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,
{rcutler,zliu,zhang,alexac}@microsoft.com
mturk@cs.ucsb.edu

May, 2005

MSR-TR-2005-48

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

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. 1-1: 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. 1-2: Result after head-size equalization

This paper will focus on the methods of automatic table edge detection. The whole algorithm runs in the following way:

Uncluttered table (setup):
- Detect general edge points
- Extract table edge points
- Learn parametric table model

Cluttered table (normal usage):
- Detect general edge points
- Extract table edge points
- Fit parametric table model

There will be 8 sections in this paper. Section 1 is the introduction. Section 2 talks about the general parametric model of the table edge in panorama images. Section 3 presents the symmetry voting and point filtering for extracting table edge points. Section 4 presents the quadratic edge fitting without table model. Section 5 presents trigonometry fitting with table model and compares its performance with quadratic fitting in case of partial table edges. Section 6 further provides an optional solution for real-time edge detection by ICP during conference. Section 7 presents the intensive testing on synthetic data. We will summarize in Section 8.

2. Basic Formulations

We talk about the formulation of the cylindrical projection of the table in this section.

2.1. Rectangular table with camera at canter

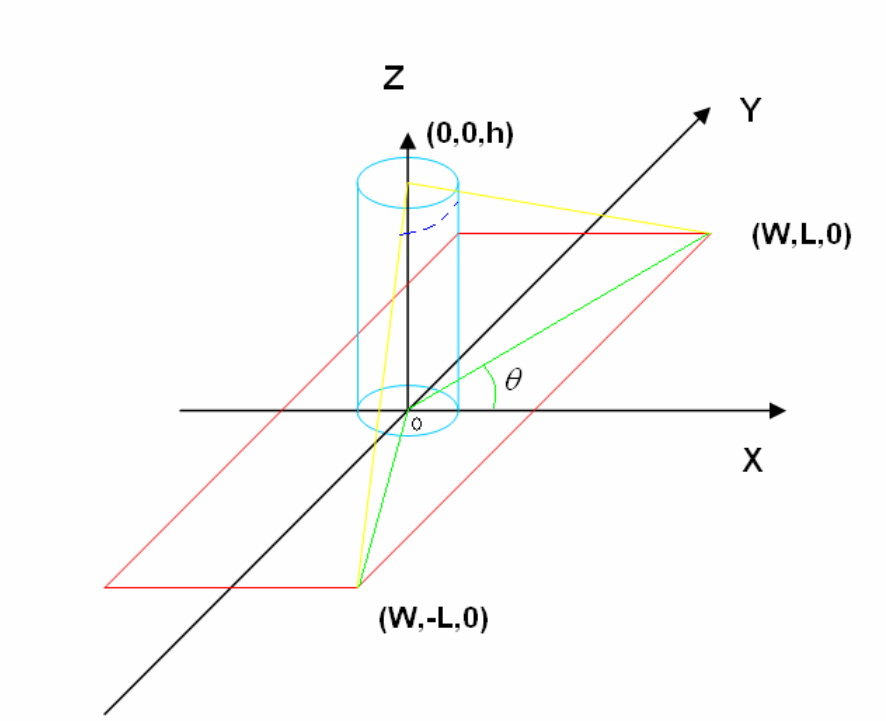

Fig. 2-1: Illustration of the cylindrical projection of the table edges.

We first consider the case that the Ring Camera is at the center of a rectangular table of $2W * 2L$. 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 $(W,-L,0)$ is the intersection of the plane $\frac{x}{W}+\frac{z}{h}=1$ and cylindroid $x^2 + y^2 = r^2$. It is illustrated as a dash curve in Fig. 1-1. The cylindroid in Cartesian coordinates can also be represented in cylindrical coordinate as $\begin{cases} x = r\cos\theta \\ y = r\sin\theta \end{cases}$. So the intersection curve becomes

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

It begins from the intersection of the cylondroid and the yellow line ($(W,-L,0)->(0,0,h)$), and ends at the intersection of the cylondroid and the yellow line ($(W,L,0)->(0,0,h)$). That is $\theta\in[-\arccos(t),\arccos(t)], t=\sqrt{\frac{W^2}{W^2+L^2}}$ on curve (2-1).

In the same way, the project of the table edge from $(W,L,0)$ to $(-W,L,0)$ is

$$z = h(1-\frac{r\sin\theta}{L}), \text{ with } \theta\in[\arccos(t),\pi-\arccos(t)], t=\sqrt{\frac{W^2}{W^2+L^2}} \qquad (2\text{-}2)$$

the project of the table edge from $(-W,L,0)$ to $(-W,-L,0)$ is

$$z = h(1 + \frac{r\cos\theta}{W}), \text{ with } \theta \in [\pi - \arccos(t), \pi + \arccos(t)], t = \sqrt{\frac{W^2}{W^2 + L^2}} \quad (2\text{-}3)$$

the project of the table edge from $(-W, -L, 0)$ to $(W, -L, 0)$ is

$$z = h(1 + \frac{r\sin\theta}{L}), \text{ with } \theta \in [\pi + \arccos(t), 2\pi - \arccos(t)], t = \sqrt{\frac{W^2}{W^2 + L^2}} \quad (2\text{-}4)$$

When we unfold the cylindroid film, the panorama image will be like Fig. 2-2.

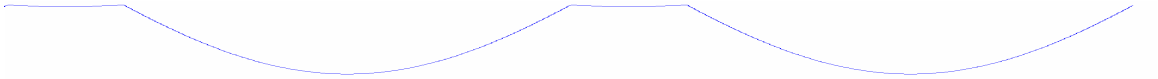

Fig. 2-2: Perfect projected table edge on panorama images

Fig. 2-3: An example of the real images

2.2. Camera with position and orientation change

When the orientation of the Ringcam changes, the projected table edge will just shift to left/right by the corresponding angle. It will not change the shape of the curve.

When the camera is not at the center of the table, $(\Delta w, \Delta l)$. The intersection of the plane $\frac{1}{W}x + \frac{(1 - \Delta w/W)}{h}z = 1$ and the cylindroid $(x - \Delta w)^2 + (y - \Delta l)^2 = r^2$ (that is equal to $\begin{cases} x = \Delta w + r\cos\theta \\ y = \Delta l + r\sin\theta \end{cases}$) is

$$z = h(1 - \frac{r\cos\theta}{W - \Delta w}) = h(1 - \frac{r\cos\theta}{W(1 - \frac{\Delta w}{W})}) = h(1 - \frac{r\cos\theta}{W}(1 + \frac{\Delta w}{W})) \quad (2\text{-}5)$$

(we neglect high order taylor expansion here since $\Delta w << W, \Delta l << L$)

$\theta \in [-\phi_1, \phi_2]$,

with $\phi_1 = \arccos(\frac{W - \Delta w}{\sqrt{(W - \Delta w)^2 + (L - \Delta l)^2}}), \phi_2 = \arccos(\frac{W - \Delta w}{\sqrt{(W - \Delta w)^2 + (L + \Delta l)^2}})$

Similarly, the projections of the other table edges are

$$z = h(1 - \frac{r\sin\theta}{L}(1 + \frac{\Delta l}{L}))\text{, with } \theta \in [\phi_2, \pi - \phi_3] \quad (2\text{-}6)$$

$$\phi_3 = \arccos(\frac{W + \Delta w}{\sqrt{(W + \Delta w)^2 + (L - \Delta l)^2}})$$

$$z = h(1 + \frac{r\cos\theta}{W + \Delta w}) = h(1 + \frac{r\cos\theta}{W}(1 - \frac{\Delta w}{W}))\text{, with } \theta \in [\pi - \phi_3, \pi + \phi_4] \quad (2\text{-}7)$$

$$\phi_4 = \arccos(\frac{W + \Delta w}{\sqrt{(W + \Delta w)^2 + (L + \Delta l)^2}})$$

$$z = h(1 + \frac{r\sin\theta}{L}(1 - \frac{\Delta l}{L}))\text{, with } \theta \in [\pi + \phi_4, 2\pi - \phi_1] \quad (2\text{-}8)$$

2.3. Model for Tilt Camera

In practical situations, there is usually small tilt for cameras. We model this kind of situation in this section.

The focus of the camera tilt α degrees along direction ω. We can also see that the table first tilt $-\alpha$ degrees around the axle $y = \tan(\omega + \frac{\pi}{2})x$ (green line in Fig. 2-4).

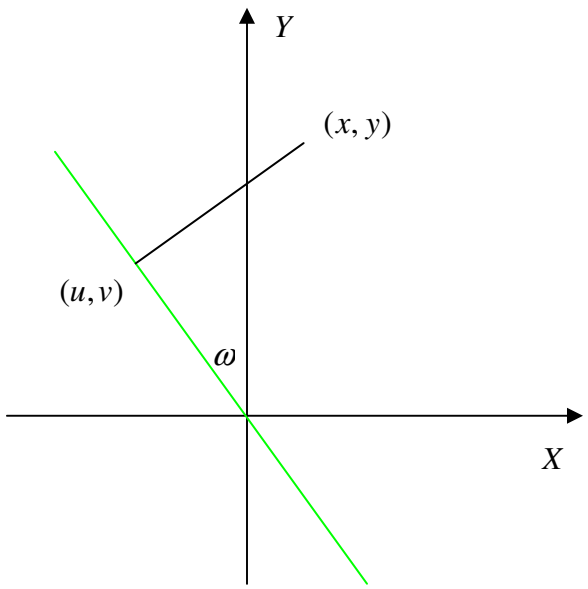

Fig. 2-4. Table rotate illustration (top view)

A point (x,y,0) in X-Y plane rotate around axle $y = \tan(\omega + \frac{\pi}{2})x$ degree α. Its projection on the axle is (u,v). Without lost of generality, we can add constraint $\omega \in [0, 2\pi], \alpha > 0$.

$(u, v) = (x\sin\omega - y\cos\omega) * (\sin\omega, -\cos\omega)$;

Its coordinates after rotation is (x', y', z') where

$$(x', y') = (u, v) + ((x, y) - (u, v)) * \cos\alpha;$$
$$x' = \sin\omega(x\sin\omega - y\cos\omega)(1 - \cos\alpha) + x\cos\alpha;$$
$$y' = -\cos\omega(x\sin\omega - y\cos\omega)(1 - \cos\alpha) + y\cos\alpha;$$
$$z' = (x\cos\omega + y\sin\omega) * \sin\alpha;$$
$$d = \sqrt{x'^2 + y'^2} = \sqrt{(x\sin\omega - y\cos\omega)^2 \sin^2\alpha + (x^2 + y^2)\cos^2\alpha}\,.$$

So the projection of the point on the cylindrical film is

$$\theta = \arctan(\frac{y'}{x'}) + cond - hs;$$
$$\text{Im\_}\, y = h - \frac{r(h - z')}{d};$$

Where $cond = \pi$ when $x' < 0$; $cond = 2\pi$; when $y' < 0, x' > 0$.

θ is proportional to the x coordinate in panorama image coordinate, Im_y can be transformed to y coordinates in panorama image coordinate by a simple vertical shift.

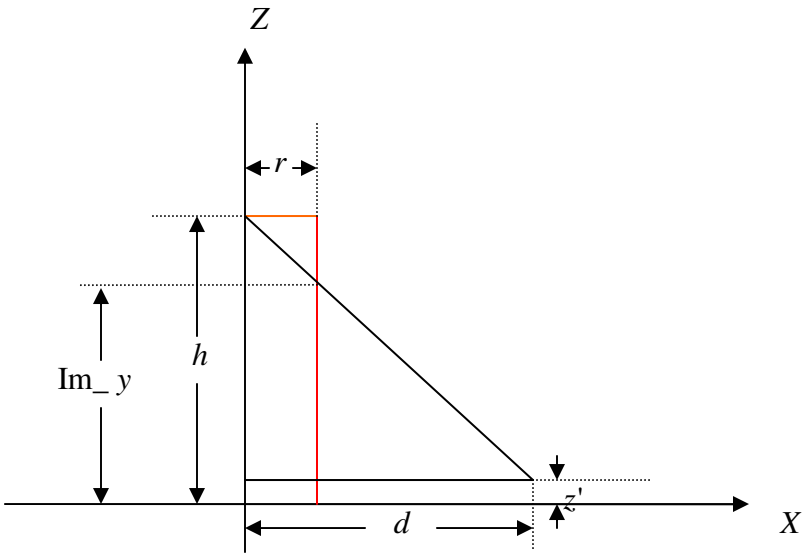

Fig. 2-5. Table rotate illustration (side view)

2.4. Various Table Shapes

2.4.1. Boat shape long table edges

The long table edge can be boat shape. We will analyze the error if we simulate this kind of table shape using the rectangular shape. Assume the camera is at the center of the table, and

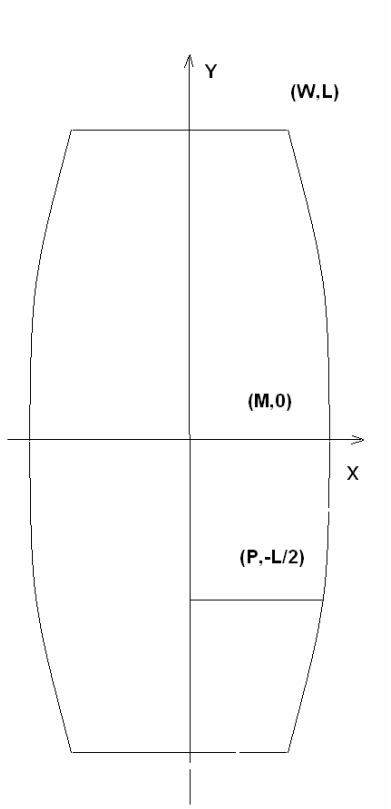

Fig. 2-6. Boat-shape table

We use four cubic splines connecting (W,-L), (P,-L/2), (M,0), (P,L/2),(W,L) to simulate the boat-shape table edge. Usually, W<P<M, so we assume P=a*W, M=b*W. The equations of the splines are

$t = y / L; K = a - 1 - (b-1)/2;$

$$x = W*(1+2*(a-1)*(t+1)+4*K*(t+1)^2-8*K*(t+1)^3); -L < y < -L/2;$$

$$x = W*(a+(b-1)*(t+\frac{1}{2})+4*(2b-3a+1)*(t+\frac{1}{2})^2+8*(2a-\frac{3b}{2}-\frac{1}{2})*(t+\frac{1}{2})^3); -L/2 < y < 0;$$

$$x = W*(b+4*(3a-\frac{5}{2}b-\frac{1}{2})*t^2+8*(\frac{3}{2}b-2a+\frac{1}{2})*t^3); 0 < y < L/2;$$

$$x = W*(a+(1-b)*(t-\frac{1}{2})-8*K*(t-\frac{1}{2})^2+8*K*(t-\frac{1}{2})^3); L/2 < y < L$$

The point (u,v) will be mapped on the cylindrical film as $\begin{cases} \theta = \arctan(v/u) \\ z = (1-\frac{r}{\sqrt{u^2+v^2}})*h \end{cases}$. So when the point is moved to $(u+\Delta,v)$, the projection is

$$\begin{cases}\theta' = \arctan(\ v/(u+\Delta)) \\ z' = (1-\dfrac{r}{\sqrt{(u+\Delta)^2+v^2}})*h\end{cases}$$. We can see that using Taylor extension,

$$k=\frac{\Delta}{u}; R=\frac{v}{u};$$

$$\theta-\theta' = \arctan(\frac{\Delta * v}{u^2+v^2+u*\Delta}) \approx \frac{\Delta * v}{u^2+v^2+u*\Delta} \approx \frac{k*R}{1+R^2}*(1-\frac{k}{1+R^2})$$

$$z-z' \approx \frac{r*h}{\sqrt{u^2+v^2}}(1-\frac{k}{(1+R^2)}-\frac{k^2}{2*(1+R^2)})$$

Now our question is what is the best approximation for this kind of table edges and what is the maximum error in that case.

2.4.2. Round End Table

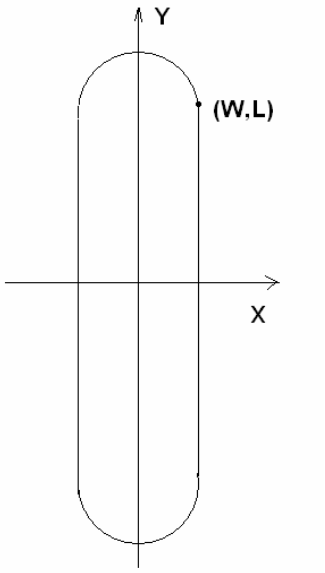

Fig. 2-7. Round end table

If the table end is a half circle, and the camera is at the center of the table, the half circle in the upper part of Fig. 2-7 can be represented as $\begin{cases}u = W\cos\alpha \\ v = L + W\sin\alpha\end{cases}, \alpha \in [0,\pi]$. Its projection on the cylindrical film is

$$\begin{cases}\theta = \arctan(\frac{L}{W}+tg\alpha) \in [\arctan\frac{L}{W}, \pi-\arctan\frac{L}{W}] \\ z = (1-\frac{r}{\sqrt{W^2+L^2+2*L*W*\sin\alpha}})*h \in [(1-\frac{r}{\sqrt{W^2+L^2}})*h, (1-\frac{r}{W+L})*h]\end{cases}$$

The closed form equation for the projected curve is

$$z = (1 - \frac{r}{\sqrt{W^2 + L^2 + 2 * L * W * \frac{tg\theta - L/M}{\sqrt{1 + (tg\theta - L/M)^2}}}}) * h$$

If we use a straight line to approximate it, the maximum error is

$$\frac{(1 - \frac{r}{W + L}) * h - (1 - \frac{r}{\sqrt{W^2 + L^2}}) * h}{2} \approx \frac{r * h}{2\sqrt{W^2 + L^2}} (\frac{W * L}{W^2 + L^2} - \frac{3}{2} (\frac{W * L}{W^2 + L^2})^2)$$

2.4.3. Trapezoid End table

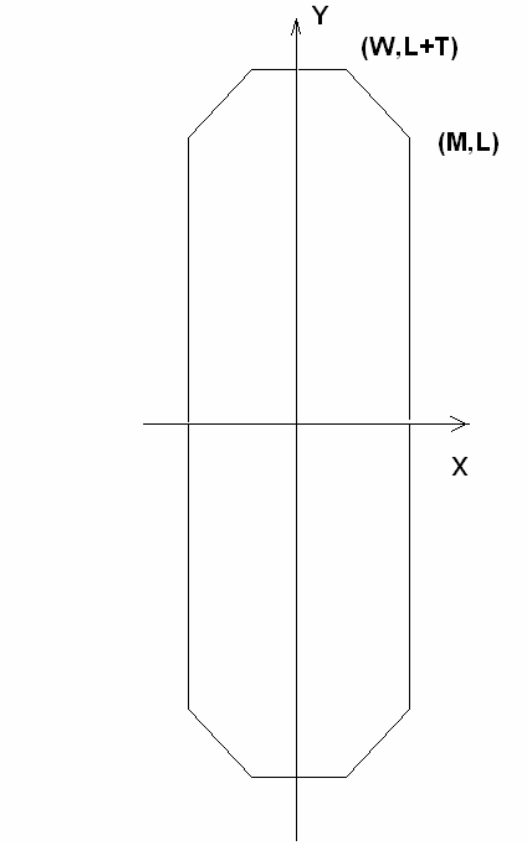

Fig. 2-8. Trapezoid end table

If the table shape is like Fig. 2-8, we need to consider the projection of the line from (M,L) to (W, L+T). Assume everything else keeps unchanged, the projected curve is the intersection of the plane

$$\frac{x}{a} + \frac{y}{b} + \frac{z}{h} = 1; \quad where \qquad a = M - \frac{(W - M) * L}{T}, b = L - \frac{T * M}{W - M}; x \in [W, M]$$

And the cylindroid $x^2 + y^2 = r^2$.

Its closed form solution is

$$z = h*(1-r*(\frac{\cos\theta}{a}+\frac{\sin\theta}{b})) \\ = h*(1-r*\sqrt{\frac{1}{a^2}+\frac{1}{b^2}}\cos(\theta+\arctan(\frac{T}{M-W}))) \quad ,\theta \in [\arctan\frac{L}{M},\frac{L+T}{W}].$$

3. Symmetry Voting

We can use the symmetry property of the table to calculate better initialization for camera position and orientation.

If we apply a general image segmentation algorithm such as EDISON [7], the result is quite noisy. The edge points higher than the highest of all possible table edges can be removed. The highest edge point is determined by (see Eq .2-1): max $h(1-\frac{r}{\sqrt{W^2+L^2}})$. Assume all conference table are smaller that 8 feet by 32 feet. The high threshold for all edge points can be calculated offline.

3.1. Symmetry Voting

The key idea is that the conference tables usually have a dual bilateral symmetry as shown in Fig. 3-1 and 3-2. We exploit this symmetry to estimate the RingCam orientation *hs* (horizontal shift), vs (vertical shift) and offset Δw and Δl.

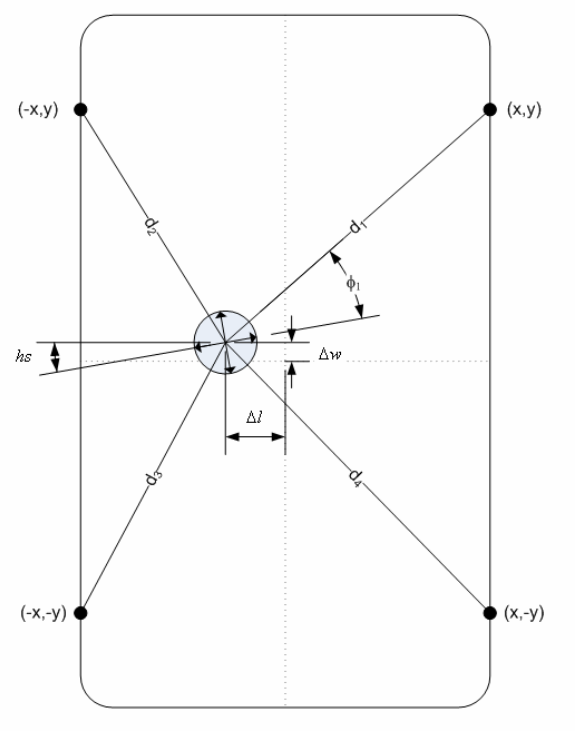

Fig. 3-1. RingCam on table (top view)

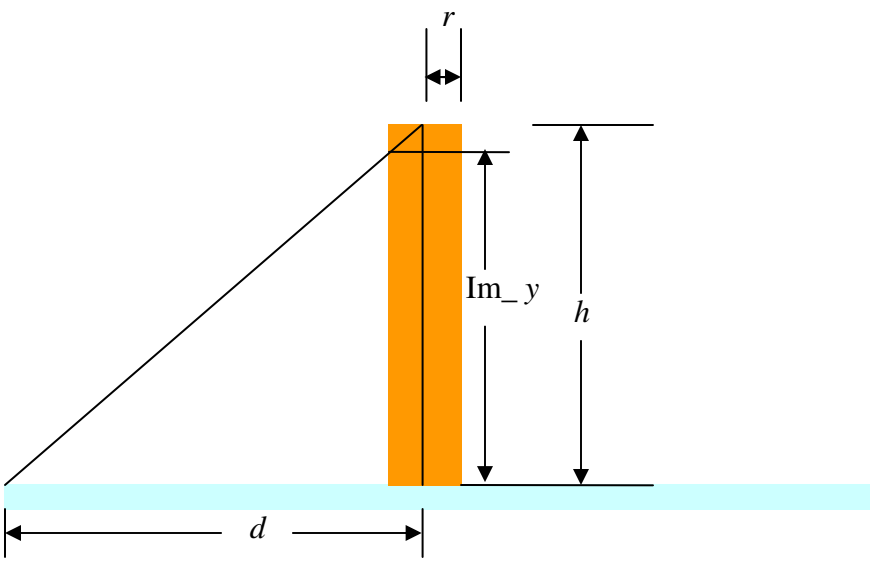

Fig. 3-2. RingCam on table (side view)

When the RingCam's axis of symmetry is normal to the table plane, a point (u,v) on the table edge in panorama image has a one-to-one mapping on the point (x, y) in world coordinate as in Fig. 3-1.

$\text{Im\_}\, y = v + vs;$

By the similar triangles, we have

$$\frac{d}{h} = \frac{d-r}{\text{Im\_}\, y}$$

$$\begin{cases} d = \dfrac{r * h}{h - v - vs} \\ \phi = \dfrac{u}{\text{Im}age\_width} + hs \end{cases} \tag{3-1}$$

By Fig. 3-1, we have

$$(x, y) = (d_1 \cos\phi_1 + \Delta w, d_1 \sin\phi_1 + \Delta l) \tag{3-2}$$

$$(-x, y) = (d_2 \cos\phi_2 + \Delta w, d_2 \sin\phi_2 + \Delta l)$$

$$(x,-y) = (d_3 \cos\phi_3 + \Delta w, d_3 \sin\phi_3 + \Delta l)$$

$$(-x,-y) = (d_4 \cos\phi_4 + \Delta w, d_4 \sin\phi_4 + \Delta l)$$

Given (x, y),

$$(d,\phi) = (\sqrt{(x-\Delta w)^2 + (y-\Delta l)^2}, \arctan(\frac{y-\Delta l}{x-\Delta w}) + cond) \tag{3-3}$$

Where $cond = \pi$ when $x - \Delta w < 0$;

$cond = 2\pi$; when $y - \Delta l < 0, x - \Delta w > 0$.

The basic algorithm is to detect edge points and do a voting scheme for the parameters $(\Delta w, \Delta l, hs.vs)$. The pseudo-code is:

```
Clear h[][][][]
For Δw =-3; Δw <= 3; Δw ++
        For Δl =-3; Δl <= 3; Δl ++
                For hs=30; hs <= 50; hs++
                   For vs=183; vs <= 187; vs++
                        For each edge point (u1,v1) , find (x, y) by Eq. (3-1)(3-2)
                             Update(-x,y, Δw , Δl ,hs,vs);
                             Update(x,-y, Δw , Δl ,hs,vs);
                             Update(-x,-y, Δw , Δl ,hs,vs);
                        End
                    End
                End
        End
End

Function Update(x,y, Δw , Δl ,hs,vs)
Find (u,v) given (x,y) by Eq. (3-3)
If an edge point (tu,tv) within a window of (u,v)
      Then h[ Δw ][ Δl ][hs][vs]+=1/distance((tu,tv), (u,v) );
```

The ideal value for vs is CameraHeight-ImageWidth/2. (exactly 185 in our synthetic data).

When there is camera tilt and the camera is at the center of the table, we can us the tilt camera model in Section 2.3. Without lost of generality, we can add constraints $\omega \in [0,2\pi], \alpha > 0$.

$$x' = \sin\omega(x\sin\omega - y\cos\omega)(1-\cos\alpha) + x\cos\alpha;$$

$$y' = -\cos\omega(x\sin\omega - y\cos\omega)(1-\cos\alpha) + y\cos\alpha;$$

$$z' = (x\cos\omega + y\sin\omega) * \sin\alpha;$$

Given edge point (u,v) and tilt direction ω and tilt angle α, we want to solve (x, y).

(x', y', z') lies on the line determined by $(0,0,h)$ and $(r\cos\phi, r\sin\phi, \text{Im\_}y)$, so we have $(x', y', z') = t(0,0,h) + (1-t) * (r\cos\phi, r\sin\phi, \text{Im\_}y); t < 0$. That is

$$\begin{bmatrix} \sin^2\omega(1-\cos\alpha)+\cos\alpha & -\sin\omega\cos\omega(1-\cos\alpha) & r\cos\phi \\ -\sin\omega\cos\omega(1-\cos\alpha) & \cos^2\omega(1-\cos\alpha)+\cos\alpha & r\sin\phi \\ \cos\omega\sin\alpha & \sin\omega\sin\alpha & \text{Im\_}y - h \end{bmatrix} * \begin{bmatrix} x \\ y \\ t \end{bmatrix} = \begin{bmatrix} r\cos\phi \\ r\sin\phi \\ \text{Im\_}y \end{bmatrix} \quad (3\text{-}4)$$

The mapping from $(-x, y), (x,-y), (-x,-y)$ to image coordinate is straightforward.

We can first voting for optimal $(\Delta w, \Delta l, hs.vs)$, then vote for optimal tilt parameters (ω, α) given the assumption that the tilt does not affect the orientation and position so much. We can also vote all of them in a 6-layer loop with higher computational expense.

When the camera center is at $(\Delta w, \Delta l)$, with the same tilt, $(x, y, 0)$ will move to

$$x' = (x\sin^2\omega + \Delta w\cos^2\omega - (y-\Delta l)*\sin\omega\cos\omega)(1-\cos\alpha) + x\cos\alpha;$$

$$y' = (-\cos\omega\sin\omega(x-\Delta w) + y\cos^2\omega + \Delta l\sin^2\omega)(1-\cos\alpha) + y\cos\alpha;$$

$$z' = ((x-\Delta w)*\cos\omega + (y-\Delta l)*\sin\omega)*\sin\alpha;$$

$$(x', y', z') = t(\Delta w, \Delta l, h) + (1-t)*(r\cos\phi, r\sin\phi, \mathrm{Im\_}\, y); t < 0$$

$$\begin{bmatrix} \sin^2\omega(1-\cos\alpha)+\cos\alpha & -\sin\omega\cos\omega(1-\cos\alpha) & r\cos\phi-\Delta w \\ -\sin\omega\cos\omega(1-\cos\alpha) & \cos^2\omega(1-\cos\alpha)+\cos\alpha & r\sin\phi-\Delta l \\ \cos\omega\sin\alpha & \sin\omega\sin\alpha & \mathrm{Im\_}\, y-h \end{bmatrix} * \begin{bmatrix} x \\ y \\ t \end{bmatrix} = \begin{bmatrix} r\cos\phi-(\Delta w\cos\omega+\Delta l\sin\omega)\cos\omega(1-\cos\alpha) \\ r\sin\phi-(\Delta w\cos\omega+\Delta l\sin\omega)\sin\omega(1-\cos\alpha) \\ \mathrm{Im\_}\, y+(\Delta w\cos\omega+\Delta l\sin\omega)\sin\alpha \end{bmatrix}$$

$$d = \sqrt{(x'-\Delta w)^2 + (y'-\Delta l)^2}$$

$$\phi = \arctan\left(\frac{y'-\Delta l}{x'-\Delta w}\right) + cond - hs;$$

$$\mathrm{Im\_}\, y = h - \frac{r(h-z')}{d}$$

3.2. Point Filtering

We can use the result of the symmetry voting to filter some noisy points out. For an edge point, we can find its three symmetrical points using the camera parameters. If there are n window centered at its symmetry points that include other edge points ($0 \le n \le 3$), we add this edge point into ($n+1$)-fold image. For an n-fold image, the remaining points have higher confidence that belong to a true table edge, so we can further add all three symmetrical points of all existing points and acquire an n-fold-flip image. Our following fitting algorithm will be based on the 3-fold-flip image because generally n=3 can achieve best tradeoff between noise elimination and table edge preservation.

Fig. 3-[3~8] is an example of a test image.

Fig. 3-3: Original Panorama Image

Fig. 3-4: Edge Image Extracted from original image

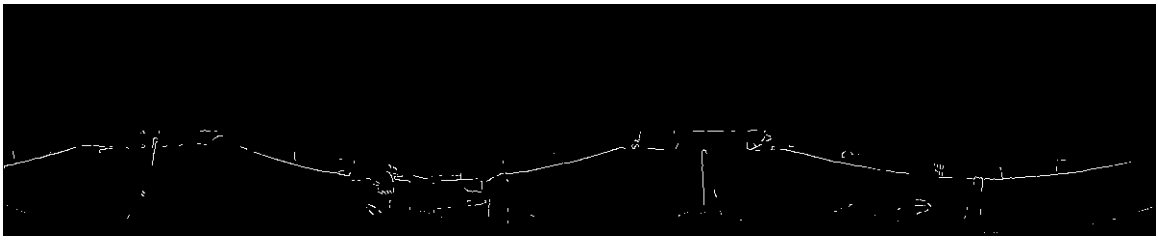

Fig. 3-5: 2-fold image after filtering

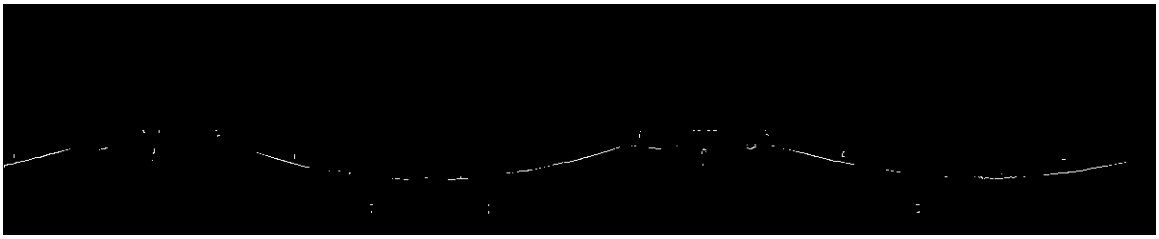

Fig. 3-6: 3-fold image after filtering

Fig. 3-7: 4-fold image after filtering

Fig. 3-8: 3-fold-flip image

4. Quadratic Fitting

4.1. Basic Algorithm

We can also use two quadratic lines to fit the 3-fold-flip image. Since this image is bilateral symmetry, we only preserve the half of the points (Fig. 4-1), that is $\phi = \frac{u}{\mathrm{Im}age\_width} + hs \in [\frac{\pi}{2}, \frac{3\pi}{2}]$, where u is x coordinate of an edge point.

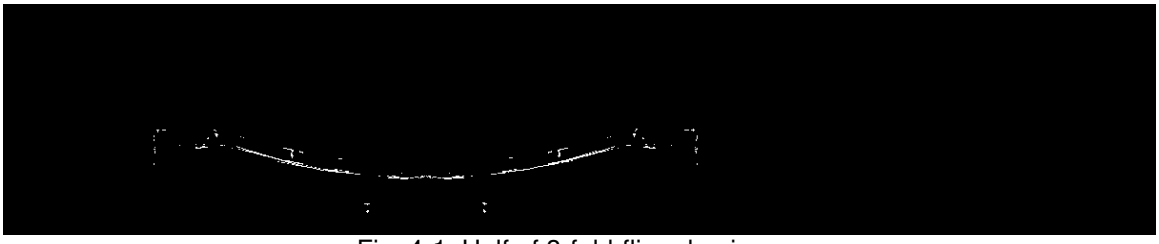

Fig. 4-1: Half of 3-fold-flip edge image

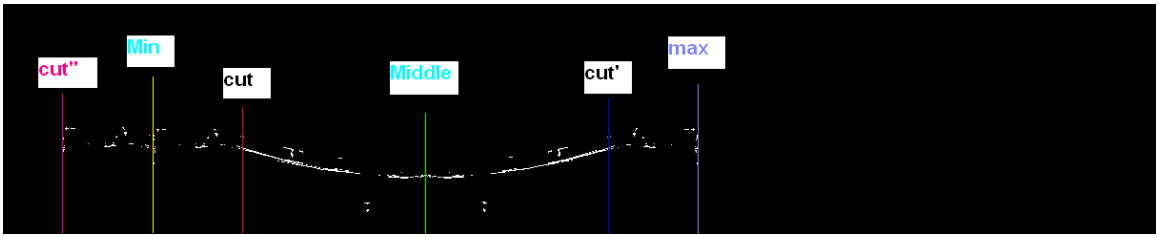

Fig. 4-2: Half of 3-fold-flip edge image for quadratic fitting

The following are some definition in Fig. 4-2

$$\min = (\frac{\pi}{2} - hs) * \mathrm{Im}age\_width/(2\pi);$$

$$\max = (\frac{3\pi}{2} - hs) * \mathrm{Im}age\_width/(2\pi);$$

$$middle = (\min + \max)/2;$$

cut corresponds to the connecting point of the two quadratic curves. Physically, it should be the projection of the table corner. ***cut'*** is the symmetry point of ***cut***. Since the points between ***min*** and ***cut***, ***cut'*** and ***max***, correspond to short table edge, and the points between ***cut*** and ***cut'*** correspond to long table edge. So $cut \in [\min, (\min + middle)/2]$.

The idea is that fit quadratic curve 1 centered at ***min*** to points within [***cut"***, ***cut***] and the quadratic curve 2 centered at ***middle*** to points within [***cut***, ***cut'***]. There are two constraints: the two quadratic curves must be continuous at ***cut***; quadratic curve 2 must be convex.

Assume edge points are categorized into two kinds after flipping points within [***cut'***, ***max***] to [***cut"***, ***cut***]:

$(u_i, v_i), i = 1,...,n.\ u_i \in [cut", cut]$. For fitting $y = a_1 + a_2(x - \min)^2$.

$(u_j, v_j), j = 1,...,m.\ u_j \in [cut, cut']$. For fitting $y = b_1 + b_2(x - middle)^2$.

We want to minimize

$$R = \sum_{i=1}^{n} (v_i - a_1 - a_2(u_i - \min)^2)^2 + \sum_{j=1}^{m} (v_j - b_1 - b_2(v_j - middle)^2)^2 \quad \text{w.r.t.} \quad a_{1,2}, b_{1,2} \quad \text{with}$$

constraint $S = a_1 + a_2(cut - \min)^2 - b_1 - b_2(cut - middle)^2 = 0$. (4-1)

By using Lagrange multiplier, we have

$$\frac{\partial(R - \lambda S)}{\partial a_i} = 0, \frac{\partial(R - \lambda S)}{\partial b_j} = 0, i, j = 1,2.$$

$$\begin{bmatrix} n & \sum_{i=1}^{n}(u_i-\min)^2 & 0 & 0 & 1 \\ \sum_{i=1}^{n}(u_i-\min)^2 & \sum_{i=1}^{n}(u_i-\min)^4 & 0 & 0 & (cut-\min)^2 \\ 0 & 0 & m & \sum_{j=1}^{m}(u_j-middle)^2 & -1 \\ 0 & 0 & \sum_{j=1}^{m}(u_j-middle)^2 & \sum_{j=1}^{m}(u_j-middle)^4 & (cut-middle)^2 \\ 1 & (cut-\min)^2 & -1 & (cut-middle)^2 & 0 \end{bmatrix} * \begin{bmatrix} a_1 \\ a_2 \\ b_1 \\ b_2 \\ \lambda \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} v_i \\ \sum_{i=1}^{n} v_i(u_i-\min)^2 \\ \sum_{j=1}^{m} v_j \\ \sum_{j=1}^{m} v_j(u_j-middle)^2 \\ 0 \end{bmatrix}$$

We calculate the median of square error: $(a_1 + a_2(u_i - \min)^2 - v_i)^2$ and $(b_1 + b_2(u_j - middle)^2 - v_j)^2$, $i = 1,...n; j = 1,...,m.$

We try all possible cut within $[\min,(\min + middle)/2]$, the one with least median error wins. Fig. 4-[3~6] are example of quadratic fitting with different level of losing edge points. We can see there will be large error when we lose most of the points of the long table edge. This problem may be fixed by trigonometry fitting which will be talked in Section 5.

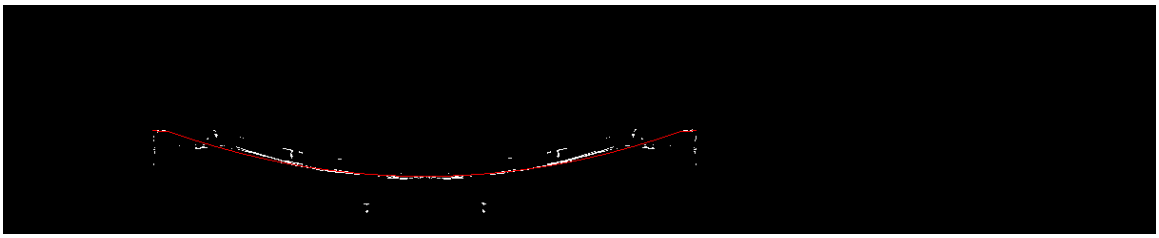

Fig. 4-3: Two quadratic curves fitting with least median square error

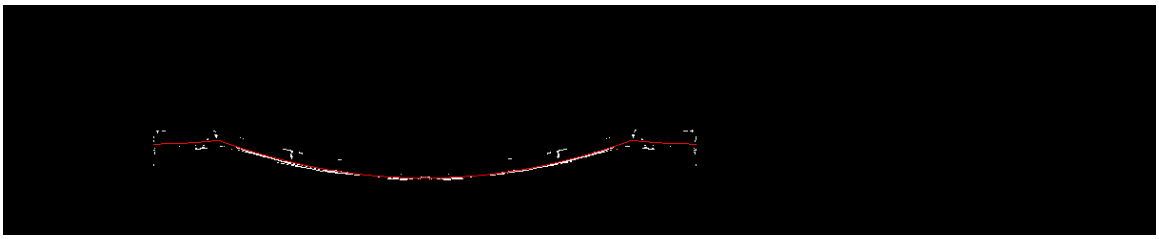

Fig. 4-4: Two quadratic curves fitting with least median square error for points above 200

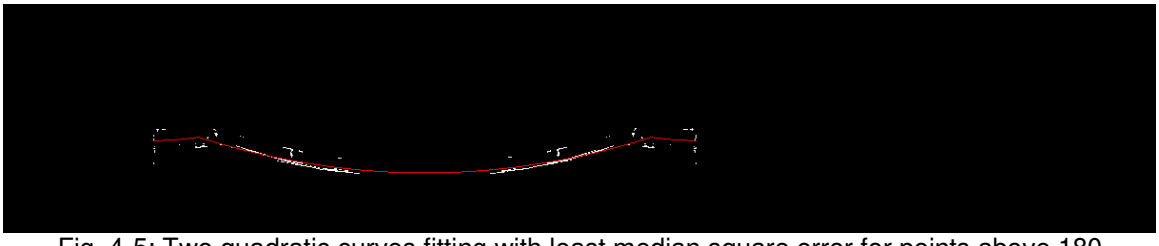

Fig. 4-5: Two quadratic curves fitting with least median square error for points above 180

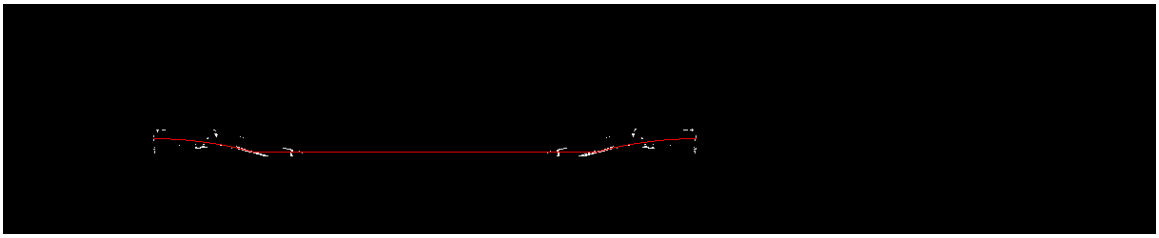

Fig. 4-6: Two quadratic curves fitting with least median square error for points above 160

4.2. Constraints of Quadratic Fitting

There are constraints in quadratic fitting that can help us tackle the fitting with partial data. Let us consider the boundary of b_2 in Section 4.1. Figure 4-7 illustrates that we use five points on the projection of the long table edge to fit a quadratic curve.

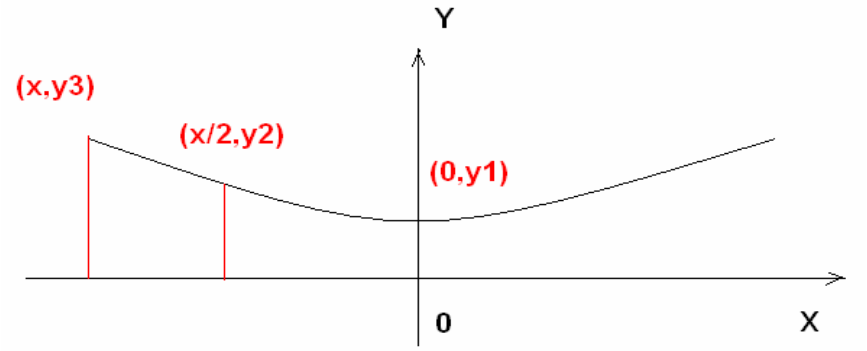

Fig. 4-7: The Long table edge

$$\frac{W}{\sqrt{W^2+L^2}} = t$$

$$x = \arccos(t) * \mathrm{Im}age\_width/(2\pi)$$

$$y_1 = h(1-\frac{r}{W});$$

$$y_2 = h(1-\frac{r\cos(\arccos(t)/2)}{W}) = h-\frac{hr}{W}\sqrt{\frac{t+1}{2}}$$

$$y_3 = h(1-\frac{rt}{W})$$

The best quadratic fitting is $y = b_1 + b_2 x^2; b_1 = y_1$

We want to minimize

$$R = (y_1 + b_2 x^2 - y_3)^2 + (y_1 + b_2 x^2/4 - y_2)^2$$

$$\frac{\partial R}{\partial b_2} = 0,$$

$$b_2 = \frac{(4y_3 + y_2 - 5y_1) * 4}{17x^2} = \frac{16hr\pi^2}{17W(\mathrm{Im}age\_width^2)} * \frac{5-\sqrt{\frac{t+1}{2}}-4t}{(\arccos(t))^2}$$

Since $\frac{W}{L} < [\frac{1}{4}, \frac{1}{1}], t \in (\frac{1}{\sqrt{17}}, \frac{\sqrt{2}}{2}), W \in [1.5\,feet, 3\,feet]$. We can get the boundary of b_2 easily. (Practically $[lb, hb]$=[0.9042e-3,1.9784e-3], when $r = 160, \text{Im}age\_width = 1005, h = 305$).

The upper boundary of a_2 can be estimated in a similar way.

$x = \arcsin(t) * \text{Im}age\_width/(2\pi)$

$$y_1 = h(1 - \frac{r}{L});$$

$$y_2 = h(1 - \frac{r\sin((\frac{\pi}{2} + \arccos(t))/2)}{L}) = h(1 - \frac{r}{\sqrt{2}L}(\sqrt{\frac{t+1}{2}} + \sqrt{\frac{1-t}{2}}))$$

$$y_3 = h(1 - \frac{rt}{W}) = h(1 - \frac{r\sqrt{1-t^2}}{L})$$

The best quadratic fitting is $y = a_1 + a_2 x^2; a_1 = y_1$

We want to minimize

$R = (y_1 + a_2 x^2 - y_3)^2 + (y_1 + a_2 x^2/4 - y_2)^2$

$$\frac{\partial R}{\partial a_2} = 0,$$

$$a_2 == \frac{(4y_3 + y_2 - 5y_1) * 4}{17x^2} = \frac{16hr\pi^2}{17L(\text{Im}age\_width^2)} * \frac{5 - 4\sqrt{1-t^2} - \frac{\sqrt{1+t} + \sqrt{1-t}}{2}}{(\arcsin(t))^2}$$

Since $\frac{W}{L} < [\frac{1}{4}, \frac{1}{1}], t \in (\frac{1}{\sqrt{17}}, \frac{\sqrt{2}}{2}), W \in [1.5\,feet, 12\,feet]$. We can get the upper boundary of a_2 easily. (Practically $ha \in$ [2.4858e-4,2.0739e-3], when $r = 160, \text{Im}age\_width = 1005, h = 305$).

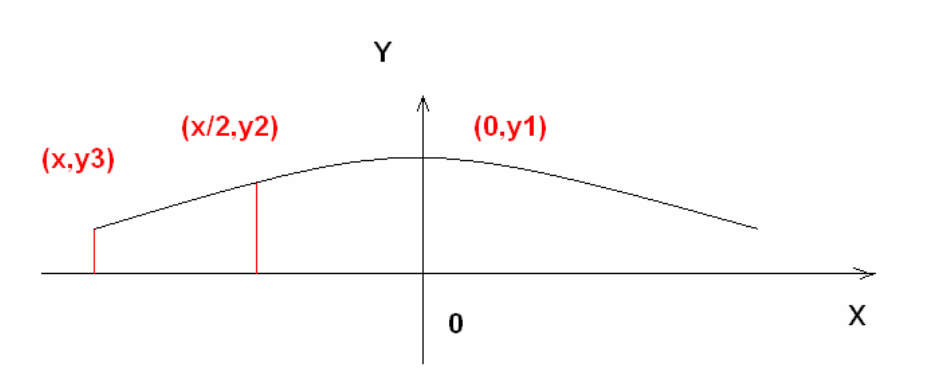

Fig. 4-8: The round end short table edge

The estimation of the lower boundary of a_2 need to consider various table shape. We use the round end table (Section 2.4.2) to estimate the upper boundary of a_2 (Fig. 4-8).

$$x = \arcsin(t) * \mathrm{Im}age\_width/(2\pi)$$

$$y_1 = h(1 - \frac{r}{W + L});$$

$$y_2 = h(1 - \frac{r}{\sqrt{W^2 + L^2 + \sqrt{2} * W * L}})$$

$$y_3 = h(1 - \frac{rt}{W})$$

The best quadratic fitting is $y = a_1 + a_2 x^2; a_1 = y_1$

We want to minimize

$$R = (y_1 + a_2 x^2 - y_3)^2 + (y_1 + a_2 x^2 / 4 - y_2)^2$$

$$\frac{\partial R}{\partial a_2} = 0,$$

$$a_2 == \frac{(4y_3 + y_2 - 5y_1) * 4}{17x^2} = \frac{16hr\pi^2}{17W(\mathrm{Im}age\_width^2)} * \frac{5t^2 - 4t - \frac{1}{\sqrt{\frac{1}{t^2} + \sqrt{2(\frac{1}{t^2} - 1)}}}}{(\arcsin(t))^2}$$

Since $\frac{W}{L} < [\frac{1}{4}, \frac{1}{1}], t \in (\frac{1}{\sqrt{17}}, \frac{\sqrt{2}}{2}), W \in [1.5 feet, 3 feet]$. We can get the lower boundary of a_2 easily. (Practically $la \in$ [-1.4479e-2,-7.5591e-4], when $r = 160, \mathrm{Im}age\_width = 1005, h = 305$).

So we combine the result and set the boundary of a_2 as $[la, ha]$=[-1.4479e-2, 2.0739e-3].

4.3. Fitting with Constraints

Now we need to solve optimization problem Eq. 3-1 with constraints: $a_2 \in [la, ha]$ and $b_2 \in [lb, hb]$.

We modify the fitting algorithm as following:

First, Search the $a_{1,2}, b_{1,2}$ as the algorithm in Section 3.1.

If a_2 or b_2 are out of boundary, we need to consider the following four cases:

I. Set $b_2 = lb$, the optimization becomes

We want to minimize

$$R = \sum_{i=1}^{n} (v_i - a_1 - a_2(u_i - \min)^2)^2 + \sum_{j=1}^{m} (v_j - b_1 - b_2(v_j - middle)^2)^2 \quad \text{w.r.t.} \quad a_{1,2}, b_1 \quad \text{with}$$

constraint $S = a_1 + a_2(cut - \min)^2 - b_1 - b_2(cut - middle)^2 = 0$.

By using Lagrange multiplier, we have

$$\frac{\partial(R - \lambda S)}{\partial a_i} = 0, \frac{\partial(R - \lambda S)}{\partial b_j} = 0, i = 1,2; j = 1;$$

$$\begin{bmatrix} n & \sum_{i=1}^{n}(u_i-\min)^2 & 0 & 1 \\ \sum_{i=1}^{n}(u_i-\min)^2 & \sum_{i=1}^{n}(u_i-\min)^4 & 0 & (cut-\min)^2 \\ 0 & 0 & m & -1 \\ 1 & (cut-\min)^2 & -1 & 0 \end{bmatrix} * \begin{bmatrix} a_1 \\ a_2 \\ b_1 \\ \lambda \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} v_i \\ \sum_{i=1}^{n} v_i(u_i-\min)^2 \\ \sum_{j=1}^{m} v_j - b_2(u_j-middle)^2 \\ b_2(cut-middle)^2 \end{bmatrix}$$

If a_2 meets the constraints, we save the mean square error. Otherwise, we set $a_2 = la$ or ha. Then

$$\begin{bmatrix} n & 0 & 1 \\ 0 & m & -1 \\ 1 & -1 & 0 \end{bmatrix} * \begin{bmatrix} a_1 \\ b_1 \\ \lambda \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} v_i - a_2(u_i-\min)^2 \\ \sum_{j=1}^{m} v_j - b_2(u_j-middle)^2 \\ -a_2(cut-\min)^2 + b_2(cut-middle)^2 \end{bmatrix}$$

The selection of ***cut*** is the same with Section 3.1. We select the ***cut*** with least mean square error and save it.

II. Same with case I, except that we set $b_2 = hb$.

III. Set $a_2 = la$, the optimization becomes

We want to minimize

$R = \sum_{i=1}^{n}(v_i - a_1 - a_2(u_i-\min)^2)^2 + \sum_{j=1}^{m}(v_j - b_1 - b_2(v_j-middle)^2)^2$ w.r.t. $a_1, b_{1,2}$ with constraint $S = a_1 + a_2(cut-\min)^2 - b_1 - b_2(cut-middle)^2 = 0$.

By using Lagrange multiplier, we have

$$\frac{\partial(R-\lambda S)}{\partial a_i} = 0, \frac{\partial(R-\lambda S)}{\partial b_j} = 0, i = 1; j = 1,2;$$

$$\begin{bmatrix} m & \sum_{j=1}^{m}(u_j-middle)^2 & 0 & 1 \\ \sum_{j=1}^{m}(u_j-middle)^2 & \sum_{j=1}^{m}(u_j-middle)^4 & 0 & (cut-middle)^2 \\ 0 & 0 & n & -1 \\ 1 & (cut-middle)^2 & -1 & 0 \end{bmatrix} * \begin{bmatrix} b_1 \\ b_2 \\ a_1 \\ \lambda \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{m} v_j \\ \sum_{j=1}^{m} v_j(u_j-middle)^2 \\ \sum_{i=1}^{n} v_i - a_2(u_i-\min)^2 \\ a_2(cut-\min)^2 \end{bmatrix}$$

If b_2 meets the constraints, we save the mean square error. Otherwise, we set $b_2 = lb$ or hb. The selection of ***cut*** is the same with Section 3.1. We select the ***cut*** with least mean square error and save it.

IV. Same with case III, except that we set $a_2 = ha$.

We compare the error in these four cases and choose the best one as our fitting result.

Fig. 4-[8~10] is the result of quadratic fitting with constraints. We can see that the fitting improved a lot in the case of partial data.

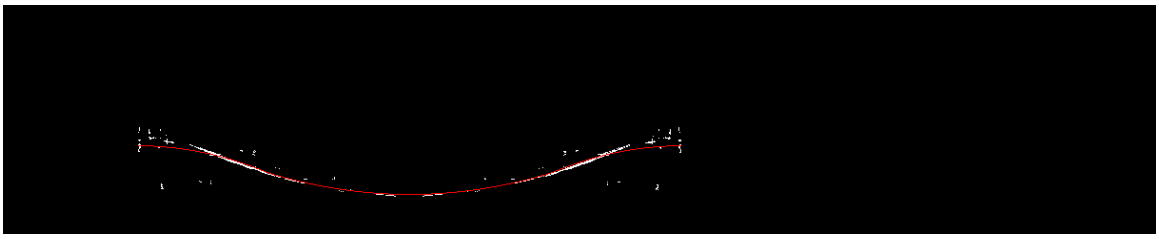

Fig. 4-8: Constrained quadratic curves fitting with least mean square error

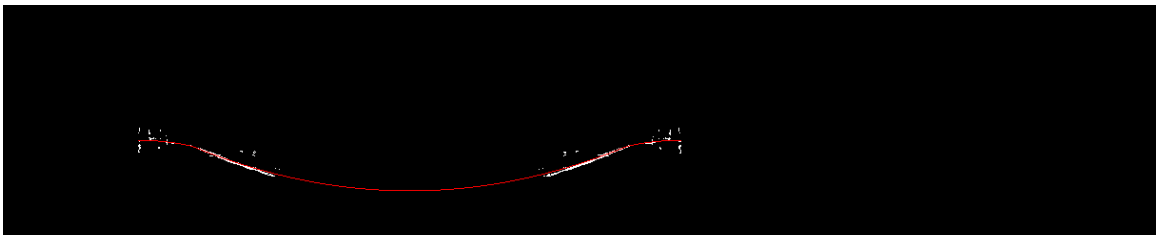

Fig. 4-9: Constrained quadratic curves fitting with least mean square error for points above 180

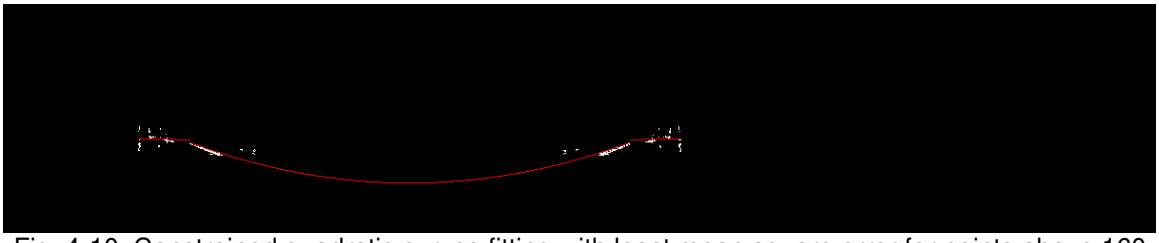

Fig. 4-10: Constrained quadratic curves fitting with least mean square error for points above 160

5. Trigonometry Fitting

5.1. Algorithm

The perfect edge of a triangular table is composed of four trigonometry curves. So we can use trigonometry curves for fitting based on the similar idea in Section 4.

The curve within [2****min-cut***, ***min***] is $x \in [-W, W], y = L.$

$$x' = (x\sin^2\omega + \Delta w\cos^2\omega - (y-\Delta l)*\sin\omega\cos\omega)(1-\cos\alpha) + x\cos\alpha;$$

$$y' = (-\cos\omega\sin\omega(x-\Delta w) + y\cos^2\omega + \Delta l\sin^2\omega)(1-\cos\alpha) + y\cos\alpha;$$

$$z' = ((x-\Delta w)*\cos\omega + (y-\Delta l)*\sin\omega)*\sin\alpha;$$

$$\phi = \arctan(\frac{y'-\Delta l}{x'-\Delta w}) + cond - hs = \frac{\text{Im}\_ x}{\text{Im}age\_width};$$

$$\gamma = \arctan(\frac{y'-\Delta l}{x'-\Delta w});$$

$$d = \sqrt{(x'-\Delta w)^2 + (y'-\Delta l)^2} = \frac{L-\Delta l}{\sin\gamma}$$

$$z' = (\frac{\cos\gamma}{\sin\gamma}\cos\omega + \sin\omega)(L-\Delta l)\sin\alpha = \frac{\cos(\gamma-\omega)\sin\alpha(L-\Delta l)}{\sin\gamma}$$

$$\text{Im}\_ y = h - \frac{r(h-z')}{d} = h(1-\frac{r\sin\gamma}{L-\Delta l}) + r\cos(\gamma-\omega)\sin\alpha$$

$$\gamma \in [\arctan\frac{L-\Delta l}{W-\Delta w}, \pi - \arctan\frac{L-\Delta l}{W+\Delta w}]$$

$\gamma, \text{Im}\_ y$ can be calculated from (u,v) easily.

The curve within [***cut***, ***cut'***] is $y \in [-L, L], x = -W.$

$$d = \sqrt{(x'-\Delta w)^2 + (y'-\Delta l)^2} = \frac{-W-\Delta w}{\cos\gamma}$$

$$z' = (\frac{\sin\gamma}{\cos\gamma}\sin\omega + \cos\omega)(-W-\Delta w)\sin\alpha = \frac{\cos(\gamma-\omega)\sin\alpha(-W-\Delta w)}{\cos\gamma}$$

$$\text{Im}\_ y = h - \frac{r(h-z')}{d} = h(1+\frac{r\cos\gamma}{W+\Delta w}) + r\cos(\gamma-\omega)\sin\alpha$$

$$\gamma \in [\pi - \arctan\frac{L-\Delta l}{W+\Delta w}, \pi + \arctan\frac{L+\Delta l}{W+\Delta w},]$$

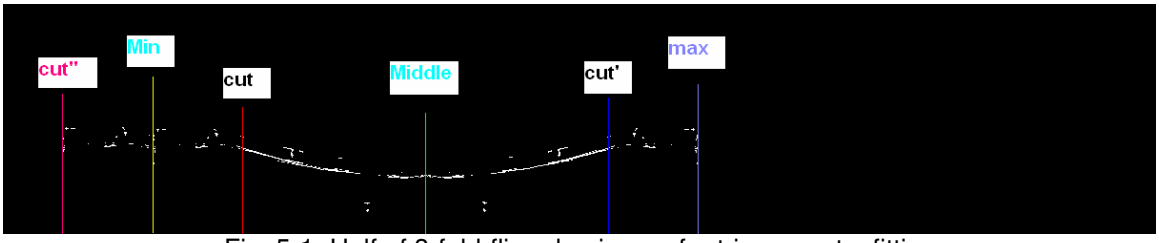

Fig. 5-1. Half of 3-fold-flip edge image for trigonometry fitting

So in Fig. 5-1,

$$\min = (\frac{\pi}{2} - hs) * \text{Im}\ age\ \_\ width\ /(2\pi);$$

$$\max = (\frac{3\pi}{2} - hs) * \text{Im}\ age\ \_\ width\ /(2\pi);$$

$$middle = (\min + \max)\ /\ 2;$$

$$cut = (\pi - \arctan\frac{L-\Delta l}{W+\Delta w} - hs) * \text{Im}\ age\ \_\ width\ /(2\pi);$$

$$cut' = (\pi + \arctan\frac{L+\Delta l}{W+\Delta w} - hs) * \text{Im}\ age\ \_\ width\ /(2\pi);$$

$$cut'' = (\arctan\frac{L-\Delta l}{W-\Delta w} - hs) * \text{Im}\ age\ \_\ width\ /(2\pi);$$

The differences between $d_1 = \min - cut''; d_2 = cut - \min; d_3 = \max - cut';$ are actually small. So we just assume $d_1 = d_2 = d_3$ in fitting.

$k = 2\pi / \mathrm{Im}age\_width;$

$$k(d_2 - d_1) = \arctan\frac{L-\Delta l}{W-\Delta w} - \arctan\frac{L-\Delta l}{W+\Delta w} = \arctan\frac{2\Delta w(L-\Delta l)}{W^2 - \Delta w^2 + (L-\Delta l)^2} \approx \frac{2\Delta wL}{W^2+L^2}$$

$$k(d_2 - d_3) = \arctan\frac{L+\Delta l}{W+\Delta w} - \arctan\frac{L-\Delta l}{W+\Delta w} = \arctan\frac{2\Delta l(W+\Delta w)}{(W+\Delta w)^2 + L^2 - \Delta l^2} \approx \frac{2\Delta lW}{W^2+L^2}$$

When we try all possible $cut \in [\min, (\min + middle)/2]$,

$$ratio = \frac{L-\Delta l}{W+\Delta w} = \tan(\frac{\pi}{2} - (cut - \min) * k);$$

Assume edge points are categorized into two kinds after flipping points within [***cut"***, ***cut']*** :

$(u_i, v_i), i = 1,...,n.\ u_i \in [cut'', cut]$. For fitting

$$\mathrm{Im}\_y_i = h(1 - \frac{r\sin\gamma_i}{L-\Delta l}) + r\cos(\gamma_i - \omega)\sin\alpha = h + r\cos(\gamma_i - \omega)\sin\alpha - \frac{1}{ratio} * \frac{hr}{(W+\Delta w)}\sin\gamma_i.$$

$(u_j, v_j), j = 1,...,m.\ u_j \in [cut, cut']$. For fitting

$$\mathrm{Im}\_y_j = h(1 + \frac{r\cos\gamma_j}{W+\Delta w}) + r\cos(\gamma_j - \omega)\sin\alpha = h + r\cos(\gamma_j - \omega)\sin\alpha + \frac{hr}{W+\Delta w}\cos\gamma_j.$$

Where $\dfrac{u}{\mathrm{Im}age\_width} + hs = \gamma;$

$$v + vs = \mathrm{Im}\_y;$$

We want to minimize R w.r.t. $t = \dfrac{hr}{W+\Delta w}$

$$R = \sum_{i=1}^{n}(h + r\cos(\gamma_i - \omega)\sin\alpha - \frac{t}{ratio}\sin\gamma_i - \mathrm{Im}\_y_i)^2 + \sum_{j=1}^{m}(h + r\cos(\gamma_j - \omega)\sin\alpha + t\cos\gamma_j - \mathrm{Im}\_y_j)^2$$

$$\frac{\partial R}{\partial t} = 0$$

$$t = \frac{\sum_{i=1}^{n}\frac{\sin\gamma_i}{ratio}(h + r\cos(\gamma_i - \omega)\sin\alpha - \mathrm{Im}\_y_i) - \sum_{j=1}^{m}\cos\gamma_j(h + r\cos(\gamma_j - \omega)\sin\alpha - \mathrm{Im}\_y_j)}{\sum_{i=1}^{n}\frac{\sin^2\gamma_i}{ratio^2} + \sum_{j=1}^{m}\cos^2\gamma_j}$$

The ***cut*** with minimum least mean square error will be chosen as the best fitting. Fig. 5-(2~6) are some results. We can see the trigonometry fitting performs very well even though most of the points at the bottom are removed.

Note that we use least mean square error instead of least median square error here for more robust performance.

Fig. 5-2. Edge Image

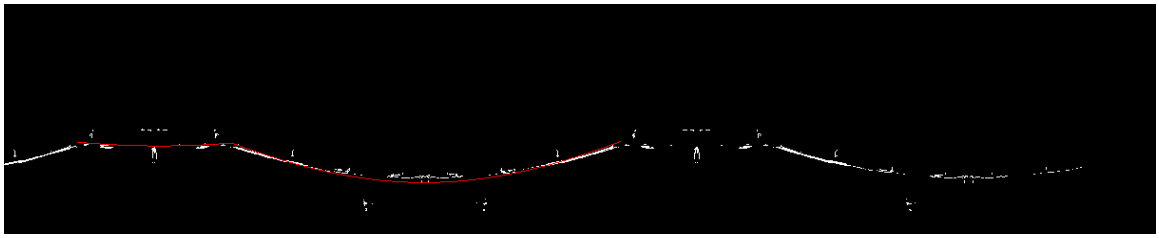

Fig. 5-3. two trigonometry curves fitting with least mean square error

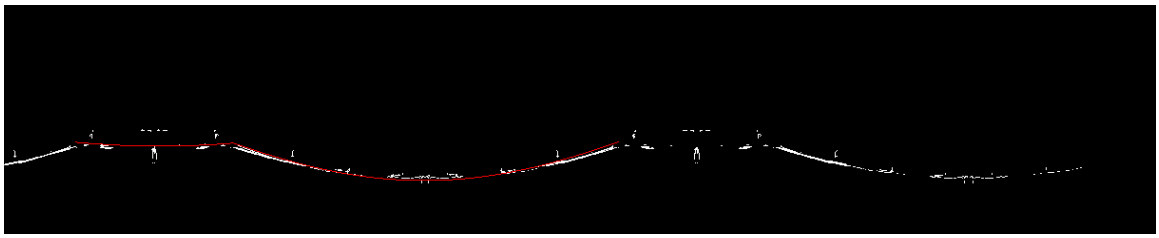

Fig. 5-4. two trigonometry curves fitting with least mean square error for points above 200

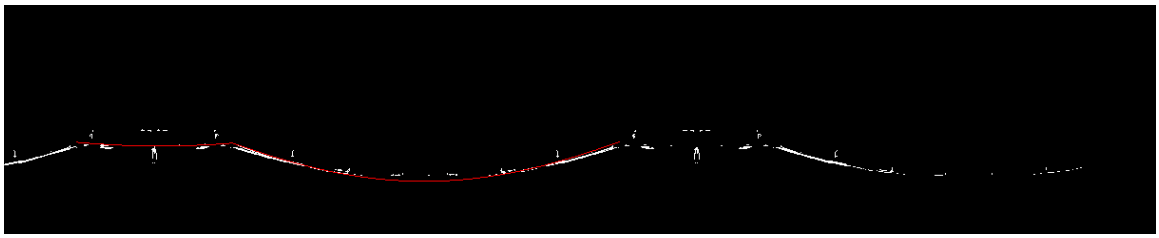

Fig. 5-5. two trigonometry curves fitting with least mean square error for points above 180

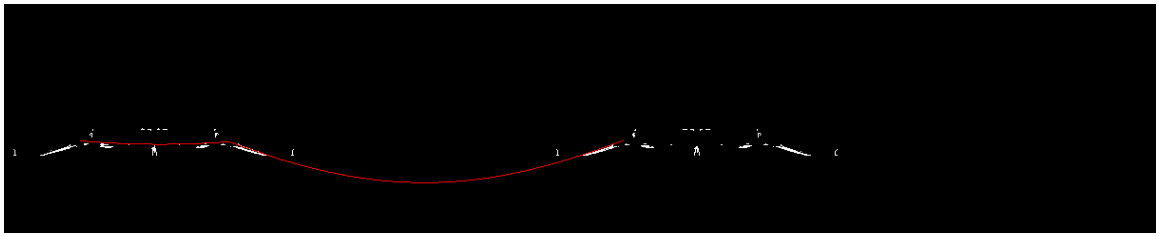

Fig. 5-6. two trigonometry curves fitting with least mean square error for points above 160

5.2. Comparison of quadratic and trigonometry fitting on real images

There are both advantage and disadvantages for the two kinds of fitting. Quadratic fitting can handle various table shapes. But current quadratic fitting assumes horizontal symmetry, so it will not have perfect fitting when there is large camera tilt. It may be improved by a generic quadratic fitting after the current algorithm. Trigonometry fitting

works well on rectangular table when there is large camera tilt and data missing. But it degrades when the table is not rectangular.

We have three sets of experiments on real data. The first is on a cluttered table. The second is on a clean table. We learn the table width/length ratio and fix the ***cut*** position. The third is on the clean table with manually added block noise. We will compare their performance under different noise level.

Fig. 5-(7~11) is the results of experiment set #1. The outputs of symmetry voting are $\Delta w = 0, \Delta l = 3, \omega = 252^o, \alpha = 2.6^o$. We can see the quadratic fitting cannot capture the tilted edges. And both methods made error on table corner position because of the noise.

Fig. 5-7. Original image of cluttered table

Fig. 5-8. General edge image

Fig. 5-9. Filtered edge points after symmetry voting

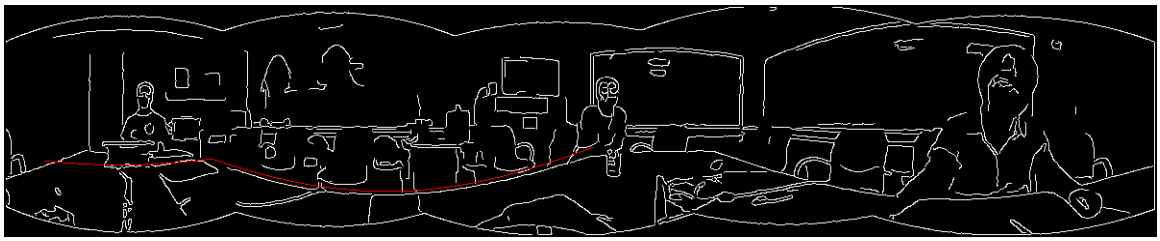

Fig. 5-10. Trigonometry fitting

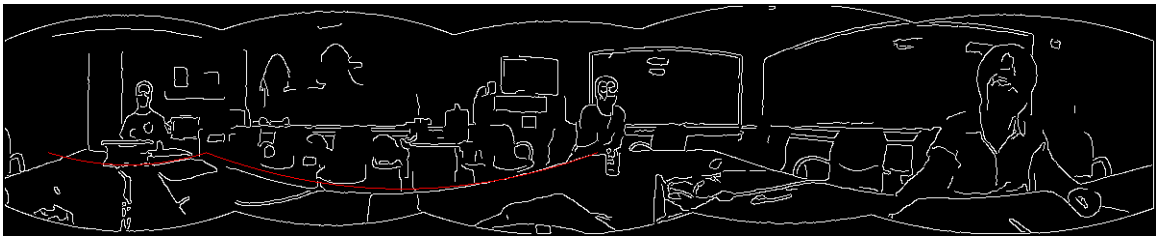

Fig. 5-11. Quadratic fitting

Fig. 5-(12~14) is the results of experiment set #2. We use the ***cut*** position learned from quadratic fitting in Fig. 5-13 to fit the noisy image in experiment set #3.

Fig. 5-12: Original Image

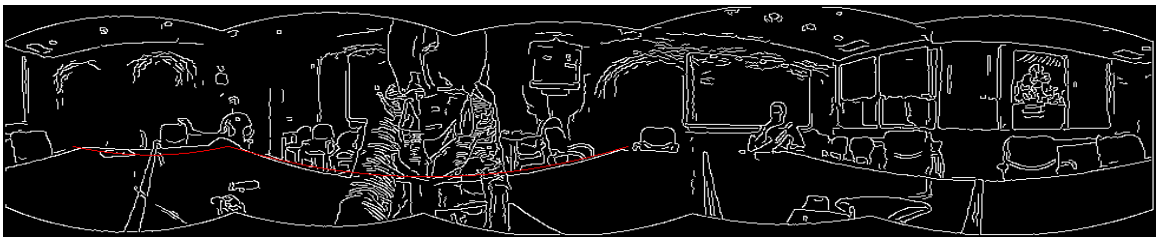

Fig. 5-13: Quadratic fitting on original image

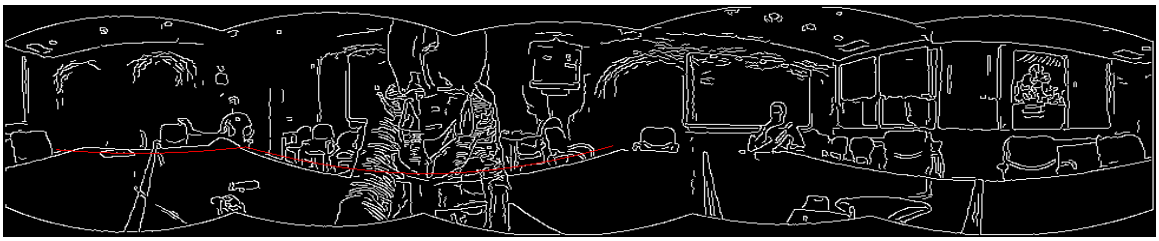

Fig. 5-14: Trigonometry fitting on original image

Fig. 5-(15~23) is the results of experiment set #3. We can see that the trigonometry fitting keeps satisfying performance with learned table width/length ratio even though the image is highly corrupted. The reason is that given camera height h and focus r, there are only two parameters (W,L) needed to be estimated in the trigonometry fitting after we apply the camera parameters (hs, ω, α) from symmetry voting. But there are five parameters (a_1, a_2, b_1, b_2, cut) for the quadratic fitting. After we fix the ***cut***, we can get the table width/length ratio directly. There is only one parameter left for the trigonometry fitting. But there are still four parameters for the quadratic fitting. We have added constraints on a_2, b_2 based on rectangular table model. Based on the same knowledge,

$a_1 = h(1-\frac{r}{L}), b_1 = h(1-\frac{r}{W})$. So we can add more constraints on quadratic fitting and expect better.

Fig. 5-15: Original image with noise N=50

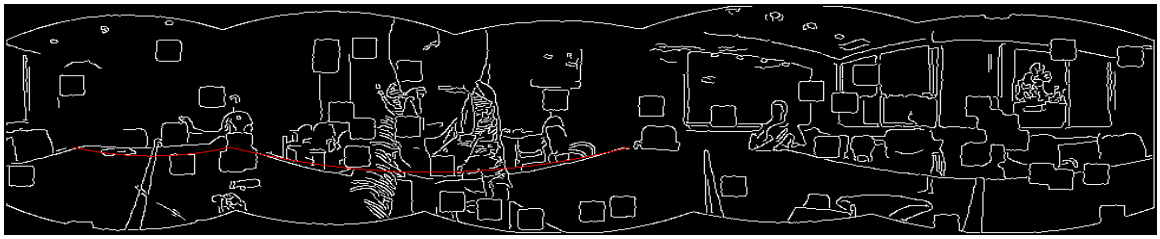

Fig. 5-16: Quadratic fitting on noise N=50

Fig. 5-17: Trigonometry fitting on noise N=50

Fig. 5-18: Original image with noise N=100

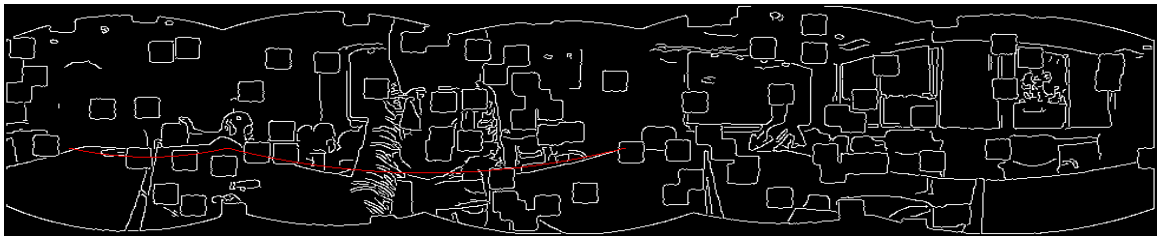

Fig. 5-19: Quadratic fitting on noise N=100

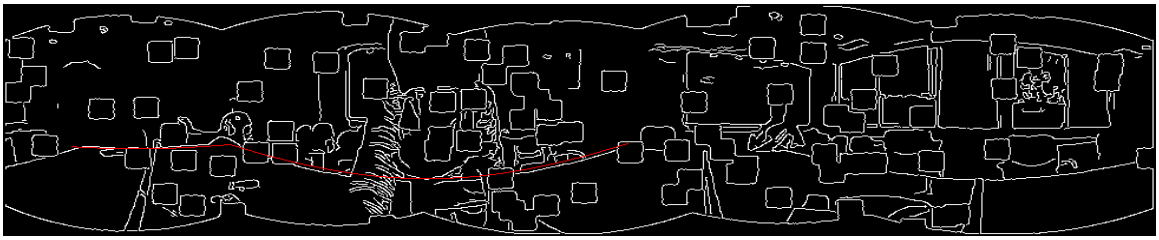

Fig. 5-20: Trigonometry fitting on noise N=100

Fig. 5-21: Original image with noise N=200

Fig. 5-22: Quadratic fitting on noise N=200

Fig. 5-23: Trigonometry fitting on noise N=200

6. During Conference: ICP

To determine the table edges, there are actually six parameters we need to estimate: the table size (W, L); the horizontal shift hs (determined by the camera orientation and where we cut the cylinder); the vertical shift vs (determined by the camera height and vertical fov); and the camera position $(\Delta w, \Delta l)$. Fig. 6-1 is the fitting found manually.

Fig. 6-1: A match found manually

We talk about the algorithm for estimating table size and camera position in this section. The details for ICP and function $ROV(\ )$ will be presented in Section 5.1 and 5.2.

Goal: Estimate parameters $(W, L, hs, vs, \Delta w, \Delta l)$

First Step: Forward mapping. We choose sizes of several common conference tables, estimate $(hs, vs, \Delta w, \Delta l)$ with initialization set $(W_i, L_i, hs_i, vs_i, \Delta w_i, \Delta l_i), i = 1,...,n.$ by using the ICP on the long edges in Section 6. The ICP will converge to $(W_i^*, L_i^*, hs_i^*, vs_i^*, \Delta w_i^*, \Delta l_i^*), i = 1,..., n.$

Second Step: Verification.

1. $k = \arg\max_i ROV(W_i^*, L_i^*, hs_i^*, vs_i^*, \Delta w_i^*, \Delta l_i^*)$
2. Fix W as W_k^*.
3. Estimate $(hs, vs, \Delta w, \Delta l)$ with initialization set $(W, L_j, hs_k^*, vs_k^*, \Delta w_k^*, \Delta l_k^*)$, $j = 1,...,m.$ by using the ICP on the long edges in Section 3. The ICP will converge to $(W, L_j^*, hs_{k,j}^{**}, vs_{k,j}^{**}, \Delta w_{k,j}^{**}, \Delta l_{k,j}^{**}), j = 1,...,m.$
4. $s = \arg\max_j ROV(W, L_j^*, hs_{k,j}^{**}, vs_{k,j}^{**}, \Delta w_{k,j}^{**}, \Delta l_{k,j}^{**})$.
5. Fix L as L_s^*. Return $(W_k^*, L_s^*, hs_{k,s}^{**}, vs_{k,s}^{**}, \Delta w_{k,s}^{**}, \Delta l_{k,s}^{**})$.

6.1. Forward Mapping

The "Iterative Closest Point" (ICP) method [6] can be used for this problem. Each time, we search the nearest points with proper orientation given some estimation. These points are used to update the three parameters. In our implementation, we just use the long table edges because the short edges are usually not accurate and occluded.

6.1.1. Camera at the center of table

Give a set of points $(x_i, y_i), i = 1,...n.$ from the Eq. 2-1, and a set of points $(x_j, y_j), j = 1,...m.$ from the Eq. 2-3. We have

$$y_i = h(1 - \frac{r\cos\theta}{W}) + H = -vscale * \cos(c * x_i + hs) + vs$$
$$= -vscale * (\cos(c * x_i)\cos(hs) - \sin(c * x_i)\sin(hs)) + vs$$
$$= -A * \cos(c * x_i) + B * \sin(c * x_i) + D$$

$$y_j = h(1+\frac{r\cos\theta}{W}) = vscale * \cos(c * x_j + hs) + vs$$
$$= vscale * (\cos(c * x_j)\cos(hs) - \sin(c * x_j)\sin(hs)) + vs$$
$$= A * \cos(c * x_j) - B * \sin(c * x_j) + D$$

Where (x, y) is the coordinates of table edges in images. c is a constant to scale $\theta \in [0, 2\pi]$ to $x \in [0, \mathrm{Im}ageWidth]$. H is a constant related to panorama cropping.

$vscale = r * h / W;$ $\quad vs = h + H;$ $\quad D = vs;$

$A = vscale * \cos(hs);$ $\quad B = vscale * \sin(hs);$

We want to minimize the square error;

$$R = \sum_i (-A * \cos(c * x_i) + B * \sin(c * x_i) + D - y_i)^2 + \sum_j (A * \cos(c * x_j) - B * \sin(c * x_j) + D - y_j)^2$$

So we need to solve the equations:

$\left\{\frac{\partial R}{\partial A} = 0; \frac{\partial R}{\partial B} = 0; \frac{\partial R}{\partial D} = 0\right.$, that is

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} * \begin{bmatrix} A \\ -B \\ D \end{bmatrix} = \begin{bmatrix} -\sum_i y_i * \cos(c * x_i) + \sum_j y_j \cos(c * x_j) \\ \sum_i y_i * \sin(c * x_i) - \sum_j y_j \sin(c * x_j) \\ \sum_i y_i + \sum_j y_j \end{bmatrix} \qquad (6\text{-}1)$$

$$a_{1,1} = \sum_i \cos^2(c * x_i) + \sum_j \cos^2(c * x_j); a_{2,2} = \sum_i \sin^2(c * x_i) + \sum_j \sin^2(c * x_j); a_{3,3} = n + m;$$

Where $a_{1,2} = a_{2,1} = \sum_i \sin(c * x_i) * \cos(c * x_i) + \sum_j \sin(c * x_j) * \cos(c * x_j);$

$$a_{1,3} = a_{3,1} = -\sum_i \cos(c * x_i) + \sum_j \cos(c * x_j); a_{2,3} = a_{3,2} = \sum_i \sin(c * x_i) - \sum_j \sin(c * x_j);$$

From Eq. 6-1 we can calculate the updated $[A \quad B \quad D]^T$, that can be easily transformed into $[hs \quad vs \quad vscale]$.

After several iterations, we will be able to arrive at a local optimal result. Fig. 6-2 is one example after ten iterations.

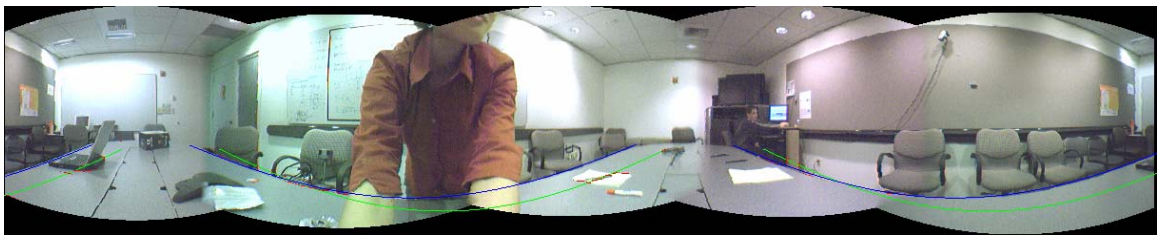

Fig. 6-2: Green lines are the original estimate. Blue lines are the search result. Spare red points are the nearest edge points found in the first iteration.

6.1.2. Camera not at the center of table

When the camera is not at table center, but $(\Delta w, \Delta l)$, the parameter estimation becomes more complex. Similarly, we get

$$y_i = h(1 - \frac{r\cos\theta}{W}(1 + \frac{\Delta w}{W})) + H = -\frac{hr}{W}(1 + \frac{\Delta w}{W}) * \cos(c * x_i + hs) + vs$$
$$= -(X + Z) * \cos(c * x_i) + (Y + V) * \sin(c * x_i) + D$$
$$y_j = h(1 + \frac{r\cos\theta}{W}(1 - \frac{\Delta w}{W})) + H = \frac{hr}{W}(1 - \frac{\Delta w}{W}) * \cos(c * x_j + hs) + vs$$
$$= (X - Z) * \cos(c * x_j) - (Y - V) * \sin(c * x_j) + D$$

Where

$$X = \frac{hr}{W}\cos(hs); \qquad Y = \frac{hr}{W}\sin(hs); \qquad Z = \frac{hr\Delta w}{W^2}\cos(hs);$$
$$V = \frac{hr\Delta w}{W^2}\sin(hs); \qquad D = h + H;$$

We want to minimize the square error:

$$R = \sum_i (-(X + Z) * \cos(c * x_i) + (Y + V) * \sin(c * x_i) + D - y_i)^2$$
$$+ \sum_j ((X - Z) * \cos(c * x_j) - (Y - V) * \sin(c * x_j) + D - y_j)^2 \qquad (6\text{-}2)$$

with constraint: $Z * Y - X * V = 0$.

Using Lagrange multiplier λ, we need to solve the equations:

$$\begin{cases} \frac{\partial(R + \lambda(Z * Y - X * V))}{\partial X} = 0; \frac{\partial(R + \lambda(Z * Y - X * V))}{\partial Y} = 0 \\ \frac{\partial(R + \lambda(Z * Y - X * V))}{\partial Z} = 0; \frac{\partial(R + \lambda(Z * Y - X * V))}{\partial W} = 0; \frac{\partial(R + \lambda(Z * Y - X * V))}{\partial D} = 0 \end{cases},$$

That is Eq (3-3):

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} - \lambda & a_{1,5} \\ a_{2,1} & a_{2,2} & a_{2,3} + \lambda & a_{2,4} & a_{2,5} \\ a_{3,1} & a_{3,2} + \lambda & a_{3,3} & a_{3,4} & a_{3,5} \\ a_{4,1} - \lambda & a_{4,2} & a_{4,3} & a_{4,4} & a_{4,5} \\ a_{5,1} & a_{5,2} & a_{5,3} & a_{5,4} & a_{5,5} \end{bmatrix} * \begin{bmatrix} X \\ Z \\ Y \\ V \\ D \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{bmatrix} \qquad (6\text{-}3)$$

Where

$$a_{1,1} = a_{2,2} = \sum_i \cos^2(c * x_i) + \sum_j \cos^2(c * x_j); a_{1,2} = a_{2,1} = \sum_i \cos^2(c * x_i) - \sum_j \cos^2(c * x_j);$$

$$a_{1,3} = a_{3,1} = a_{2,4} = a_{4,2} = -\sum_i \sin(c * x_i) * \cos(c * x_i) - \sum_j \sin(c * x_j) * \cos(c * x_j);$$

$$a_{1,4} = a_{4,1} = a_{2,3} = a_{3,2} = -\sum_i \sin(c * x_i) * \cos(c * x_i) + \sum_j \sin(c * x_j) * \cos(c * x_j);$$

$$a_{1,5} = a_{5,1} = -\sum_i \cos(c * x_i) + \sum_j \cos(c * x_j); a_{2,5} = a_{5,2} = -\sum_i \cos(c * x_i) - \sum_j \cos(c * x_j);$$

$$a_{3,3} = a_{4,4} = \sum_i \sin^2(c * x_i) + \sum_j \sin^2(c * x_j); a_{3,4} = a_{4,3} = \sum_i \sin^2(c * x_i) - \sum_j \sin^2(c * x_j);$$

$$a_{3,5} = a_{5,3} = \sum_i \sin(c * x_i) - \sum_j \sin(c * x_j); a_{4,5} = a_{5,4} = \sum_i \sin(c * x_i) + \sum_j \sin(c * x_j);$$

$$a_{5,5} = n + m;$$

$$c_1 = -\sum_i y_i * \cos(c * x_i) + \sum_j y_j \cos(c * x_j);$$

$$c_2 = -\sum_i y_i * \cos(c * x_i) - \sum_j y_j \cos(c * x_j);$$

$$c_3 = \sum_i y_i * \sin(c * x_i) - \sum_j y_j \sin(c * x_j);$$

$$c_4 = \sum_i y_i * \sin(c * x_i) + \sum_j y_j \sin(c * x_j);$$

$$c_5 = \sum_i y_i + \sum_j y_j$$

Eq. (6-2) and (6-3) are nonlinear, that can be solved by Newton's method. We rewrite Eq. (6-3) as

$$(A + \lambda B) * \begin{bmatrix} X \\ Z \\ Y \\ V \\ D \end{bmatrix} = C \text{ where } A = [a_{i,j}]_{5\times5}; C = [c_i]_{5\times1}; B = \begin{bmatrix} 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix};$$

Furthermore, we represent $A = [A_1, A_2, A_3, A_4, A_5]; B = [B_1, B_2, B_3, B_4, B_5]; A_i, B_j \in R^{5\times1}$.

So the nonlinear equations become

$$F(X, Z, Y, V, D, \lambda) = (A_1 + \lambda B_1) * X + (A_2 + \lambda B_2) * Z + (A_3 + \lambda B_3) * Y + (A_4 + \lambda B_4) * V + (A_5 + \lambda B_5) * D - C;$$

$$G(X, Z, Y, V, D, \lambda) = X * V - Z * Y$$

All the partial derivatives of F(…) and G(…) are linear. So given some seed $(X_0, Z_0, Y_0, V_0, D_0, \lambda_0)$, the updating equations are also linear:

$$\begin{cases} (A_1 + \lambda_0 B_1) * X + (A_2 + \lambda_0 B_2) * Z + (A_3 + \lambda_0 B_3) * Y + (A_4 + \lambda_0 B_4) * V + (A_5 + \lambda_0 B_5) * D \\ \quad + (B_1 * X_0 + B_2 * Z_0 + B_3 * Y_0 + B_4 * V_0 + B_5 * D_0) * \lambda = \\ \lambda_0 (B_1 * X_0 + B_2 * Z_0 + B_3 * Y_0 + B_4 * V_0 + B_5 * D_0) + C; \\ -V_0 * X + Y_0 * Z + Z_0 * Y - X_0 V = -X_0 V_0 + Z_0 Y_0; \end{cases}$$

That is one step in Newton's algorithm:

$$\begin{bmatrix} & & & -V_0 \\ & A+\lambda_0 B & & Y_0 \\ & & & Z_0 \\ & & & -X_0 \\ & & & 0 \\ -V_0 & Y_0 & Z_0 & -X_0 & 0 & 0 \end{bmatrix}_{6\times 6} * \begin{bmatrix} X \\ Z \\ Y \\ V \\ D \\ \lambda \end{bmatrix} = \begin{bmatrix} c_1 - \lambda_0 * V_0 \\ c_2 + \lambda_0 * Y_0 \\ c_1 + \lambda_0 * Z_0 \\ c_1 - \lambda_0 * X_0 \\ c_5 \\ -X_0 V_0 + Z_0 Y \end{bmatrix} \tag{6-4}$$

Fig. 6-3 is an example for this kind of scenarios. The algorithm succeeds to converge to the correct table edges after several iterations.

Fig. 6-3: Same setting with camera not at the center of the table.

But when the initialization shifts too much, the result can still be wrong. One possible solution is to use multiple seeds since we will discard most of the candidates any way during verification stage.

6.1.3. Short table edge detection

When the table is clear, we can detect the short table edge in the same way.

$$y_i = h(1 - \frac{r\sin\theta}{L}(1+\frac{\Delta l}{L})) + H = -\frac{hr}{L}(1+\frac{\Delta l}{L}) * \sin(c * x_i + hs) + vs$$

$$= -(X+Z) * \sin(c * x_i) + (Y+V) * \cos(c * x_i) + D$$

$$y_j = h(1 + \frac{r\cos\theta}{L}(1-\frac{\Delta l}{L})) + H = \frac{hr}{L}(1-\frac{\Delta l}{L}) * \sin(c * x_j + hs) + vs$$

$$= (X-Z) * \sin(c * x_j) - (Y-V) * \cos(c * x_j) + D$$

Where

$$X = \frac{hr}{L}\cos(hs); \qquad Y = \frac{hr}{L}\sin(hs); \qquad Z = \frac{hr\Delta l}{L^2}\cos(hs);$$

$$V = \frac{hr\Delta l}{L^2}\sin(hs); \qquad D = h + H;$$

We want to minimize the square error:

$$R = \sum_i (-(X+Z) * \sin(c * x_i) + (Y+V) * \cos(c * x_i) + D - y_i)^2$$

$$+ \sum_j ((X-Z) * \sin(c * x_j) - (Y-V) * \cos(c * x_j) + D - y_j)^2$$

with constraint: $Z * Y - X * V = 0$.

We can apply Lagrange multiplier and use Newton method to solve the problem as well.

6.2. Verification

Form staple.com and officedepot.com, we collect eleven kinds of the common conference table sizes. We apply the ICP algorithm in Section 2 with different table sizes and compare the rate of overlapping (ROV), which is defined as

$$\frac{\text{the number of the edge points within 1 pixel to the searched table edge}}{\text{the number of pixels on the searched table edge}}$$

Since we only detect long edges in Section 3, we approximate the short edges by connecting the end points of the long edges. The result is shown in table 6-1 and Fig 6-[4~7].

Fig. 6-4. table size #2

Fig. 6-5. table size #4

Fig. 6-6. table size #8

Fig. 6-7. table size #10

Table 6-1: ROV comparison for different table size

#	Table Size	L/W ratio	ROV (long edges)	ROV (short edges)
1	48"x144" (true)	3.5	64%	45%
2	48"x120"	2.5	65%	15%
3	48"x96"	2	48%	16%
4	36"x72"	2	41%	21%
5	36"x96"	2.67	38%	41%
6	42"x120"	2.85	60%	23%
7	36"x84"	2.3	39%	27%
8	30"x60"	2	30%	28%
9	44"x96"	2.18	59%	17%
10	24"x72"	3	25%	5%
11	45"x80"	1.78	57%	27%

Given camera height and focus, the "vertical" position of long edges is determined by the width of the table; and the "horizontal" position of long edges is determined by camera orientation. So with proper initialization, the table size with correct width will have high ROV for long edges.
The verification stage can be performed in two steps: the first step is to search the best match using table size #2, #4, #8, #10. Fix the width by the one with best ROV; the second step is to tune the table length and camera orientation if possible.

7. Testing on Synthetic Data

We use Direct3D to generate synthetic data for five cameras in real RingCam. The cameras' focuses are set as the same point precisely and their view directions are distributed around the center uniformly. The stitching from single perspective view to cylindrical view can be inferred from Section 1.1 and Fig, 6-1. Note that we need to scale the cylindrical view by $\frac{2*\sin(36^o)}{72*\pi/180}$ to keep the object have similar size in perspective mapping.

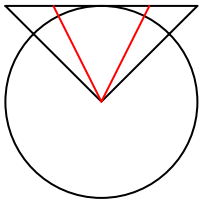

Fig. 7-1: From perspective mapping to cylindrical mapping

We test the camera tilt model by using the synthetic data. The experimental results on camera tilt model are in Table 6-1 and Fig. 6-(2~3). We set the step for ω is 2^o, for α is 0.2^o. The average error for ω is 5.14^o, for α is 0.0786^o. ($\Delta w, \Delta l$ also compensate for the error to some extend). We can see that ω is prone to have large error when α is small, while ω does not affect α that much. The symmetry voting works well for camera tilt detection.

Table 7-1: Camera tilt detection results

Correct Tilt Parameter($\Delta w = \Delta l = 0$)		Detected Camera parameters			
ω (degree)	α (degree)	ω (degree)	α (degree)	Δw (mm)	Δl (mm)
50	0.2	62	0.2	0	-3
100	0.2	120	0.2	0	-1
150	0.2	158	0.2	1	0
200	0.2	206	0.2	1	0
250	0.2	252	0.2	0	1
300	0.2	312	0.2	0	1
350	0.2	358	0.4	-3	0
50	0.4	44	0.6	-3	-3
100	0.4	106	0.4	0	-1
150	0.4	162	0.6	3	-3
200	0.4	208	0.6	3	3
250	0.4	252	0.4	0	1
300	0.4	308	0.4	-1	1
350	0.4	354	0.6	-3	0
50	0.6	48	0.6	-1	-1
100	0.6	98	0.6	0	-1
150	0.6	152	0.8	3	-3
200	0.6	202	0.8	3	1
250	0.6	250	0.6	0	1
300	0.6	304	0.6	-1	1
350	0.6	352	0.8	-3	1
50	1.4	48	1.4	-1	-1
100	1.4	106	1.4	3	-3
150	1.4	152	1.4	3	-2
200	1.4	200	1.6	3	1
250	1.4	246	1.6	3	3
300	1.4	298	1.4	0	0
350	1.4	350	1.6	-3	1

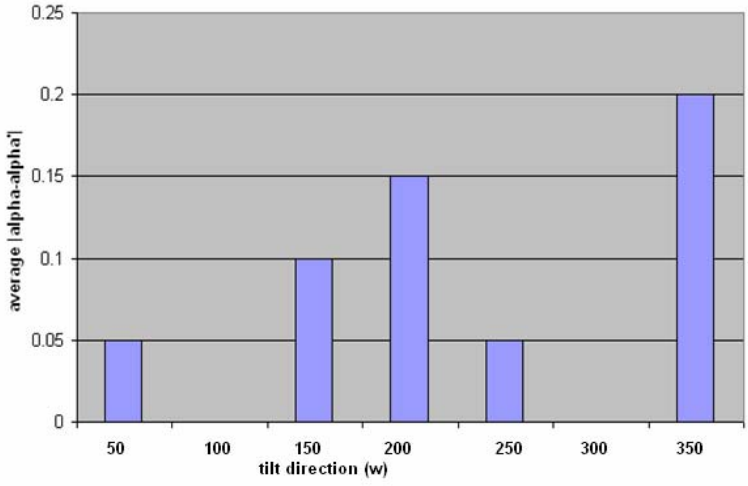

Fig. 7-2: error of α under different ω

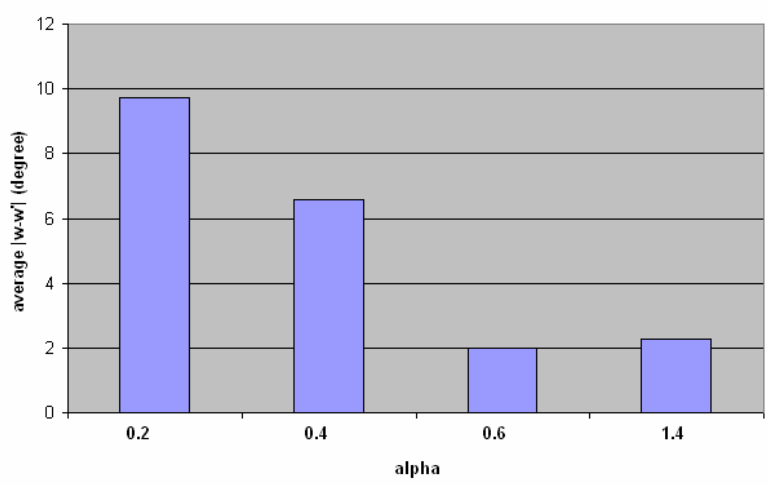

Fig. 7-3: error of ω under different α

We build the model for boat-shaped table (Section 2.4.1) and manually add random blocks to simulate noise. The noise level is controlled by the number of the blocks N. The results on camera tilt vs. noise vs. fitting methods are in Fig. 7-(4~16).

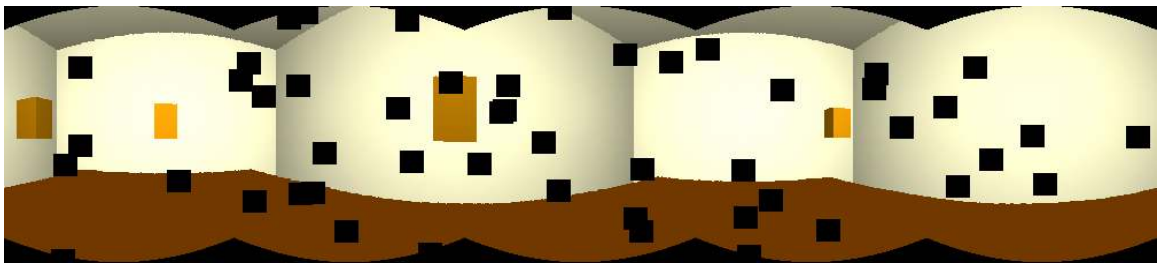
Fig. 7-4: Synthetic panorama image with N=50

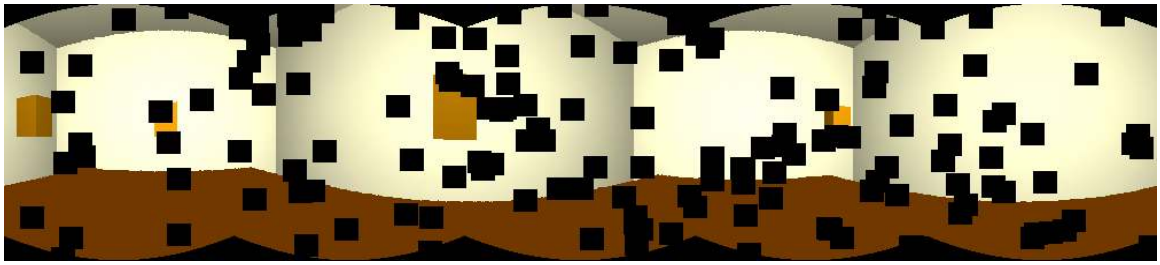
Fig. 7-5: Synthetic panorama image with N=100


Fig. 7-6: Synthetic panorama image with N=200

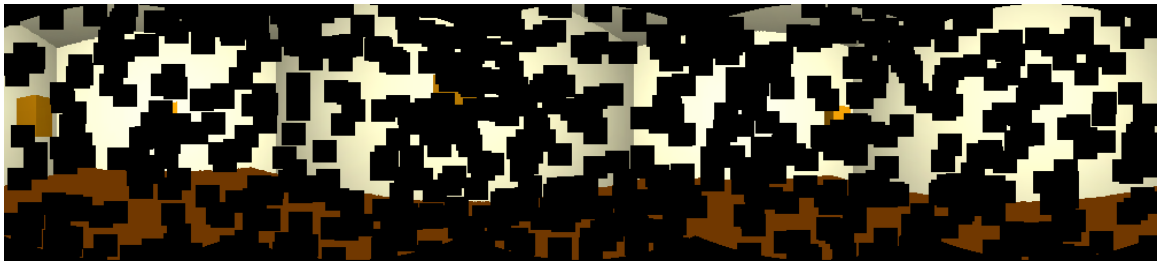

Fig. 7-7: Synthetic panorama image with N=300

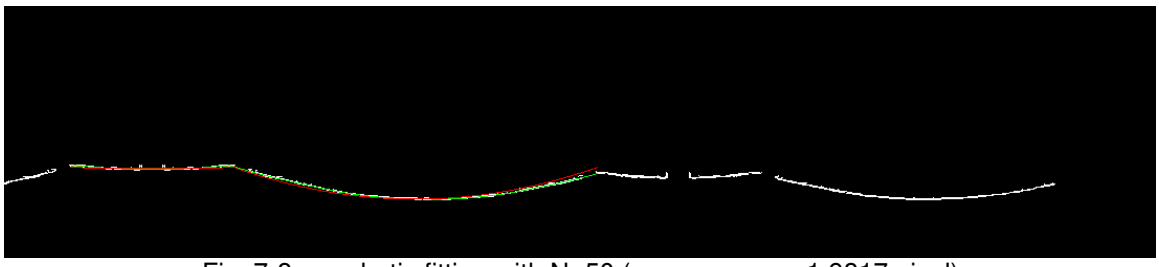

Fig. 7-8: quadratic fitting with N=50 (average error=1.9817 pixel)

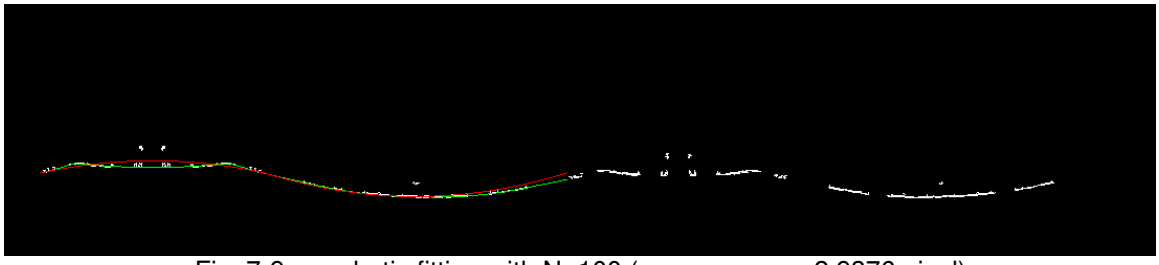

Fig. 7-9: quadratic fitting with N=100 (average error=2.9376 pixel)

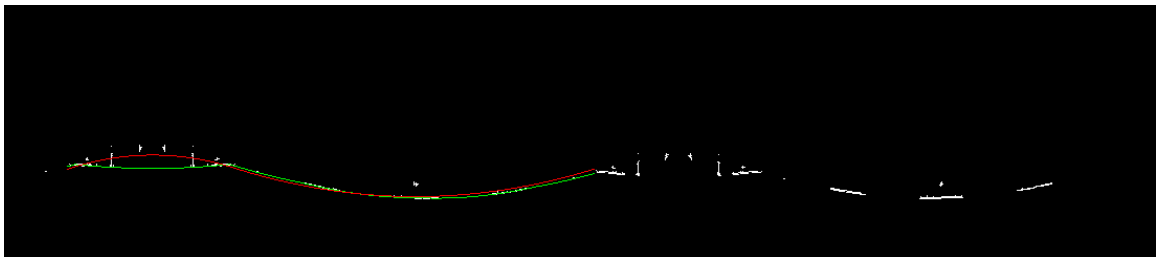

Fig. 7-10: quadratic fitting with N=200 (average error=5.2159 pixel)

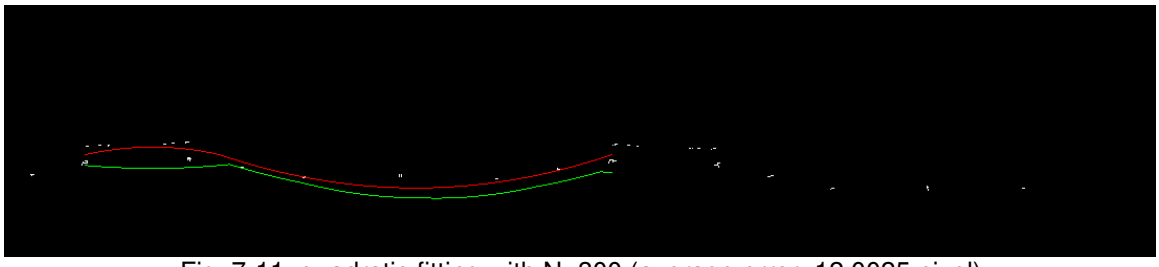

Fig. 7-11: quadratic fitting with N=300 (average error=12.0025 pixel)

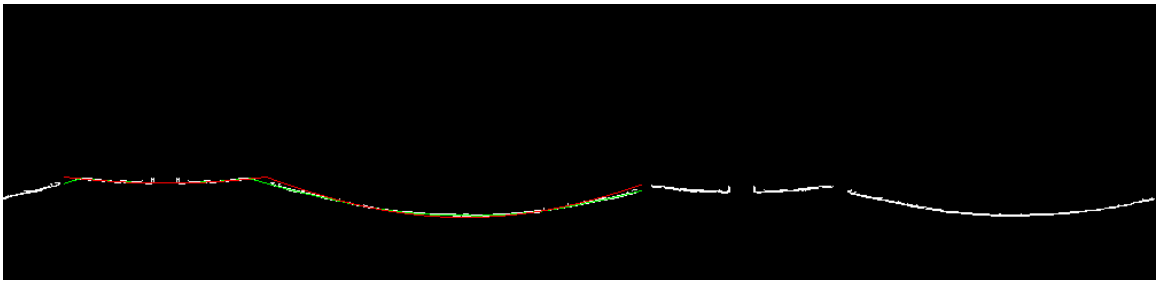

Fig. 7-12: trigonometry fitting with N=50 (average error=1.8649 pixel)

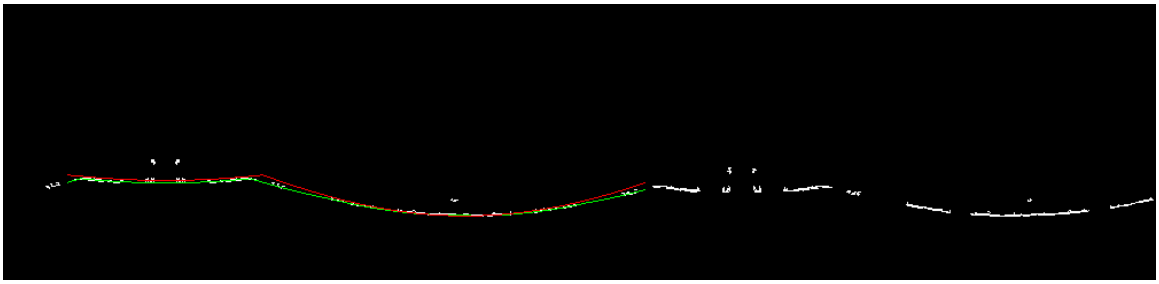

Fig. 7-13: trigonometry fitting with N=100 (average error=2.6267 pixel)

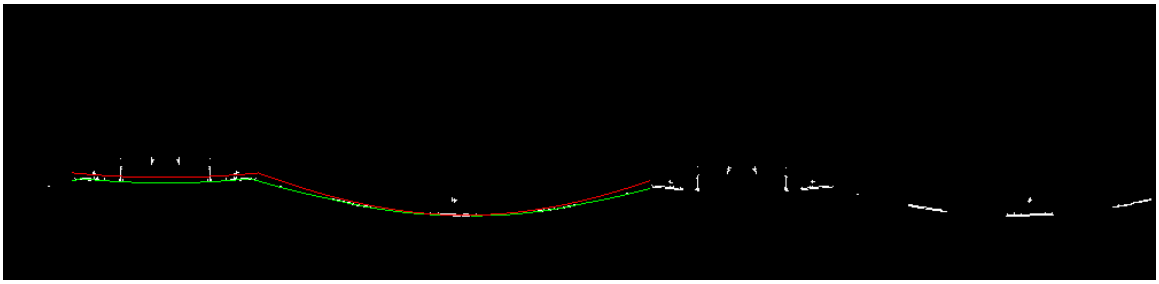

Fig. 7-14: trigonometry fitting with N=200 (average error=3.7321 pixel)

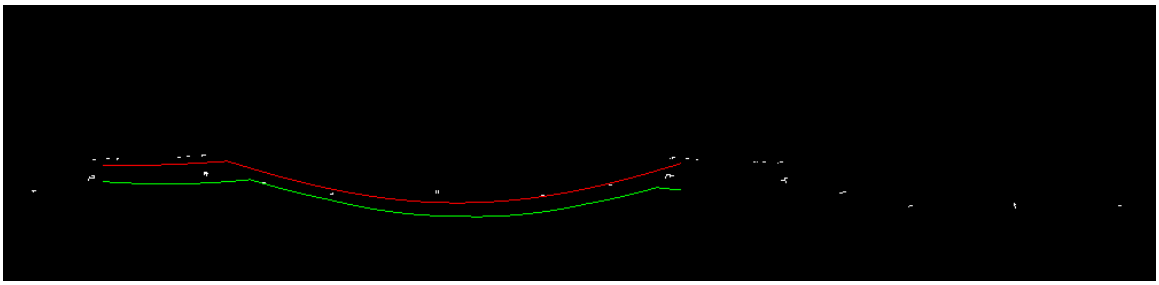

Fig. 7-15: trigonometry fitting with N=300 (average error=13.5782 pixel)

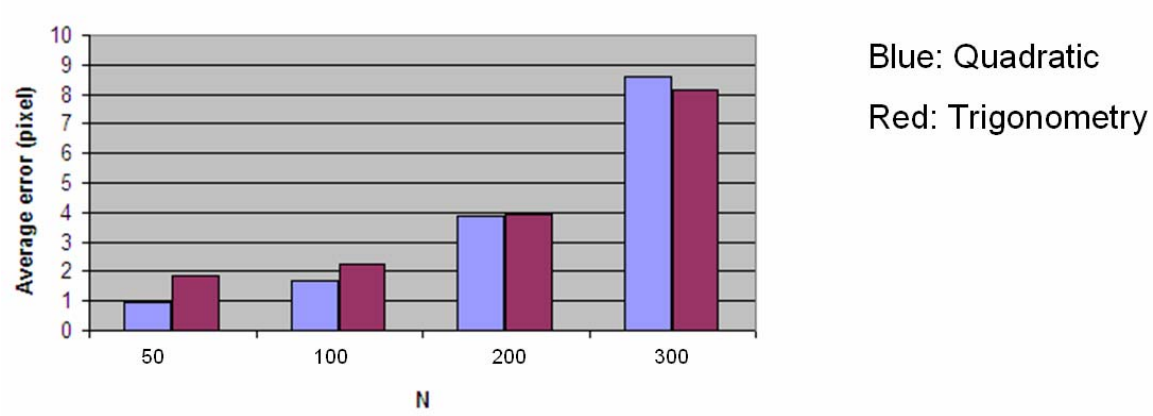

Fig. 7-16: Comparison of the average error

We can see that the quadratic fitting outperform trigonometry fitting on boat shaped table. But quadratic fitting can not catch large camera tilt.

We also fix the camera tilt ($\omega = 90^o, \alpha = 0.6^o$) and noise level (N=150), but change the table width/length ratio (1:2, 1;3, 1;4) of rectangular and boat shaped table. The experimental results are in Fig. 7-(17~19). The results show that the quadratic fitting and trigonometry fitting both work well (maximum error less than 2 pixels). The quadratic fitting perform slightly better than trigonometry fitting on boat table, but on the contrary on rectangular table with camera tilt.

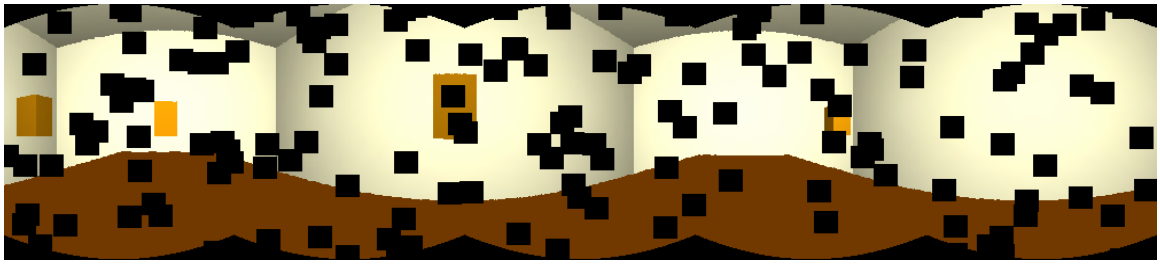
Fig. 7-17: Boat table (1:4) with noise N=150

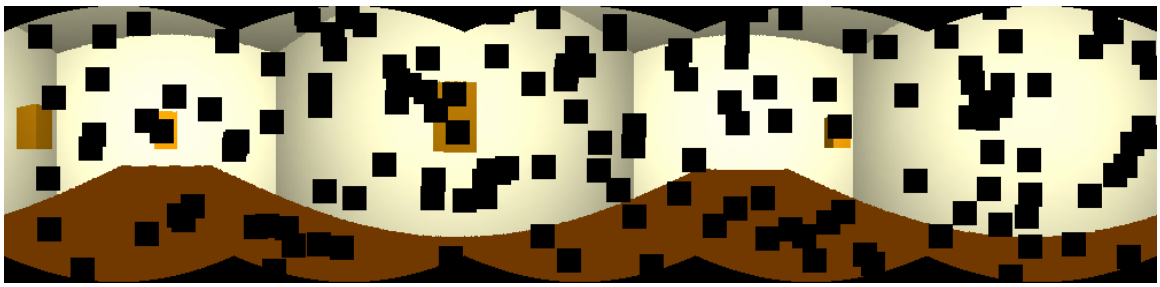
Fig. 7-18: Rectangular table (1:4) with noise N=150

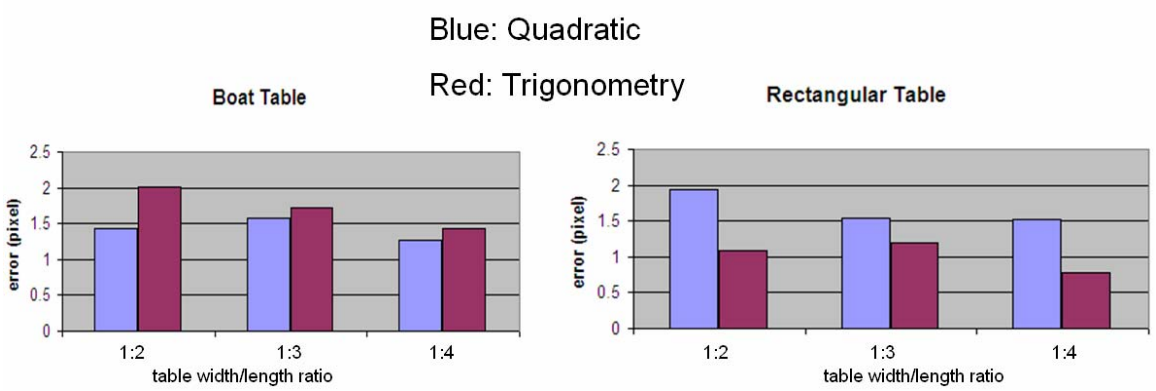

Fig. 7-19: Error comparison

8. Summary

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. We propose a first-learning-then-fitting scheme for robust detection. The corner position is determined by table width/length ratio, which can be learned more correctly when table is clean. In the fitting stage for normal usage, we set the width/length ratio and just looking for table width in optimization. At that time, the points from long table edge are more likely visible and can contribute to a correct solution even though the corner and short edges are occluded.

Symmetry voting can detect camera parameters and extract the table edges reliably. Both quadratic and trigonometry fitting work robustly under various camera/table parameters, noise and missing data.

We test the algorithm intensively and compare the two fitting methods. The experimental results on both real images and synthetic images support the effectiveness of our algorithm under various situations. Quadratic fitting can handle arbitrary table shapes. Trigonometry fitting is better at handling missing data and large camera tilt.

For future improvement, we may apply a generic quadratic fitting after the current one to handle large camera tilt and add more constraints, and speed up the computation for the real product.

Reference

[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, "Real-Time Warps for Improved Wide-Angle Viewing", MSR Technical Report TR-2002-110.
[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] http://www.research.microsoft.com/~zhang/
[7] P. Meer, B. Georgescu: "Edge detection with embedded confidence." IEEE Trans. Pattern Anal. Machine Intell., 23, 1351-1365, December 2001.