first vote for $\Delta w, \Delta l, hs$, then vote for camera tilt. The following is the algorithm of the first step voting.

```
Clear global h[ ][ ][ ]
For $\Delta w$=w_min; $\Delta w$ <= w_max; $\Delta w$++
  For $\Delta l$ =l_min; $\Delta l$ <= l_max; $\Delta l$ ++
    For hs=hs_min; hs <=hs_max; hs++
      For each edge point $(v,\varphi)$ , find $(x,y)$ by Eq. (2-3)
          Update(-x,y, $\Delta w$ , $\Delta l$ ,hs);
          Update(x,-y, $\Delta w$ , $\Delta l$ ,hs);
          Update(-x,-y, $\Delta w$ , $\Delta l$ ,hs);
End all
Find the maximum of h[ ][ ][ ] and return its index as ( $\Delta w$ , $\Delta l$ ,hs).

Function Update(x,y, $\Delta w$ , $\Delta l$ ,hs)
    Find $(v,\varphi)$ given (x,y) by Eq. (2-3)
    If an edge point $(tv,t\varphi)$ falls within a window of $(v,\varphi)$
      Then h[ $\Delta w$ ][ $\Delta l$ ][hs]+=1/distance( $(tv,t\varphi)$ $(v,\varphi)$ );
```

In the second voting, we fix the $\Delta w, \Delta l, hs$, and use a similar algorithm to vote for (ω, α) by using functions $F_{1,2}, F_{1,2}^{-1}$. Note that we could potentially use the newly estimated (ω, α) and vote for c again. But we found that the first iteration is usually enough.

After we obtain the optimal values for $\Delta w, \Delta l, hs$ and (ω, α), we then filter out the edge map using the symmetry properties. Given any point on the edge map which is the projection of $(x, y, 0)$, we say it *satisfies symmetry property* if at least two[1] of its symmetric projections are also on the edge map.

We remove all the points on the edge map which do not satisfy the symmetry property.

For each point that satisfies the symmetry property, we add the other three projections of its symmetric points to the edge map. The resulting map is called *symmetry-enhanced edge map.* Fig. 12 shows the symmetry-enhanced edge map after applying the filtering operation on the edge map of Fig. 11.

4. Fitting Algorithms

The points on the symmetry-enhanced edge map are the inputs of the fitting algorithm. We have developed two different fitting techniques: trigonometry fitting and quadratic fitting. Due the page limit, we will only give a brief summary for each fitting method.

The trigonometry fitting is a model-based approach. By assuming a rectangular table, the projected curve has the form of Eq. (2-2) where *W, L* are the unknown parameters that we need to estimate. Since the edge map is symmetry-enhanced, we only need to use the points on two of the four table edges. As shown in Fig. 11, the two sections are the one between *cut"* and *cut,* and the one between *cut* and *cut'* where *cut*, *cut'*, and *cut"* are all functions of *W, L.*

The limitation of the trigonometry fitting is that it assumes a rectangular shape. For non-rectangular tables, the fitting is not as accurate.

Quadratic fitting does not have such limitations. In quadratic fitting, we use two quadratic curves to fit the edge points. To regulate the fitted curve, we require that each of the quadratic curve has the form $y = a + b(x - c)^2$.

The red curves in Figs. 13 and 14 are the results of the two fitting methods.

After we obtain the table dimensions and camera parameters, we estimate the top curve as the following. We first assume an average height of a person (sitting at the table) above the table surface. Then we create a virtual table surface at this height and project its boundary to the cylindrical film. The details are omitted here. We then apply the SVU scaling function to equalize people's head sizes. Fig. 15 shows the result based on trigonometry fitting.

5. Experimental Results

We have tested our algorithm intensively on both synthetic data and real data. The synthetic data are generated by a 3D graphics rendering program. The purpose of using synthetic data is to test the robustness of our algorithm on various conditions including different camera tilts, different noise levels, and different table shapes.

To measure the performance when there is camera tilt, we set the step size of ω to be 50^o, and α to be 0.2^o and generate synthetic images with different camera tilts. For each synthetic image, we first use EDISON [8] software to generate the edge map. Then our symmetry voting algorithm is used to estimate ω and α. The average error for the estimated ω is 5.14^o, and for α is 0.0786^o. Fig. 5 shows the error of ω for different values of α. It is interesting to note that for smaller α, the estimation error of ω is larger. The reason is that when α is small, the direction of the tilt ω becomes ambiguous. At the extreme when α=0, ω is arbitrary.

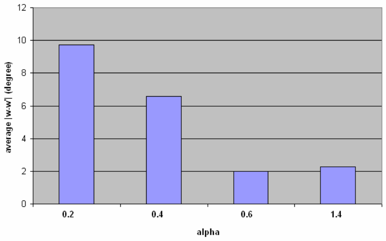

Fig. 5: Error of ω under different α

To measure the performance under different noise levels, we add random black squares of size 10*10 pixels to the synthetic images. The noise level was controlled by the number of the blocks *N.* Fig. 6 in the last page is an example of the synthetic image with 100 blocks. We use a boat-shaped (non-rectangular) table for this experiment. The results are in Fig. 7. The average error is the average distance (in pixels) between the fitting curve and the ground truth. We can see that in less noisy conditions, quadratic fitting works better than the trigonometric fitting. The reason is that the table shape is non-rectangular. When the noise increases, trigonometric fitting has a stronger regulation thus is more robust against noises.

To measure the performance with different table sizes and shapes, we generate synthetic data with two table shapes: boat shape and rectangular shape, and three different aspect ratios: 1:2, 1:3, and 1:4. The camera tilt is set to $\omega = 90^o, \alpha = 0.6^o$. The noise level is set to N=150. Fig. 8 shows the results for boat-shaped tables. Fig. 9 shows the results for rectangular tables. We can see that in all circumstances, the maximum error is less than 2 pixels for both fitting methods.

[1] Requiring all the three projections on the edge map would be too strict due to inaccuracies of the estimated parameters and image noises.

Quadratic fitting works better for boat-shaped table because it is able to handle arbitrary table shapes.

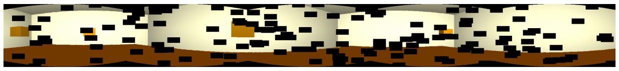

Fig. 6: Synthetic panorama image with N=100. The yellow blocks simulate human heads.

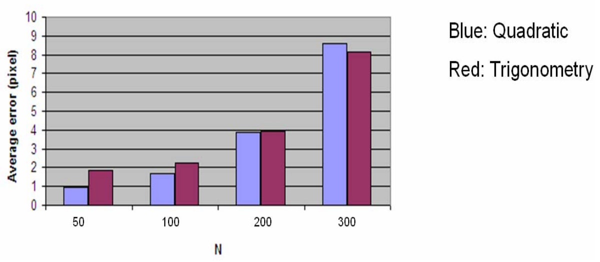

Fig. 7: Fitting error vs. noises on boat-shaped table

6. Conclusion

We have developed a novel technique to automatically detect table boundaries on 360° panorama images in meeting rooms. As a result, we are able to automatically generate SVU-scaling functions to equalize people's head sizes resulting in better video conferencing experience. Experiments show that our algorithm is robust under very noisy conditions. *(A longer version of this paper with more technical details can be found at http://research.microsoft.com/~zliu/TR-2005-48.pdf)*

References

[1] R. Cutler, Y. Rui, A. Gupta, JJ Cadiz, I. Tashev, L. He, A. Colburn, Z. Zhang, Z. Liu, S. Silverberg, "Distributed Meetings: A Meeting Capture and Broadcasting System", ACM Multimedia 2002.

[2] Z. Liu, M. Cohen, "Head-Size Equalization for Better Visual Perception of Video Conferencing", IEEE International Conference on Multimedia and Expo, Amsterdam, The Netherlands, July 2005.

[3] R. Stiefelhagen, X. Chen, J. Yang, "Capturing Interactions in Meetings with Omnidirectional Cameras, International workshop on Multimedia Technologies in E-learning and Collaboration". Nice, France, 2003.

[4] A. Waibel, T. Schultz, M. Bett, M. Denecke, R. Malkin, I. Rogina, R. Stiefelhagen, J. Yang, "SMaRT: The Smart Meeting Room Task At ISL", ICASSP 2003.

[5] F. Wallhoff, M. Zobl, G. Rigoll, I. Potucek, "Face Tracking in Meeting Room Scenarios Using Omnidirectional Views", IEEE International Conference on Pattern Recognition, 2004.

[6] H. Nanda, R. Cutler, "Practical Calibrations for a real-time digital omnidirectional camera", Technical Sketches, Computer Vision and Pattern Recognition, Hawaii, US, Dec 2001.

[7] http://www.behere.com

[8] P. Meer, B. Georgescu: "Edge detection with embedded confidence." IEEE Trans. Pattern Anal. Machine Intell., 23, 1351-1365, December 2001.

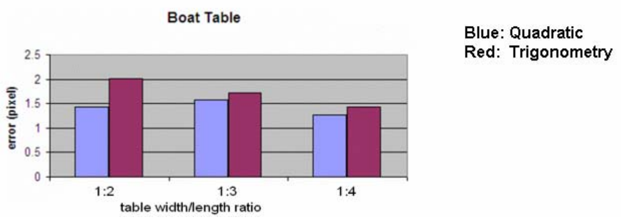

Fig. 8: Fitting error vs. different table sizes for the boat shaped table.

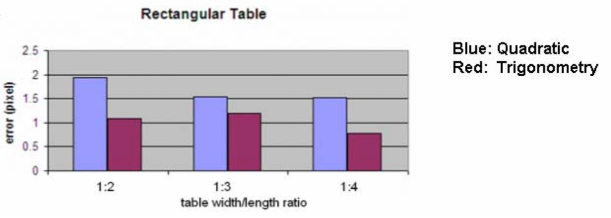

Fig. 9: Fitting error vs. different table sizes for the rectangular table.

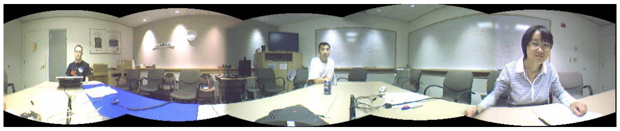

Fig. 10: A real image with clutter background.

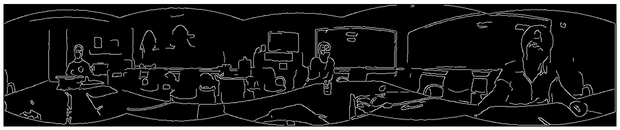

Fig. 11: Edge map extracted from original image.

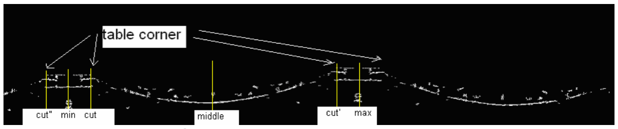

Fig. 12: Symmetry-enhanced edge map.

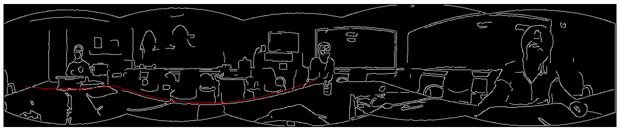

Fig. 13. Result of trigonometry fitting.

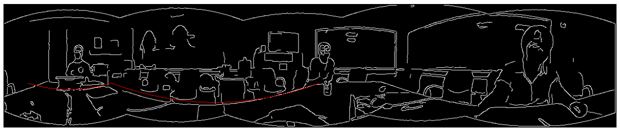

Fig. 14. Result of quadratic fitting

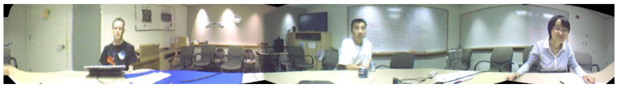

Fig.15. Head-size equalization based on the curve from trigonometry fitting.