Local Optimization of Distortions in Wide-Angle Images Using Moving Least-Squares

Yoshihiro Kanamori* University of Tsukuba
Nguyen Huu Cuong† The University of Tokyo
Tomoyuki Nishita‡ The University of Tokyo

Abstract

We present a fast algorithm for reducing the undesired distortions (e.g., bending of straight lines) that often appear in wide-angle images especially taken with fisheye lenses. Our method allows the user to intuitively specify curves to be straightened by using quadratic control curves, and then automatically warps the image based on *moving least-squares (MLS)* while minimizing distortions in the vicinity of the curves. Unlike previous methods, our method provides results instantly, and allows the user to manipulate curves to obtain better results once after the initial solution is given. We demonstrate the effectiveness of our method with a variety of wide-angle images, together with comparisons to related projections.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation—Bitmap and framebuffer operations

Keywords: wide-angle images, image warping, moving-least squares

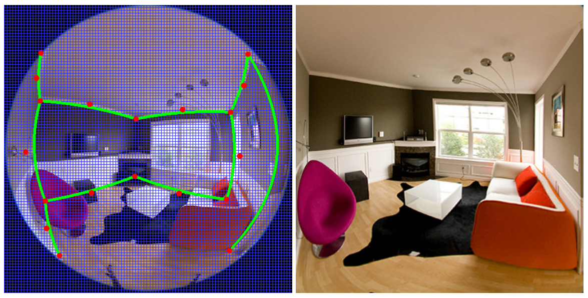


Figure 1: Left: Given a fisheye image (overlaid with a blue 100×100 grid used for computation), the user specifies a curve (green) as a constraint by clicking on two endpoints (red). The red point at the middle of each curve is a control point that defines the curve. Right: The result obtained within 41 seconds. Our system automatically gives an initial solution within a second, and then the user can manipulate the image in real time.

1 Introduction

Wide-angle images can provide rich visual information of a scene thanks to the wide field of view. Unlike what we perceive when viewing the world with our eyes, however, wide-angle images inevitably expose distortions such as bending of straight structures and stretching/shearing of objects. This is because any existing techniques that project a 3D scene into a 2D plane cannot avoid introducing distortions [Zorin and Barr 1995], and such distortions become more apparent as the field of view gets wider.

To better handle distortions in wide-angle images, several approaches have been proposed. Zorin and Barr [1995] presented a global projection that trades off between preservation of straight lines and circles. However, their method is not aware of the contents of the image. Zelnik-Manor et al. [2005] proposed a simple approach that separates a projection plane for linear perspective into multiple planes, and stitches them to synthesize an image. In their approach, however, the seams between projection planes must coincide with existing discontinuities in the scene, and curved linear structures passing over such seams cannot be straightened.

To preserve the shapes of salient objects, two contemporary approaches [Kopf et al. 2009; Carroll et al. 2009] allow the user to explicitly specify curved linear structures to be straightened. In Kopf et al.'s method [2009], wide-angle images are mapped onto a cylindrical surface, and constraints are specified as curvilinear polygons enclosing salient objects on the surface. Their system deforms the 3D cylinder so that the enclosed regions become flat, and then unfolds the deformed cylinder onto a 2D plane using isometric parameterization. Whereas Kopf et al.'s method forces the user to enclose objects, Carroll et al.'s method [2009] just lets the user specify curves along curved lines. Carroll et al.'s method maps the input image onto a *viewing sphere* centered at the viewpoint, and performs least-squares optimization on the spherical grid with respecting conformality and user-specified constraints. Both of the two methods are well formulated, but lack adequate interactivity; because the task of distortion reduction has no ground truth (exploring this based on, e.g., human perception, may open a new research field), the user requires trial and error. However, these two methods take a few [Kopf et al. 2009] or several ten [Carroll et al. 2009] seconds until the computation completes, every time the user modifies constraints.

In this paper, we present a fast method to assist the user in reducing undesired distortions in wide-angle images. With our intuitive interface similar to that of Carroll et al., the user can input curves as constraints to be straightened. The system heuristically straightens the curves and instantly warps the input image using *moving least-squares (MLS)* [Schaefer et al. 2006]. Although the initial solution often suffices, the user can explore better results in real time by manipulating constraints (see Figure 1). Technically, we solve local optimization problems for each grid point in a regular 2D grid (the blue grid in the left of Figure 1) instead of global optimization on cylindrical [Kopf et al. 2009] or spherical surfaces [Carroll et al. 2009], or a 2D grid as done in low-distortion texture mapping [Gal et al. 2006] and content-aware image resizing [Wolf et al. 2007; Wang et al. 2008]. The technical contribution of this paper lies in the derivation of the approximative closed-form solution of MLS warping with curves, which yields no visible difference compared to the reference result obtained via numerical integration.

*e-mail: kanamori@cs.tsukuba.ac.jp
†e-mail: huucuong@nis-lab.is.s.u-tokyo.ac.jp
‡e-mail: nis@is.s.u-tokyo.ac.jp

2 Our Method

2.1 Overview

Given an input image taken with a fisheye lens, our method reduces distortions in the image by straightening curved lines. The user specifies constraints by clicking on the two endpoints of each curved line to be straightened, and then our system computes and draws a curve along it. Our interface is similar to that of Carroll et al. [2009], but the algorithm is quite different; the constraint is represented with a quadratic Bézier curve (Section 2.2) laid on a planar grid, instead of a line on the viewing sphere. Given constraints, our system computes weights for MLS warping (Section 2.3) and then straightens the Bézier curves in a heuristic way (Section 2.4) to reduce salient distortions in the input image. If the initial solution does not suffice, the user can further warp the image by manipulating the Bézier curves in real time, which we consider important to provide the user instant feedback. The details are described in the following subsections.

2.2 Constraints for Curved Lines

We use parametric curves suited for warping to trace curved lines. Lines in the 3D world are projected onto a plane differently due to the mapping functions of fisheye lenses. Popular mapping functions such as *orthographic* $r = f \sin\theta$ and *equisolid angle* $r = 2f\sin\theta/2$ (where r is the distance from the image center, f is the focal length and θ is the angle from the optical axis) transform 3D lines into elliptical or similar arcs. Thus, elliptical arcs in a parametric form are an option, but are not appropriate to represent straight lines. Instead we adopt Bézier curves due to their controllability. Regarding the degree of Bézier curves, arbitrary degrees work in the MLS warping framework [Schaefer et al. 2006]. Here we seek interactivity of warping, and thus adopt quadratic Bézier curves to simplify the derivation of a closed-form solution for MLS warping, which allows fast computation. The locus of a quadratic Bézier curve is determined as follows. By back-projecting user-specified two endpoints onto the viewing sphere, we obtain a circular arc that corresponds to a 3D line. The Bézier curve is then fitted to the projected circular arc using least-squares, and thereby the position of the middle control point is automatically determined.

2.3 MLS Warping with Quadratic Bézier Curves

Our method reduces wide-angle distortions using MLS warping [Schaefer et al. 2006]. Although the original MLS warping uses points and line segments as control handles (please refer to the paper [Schaefer et al. 2006] for details), approximating curved lines with many line segments causes high computational cost. Here we extend MLS warping to handle quadratic Bézier curves and derive an approximative closed-form solution.

Suppose that the user inputs N curves $C_i = \{\mathbf{p}_i(t)\}$ $(i = 1,2,\ldots,N,\ \mathbf{p}_i(t) \in \mathbb{R}^2,\ t \in [0,1])$. Assuming that the input image is texture-mapped on a grid, we compute a deformation function $f_{\mathbf{v}}(\mathbf{x}) = \mathbf{x}M + \mathbf{T}$ (where $\mathbf{x}$ is a vector, M a matrix and $\mathbf{T}$ a translation vector) for each grid point $\mathbf{v}$ so that it minimizes

$$\begin{aligned}&\sum_i^N \int_0^1 w_i(t)\,\|f_{\mathbf{v}}(\mathbf{p}_i(t)) - \mathbf{q}_i(t)\|^2\,dt\\ &= \sum_i^N \int_0^1 w_i(t)\,\|\mathbf{p}_i(t)M + \mathbf{T} - \mathbf{q}_i(t)\|^2\,dt, \qquad (1)\end{aligned}$$

where $\{\mathbf{q}_i(t)\}$ are the displaced counterparts of $\{\mathbf{p}_i(t)\}$ and $w_i(t)$ is a weight of $\mathbf{p}_i(t)$ for each grid point $\mathbf{v}$ given by

$$w_i(t) = \frac{\|\mathbf{p}_i'(t)\|}{\|\mathbf{p}_i(t) - \mathbf{v}\|^{2\alpha}}, \qquad (2)$$

where α is a user-defined constant and $\mathbf{p}_i'(t)$ the derivative of $\mathbf{p}_i(t)$. The quadratic function in Equation (1) is quadratic with regard to the translation vector $\mathbf{T}$, and $\mathbf{T}$ can be rewritten as $\mathbf{T} = \mathbf{q}_* - \mathbf{p}_* M$, where $\mathbf{p}_*$ and $\mathbf{q}_*$ are weighted centroids

$$\mathbf{p}_* = \frac{\sum_i^N \int_0^1 w_i(t)\mathbf{p}_i(t)dt}{\sum_i^N \int_0^1 w_i(t)dt}, \quad \mathbf{q}_* = \frac{\sum_i^N \int_0^1 w_i(t)\mathbf{q}_i(t)dt}{\sum_i^N \int_0^1 w_i(t)dt}. \qquad (3)$$

Therefore, Equation (1) can be rewritten in terms of M as

$$\sum_i^N \int_0^1 w_i(t)\,\|\hat{\mathbf{p}}_i(t)M - \hat{\mathbf{q}}_i(t)\|^2 dt, \qquad (4)$$

where $\hat{\mathbf{p}}_i(t) = \mathbf{p}_i(t) - \mathbf{p}_*$ and $\hat{\mathbf{q}}_i(t) = \mathbf{q}_i(t) - \mathbf{q}_*$. The derivation so far is almost the same as that of control line segments [Schaefer et al. 2006]. In our case, because $\mathbf{p}_i(t)$ and $\mathbf{q}_i(t)$ are quadratic Bézier curves

$$\begin{aligned}\mathbf{p}_i(t) &= (1-t)^2\mathbf{a}_i + 2t(1-t)\mathbf{b}_i + t^2\mathbf{c}_i,\\ \mathbf{q}_i(t) &= (1-t)^2\mathbf{d}_i + 2t(1-t)\mathbf{e}_i + t^2\mathbf{f}_i, \qquad (5)\end{aligned}$$

where $\mathbf{a}_i$, $\mathbf{b}_i$ and $\mathbf{c}_i$ are the control points of the original curve $\mathbf{p}_i(t)$ while $\mathbf{d}_i$, $\mathbf{e}_i$ and $\mathbf{f}_i$ are the corresponding control points of $\mathbf{q}_i(t)$. Thus $\hat{\mathbf{p}}_i(t)$ and $\hat{\mathbf{q}}_i(t)$ can be rewritten as

$$\hat{\mathbf{p}}_i(t) = \begin{pmatrix}(1-t)^2 & 2t(1-t) & t^2\end{pmatrix}\begin{pmatrix}\hat{\mathbf{a}}_i\\ \hat{\mathbf{b}}_i\\ \hat{\mathbf{c}}_i\end{pmatrix}, \qquad (6)$$

$$\hat{\mathbf{q}}_i(t) = \begin{pmatrix}(1-t)^2 & 2t(1-t) & t^2\end{pmatrix}\begin{pmatrix}\hat{\mathbf{d}}_i\\ \hat{\mathbf{e}}_i\\ \hat{\mathbf{f}}_i\end{pmatrix}, \qquad (7)$$

where $\hat{\mathbf{a}}_i, \hat{\mathbf{b}}_i, \hat{\mathbf{c}}_i$ and $\hat{\mathbf{d}}_i, \hat{\mathbf{e}}_i, \hat{\mathbf{f}}_i$ are calculated similar to $\hat{\mathbf{p}}_i(t)$ and $\hat{\mathbf{q}}_i(t)$. Equation (4) then can be rewritten as

$$\sum_i^N \int_0^1 w_i(t) \left\| \begin{pmatrix}(1-t)^2 & 2t(1-t) & t^2\end{pmatrix} \left(\begin{pmatrix}\hat{\mathbf{a}}_i\\ \hat{\mathbf{b}}_i\\ \hat{\mathbf{c}}_i\end{pmatrix} M - \begin{pmatrix}\hat{\mathbf{d}}_i\\ \hat{\mathbf{e}}_i\\ \hat{\mathbf{f}}_i\end{pmatrix}\right)\right\|^2 dt. \qquad (8)$$

Similar to the original MLS warping, M can be affine, similarity or rigid transformation. Our experiments showed that affine transformation yields the best results, and thus we only elaborate on it. Affine deformation that minimizes Equation (8) can be achieved straightforward from the solution for a classic normal equation.

$$M = \left(\sum_i^N \begin{pmatrix}\hat{\mathbf{a}}_i\\ \hat{\mathbf{b}}_i\\ \hat{\mathbf{c}}_i\end{pmatrix}^T W_i \begin{pmatrix}\hat{\mathbf{a}}_i\\ \hat{\mathbf{b}}_i\\ \hat{\mathbf{c}}_i\end{pmatrix}\right)^{-1} \sum_j^N \begin{pmatrix}\hat{\mathbf{a}}_j\\ \hat{\mathbf{b}}_j\\ \hat{\mathbf{c}}_j\end{pmatrix}^T W_j \begin{pmatrix}\hat{\mathbf{d}}_j\\ \hat{\mathbf{e}}_j\\ \hat{\mathbf{f}}_j\end{pmatrix}, \qquad (9)$$

where W_i is a weighted matrix given by

$$W_i = \begin{pmatrix}\delta_i^0 & 2\delta_i^1 & \delta_i^2\\ 2\delta_i^1 & 4\delta_i^2 & 2\delta_i^3\\ \delta_i^2 & 2\delta_i^3 & \delta_i^4\end{pmatrix}, \qquad (10)$$

where δ_i^k $(k = 0,1,2,3,4)$ are the integrals of the weight function $w_i(t)$ multiplied by different quartic polynomials

$$\delta_i^k = \int_0^1 w_i(t)\,t^k(1-t)^{4-k}dt. \qquad (11)$$

The deformation function $\mathbf{f_v}$ can be written as

$$\mathbf{f_v}(\mathbf{v}) = \sum_j^N A_j \begin{pmatrix} \hat{\mathbf{d}}_j \\ \hat{\mathbf{e}}_j \\ \hat{\mathbf{f}}_j \end{pmatrix} + \mathbf{q}_*, \tag{12}$$

where A_j is a 1×3 matrix of the form

$$A_j = (\mathbf{v} - \mathbf{p}_*) \left(\sum_i^N \begin{pmatrix} \hat{\mathbf{a}}_i \\ \hat{\mathbf{b}}_i \\ \hat{\mathbf{c}}_i \end{pmatrix}^T W_i \begin{pmatrix} \hat{\mathbf{a}}_i \\ \hat{\mathbf{b}}_i \\ \hat{\mathbf{c}}_i \end{pmatrix} \right)^{-1} \begin{pmatrix} \hat{\mathbf{a}}_j \\ \hat{\mathbf{b}}_j \\ \hat{\mathbf{c}}_j \end{pmatrix}^T W_j. \tag{13}$$

Because A_j is independent of $\hat{\mathbf{d}}_i$, $\hat{\mathbf{e}}_i$ and $\hat{\mathbf{f}}_i$, A_j can be reused once after computed. For fast computation, we seek closed-form solutions of integrals δ_i^k. However, it might be difficult or even impossible to find an antiderivative which can be written in an elementary form, although we could not prove the latter.

We modified the integrands to obtain approximative closed form solutions. Warping using these approximate solutions yields no visible difference compared to reference results computed via numerical integration.

By substituting Equation (5) into the formula of the weight function (2), we have

$$w_i(t) = \frac{2\|t(\mathbf{a}_i - 2\mathbf{b}_i + \mathbf{c}_i) + (\mathbf{b}_i - \mathbf{a}_i)\|}{\|t^2(\mathbf{a}_i - 2\mathbf{b}_i + \mathbf{c}_i) + 2t(\mathbf{b}_i - \mathbf{a}_i) + (\mathbf{a}_i - \mathbf{v})\|^{2\alpha}}. \tag{14}$$

We approximate this with $\hat{w}_i(t)$ by simplifying the numerator as follows

$$\hat{w}_i(t) = \frac{2\|\mathbf{b}_i - \mathbf{a}_i\|}{\|t^2(\mathbf{a}_i - 2\mathbf{b}_i + \mathbf{c}_i) + 2t(\mathbf{b}_i - \mathbf{a}_i) + (\mathbf{a}_i - \mathbf{v})\|^{2\alpha}}. \tag{15}$$

The integrals δ_i^k are then approximated with $\hat{\delta}_i^k$ in which $w_i(t)$ are replaced by $\hat{w}_i(t)$. Let $\beta_i^j = \int_0^1 \hat{w}_i(t)\, t^j dt$ $(j = 0, 1, 2, 3, 4)$, then $\hat{\delta}_i^k$ can be rewritten as a sum of β_i^j

$$\hat{\delta}_i^k = \sum_{j=0}^{4-k} (-1)^j C_j^{4-k} \beta_i^{k+j}, \tag{16}$$

where C_j^n are binomial coefficients, and we seek the closed-form solutions of β_i^j. See Appendix A for the details.

2.4 Straightening Constraints

To reduce wide-angle distortions, our system automatically straightens user-specified constraints (quadratic Bézier curves as described in Section 2.2) and warps the input image. If necessary, the user can further warp the image in real time by manipulating the control points of Bézier curves.

The automatic process is done in a heuristic way as follows. Let **a**, **b** and **c** be the control points of a quadratic Bézier curve, **d**, **e** and **f** the corresponding control points after straightening. Keeping the middle control points fixed, i.e., $\mathbf{b} = \mathbf{e}$, the endpoints **a** and **c** are displaced to **d** and **f** so that $(\mathbf{d} - \mathbf{f})$ becomes parallel to $(\mathbf{a} - \mathbf{c})$, while preserving the length of the curve and the distance ratio between adjacent control points; $\|\mathbf{d} - \mathbf{e}\| / \|\mathbf{f} - \mathbf{e}\| = \|\mathbf{a} - \mathbf{b}\| / \|\mathbf{c} - \mathbf{b}\|$. Additionally, the straightened curves are rotated to be aligned along horizontal or vertical axes around the middle control point if specified.

3 Results

Our algorithm was implemented using C++ and OpenGL, and parallelized using OpenMP. The experiments in this paper were conducted on a laptop with a Core 2 Dual 2.66 GHz CPU. The fisheye images in Figures 1 and 3 measure 512×512 pixels, and 800×633 pixels for Figure 4. We use grids consisting of 100×100 vertices for all input images.

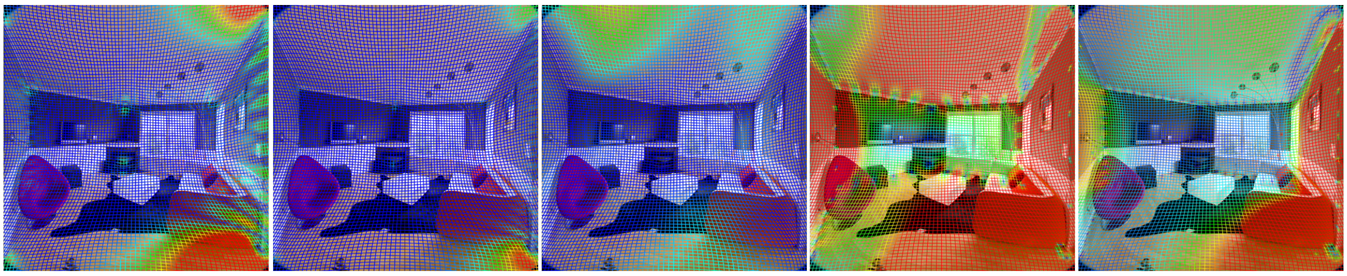
First, we explain the user session for Figure 1, which was recorded in the accompanying video. The user laid ten constraints (about 20 seconds), and then the system computed an initial solution (0.46 seconds). After the user manipulated the image in real time (about 7 msec per input), he obtained a satisfactory result (about 20 seconds for manipulation). Note that the right image of Figure 1 was manually cropped. Considering that [Carroll et al. 2009] takes more than 15 seconds for a single solve on a similar computational environment, our approach seems overall much more efficient than theirs.

We investigate the errors induced by approximation, namely the use of $\hat{w}_i(t)$ instead of $w_i(t)$ (Equations (14) and (15)). To obtain a reference, we perform numerical integration with regard to parameter t with 100 sub-intervals. Figure 2 shows errors in three types of results; numerical integration, the proposed method and segment-based warping [Schaefer et al. 2006]. The corresponding timings are shown in Table 1. Overall the proposed method works fast while restraining errors compared to others. In numerical integration, although the accuracy increases with more sub-intervals, the weight computation (required every time the user puts/deletes a control curve) took more than one second even for five sub-intervals. In segment-based warping, the errors are large even with ten segments to approximate each curve. This implies that much more segments are required to make the errors as small as those of the proposed method, and thus the computational cost becomes much higher.

For the comparison of quality, Figure 3 shows our result and that of Carroll et al.[2009]. Note that the result of [Carroll et al. 2009] is not a reference; currently there is no ground truth in the task of reducing wide-angle distortions. Compared to the result of perspective projection which keeps all 3D lines straight, both methods successfully straighten salient lines. However, the purple chair in our result is distorted unnaturally. This is due to the disadvantage of local optimization, and can be alleviated by adding a little more constraints.

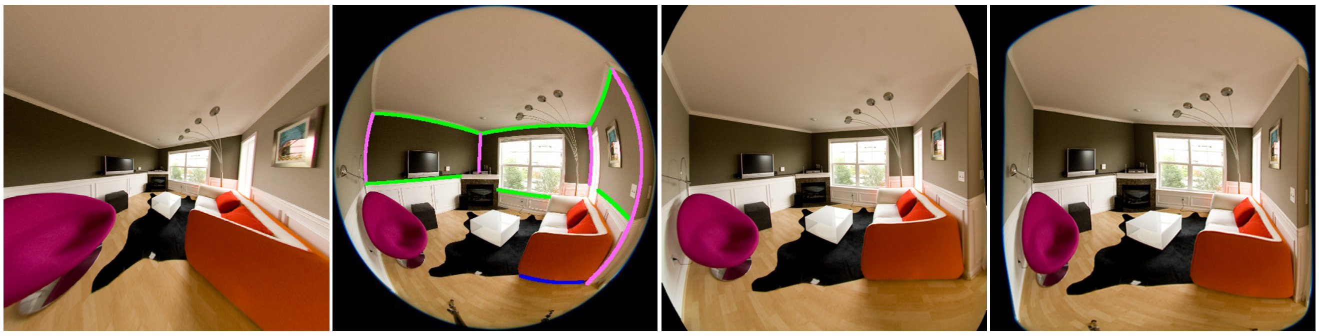
In Figure 4 we compare our result with those of Zelnik-Manor et al. [2005]. In this example, (a) the chair is distorted due to the seam between projection planes. Thus Zelnik-Manor et al. segment out foreground objects, performing local perspective projections in (b). Unfortunately, discontinuity artifacts remain especially around the monitors. In contrast, (d) our method successfully straightened (c) the user-specified 16 curved lines without introducing visible distortions.

Figure 5 shows the screenshot of our interactive viewer with which the user can look around a 360° panorama with a low-distortion fisheye view. The left window displays a 3D sphere, and the right displays the resultant image. Using the pre-defined circular arcs on the sphere, the system computes constraints and synthesizes the warped fisheye view on the fly, every time the user changes the view. In this example, we mapped a 360° panorama image with the size of 1024×512 onto the sphere and used 15 constraints. The computational time was about 90 msec for each view change.



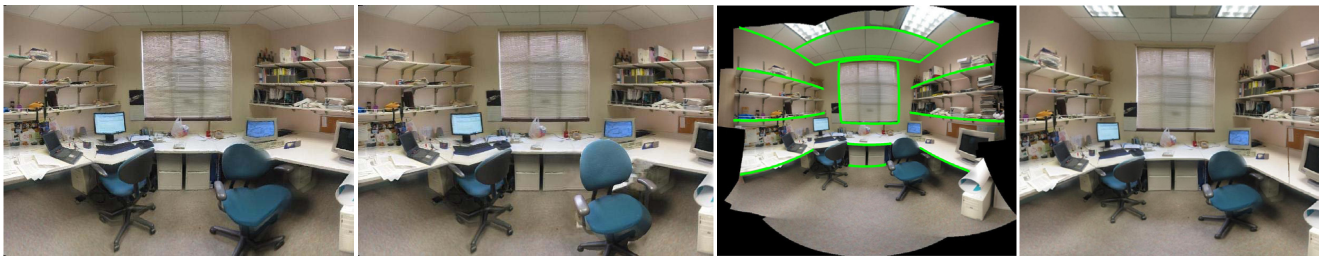
(a) Numerical integration $N_t = 5$ (b) Numerical integration $N_t = 10$ (c) Proposed method (d) Segment approximation $N_{seg} = 5$ (e) Segment approximation $N_{seg} = 10$

Figure 2: Pseudo color images whose colors indicate the magnitude of the error at each vertex in warped grids. The layout of curves is the same as Figure 1. The errors at blue vertices are zero while those at red vertices are more than two pixels. (a)(b) Numerical integration was performed with N_t sub-intervals. (d)(e) Each curve was approximated with N_{seg} segments and Schaefer et al.'s segment-based warping was applied. The errors in the result of (c) the proposed method are moderate (1.6 pixels at a maximum) compared to others. See also Table 1 for the timings.



(a) Perspective (b) 11 constraints for [Carroll et al. 2009] (c) Result of [Carroll et al. 2009] (d) Our result

Figure 3: Comparison between the results of (a) perspective projection, (c) Carroll et al. and (d) ours (the same as Figure 1, but roughly cropped). Note that (c) is not a ground truth. Both in (c) and (d), salient lines are successfully straightened.



(a) Multi-plane [Zelnik-Manor et al. 2005] (b) Multi-view [Zelnik-Manor et al. 2005] (c) Our 16 constraints (d) Our result

Figure 4: Comparison between the results of (a) multi-plane and (b) multi-view techniques of Zelnik-Manor et al. and (d) ours. Distortions due to discontinuities with (a) multi-plane projection are only partially resolved with (b) multi-view projection. In contrast, (d) our method successfully reduces almost all noticeable distortions with (c) 16 constraints (green).

Table 1: Timings (seconds) for the results of Figure 2.

Methods	Weight Comp.	Warping	Total
(a) Num. Integ. $N_t = 5$	1.11	0.03	1.14
(b) Num. Integ. $N_t = 10$	2.14	0.03	2.17
(c) Proposed method	0.28	0.04	0.32
(d) Seg. Appox. $N_{seg} = 5$	0.24	0.14	0.38
(e) Seg. Appox. $N_{seg} = 10$	0.43	0.28	0.71

4 Conclusions and Future Work

We have presented a fast algorithm for reducing the undesired distortions (e.g., bending of straight lines) that often appear in wide-angle images especially taken with fisheye lenses. Our method allows the user to intuitively specify curves to be straightened by using quadratic Bézier curves (Section 2.2), and then automatically warps the image based on MLS (Sections 2.3 and 2.4). Unlike previous methods [Kopf et al. 2009; Carroll et al. 2009], our method provides results instantly, and allows the user to further manipulate

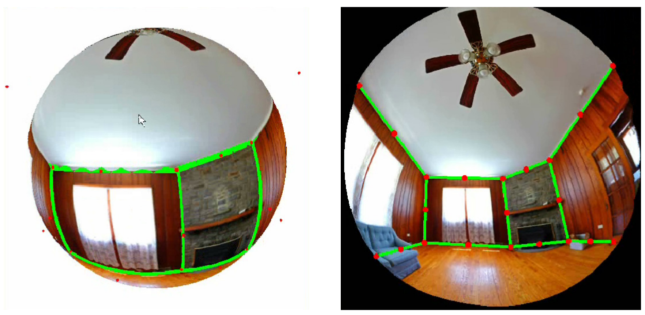

(a) Edit window (b) View window

Figure 5: Screenshot of our interactive viewer. After placing control curves in (a) the edit window, the user can look around a 360° panorama with a low-distortion fisheye view in real time in (b) the view window.

curves to obtain better results in real time, thanks to the approximative closed-form solutions for MLS warping (Section 2.3).

While we currently entrust the user to input constraints, we plan to employ automatic recognition of regions of interest to help the user accomplish this task more efficiently. By automatically extracting the most salient curves that should be straightened from the input image, the manual labor for placing curves will be eliminated.

References

Carroll, R., Agrawal, M., and Agarwala, A. 2009. Optimizing content-preserving projections for wide-angle images. *ACM Trans. Graph. 28*, 3, 1–9.

Gal, R., Sorkine, O., and Cohen-Or, D. 2006. Feature-aware texturing. In *Proceedings of Eurographics Symposium on Rendering*, 297–303.

Kopf, J., Lischinski, D., Deussen, O., Cohen-Or, D., and Cohen, M. 2009. Locally adapted projections to reduce panorama distortions. *Computer Graphics Forum (Proceedings of EGSR 2009) 28*, 4, 1083–1089.

Schaefer, S., McPhail, T., and Warren, J. 2006. Image deformation using moving least squares. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, 533–540.

Wang, Y.-S., Tai, C.-L., Sorkine, O., and Lee, T.-Y. 2008. Optimized scale-and-stretch for image resizing. *ACM Trans. Graph. 27*, 5, 1–8.

Wolf, L., Guttmann, M., and Cohen-Or, D. 2007. Non-homogeneous content-driven video-retargeting. In *Proceedings of the Eleventh IEEE International Conference on Computer Vision (ICCV-07).*

Zelnik-Manor, L., Peters, G., and Perona, P. 2005. Squaring the circles in panoramas. In *ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision*, IEEE Computer Society, Washington, DC, USA, 1292–1299.

Zorin, D., and Barr, A. H. 1995. Correction of geometric perceptual distortions in pictures. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 257–264.

A Approximative Closed-form Solutions

In this appendix, we derive the closed-form solutions for integrals $\beta_i^j = \int_0^1 \hat{w}_i(t)\, t^j dt$ (where $j = 0, 1, 2, 3, 4$ and $\hat{w}_i(t)$ is the approximated weight function defined as Equation (15)) which are required for the approximative closed-form solutions for MLS warping with quadratic Bézier curves, as described in Section 2.3.

For a curve $\mathbf{p}_i(t) = (1-t)^2\mathbf{a}_i + 2t(1-t)\mathbf{b}_i + t^2\mathbf{c}_i$ and a grid vertex $\mathbf{v}$, we calculate closed-form solutions for β_i^j in the case of $\alpha = 1$. Here we only elaborate on the derivation of β_i^0. The others can be derived similarly using computer algebra softwares. To shorten the expressions, the subscripts i are omitted in the following.

First of all, we solve the following integrals

$$\gamma = \int_0^1 \frac{dt}{||\mathbf{p}_i(t) - \mathbf{v}||^2}, \tag{17}$$

which will be used in the following steps. Let

$$\mathbf{A} = \mathbf{a} - 2\mathbf{b} + \mathbf{c}, \quad \mathbf{B} = \mathbf{b} - \mathbf{a}, \quad \mathbf{C} = \mathbf{a} - \mathbf{v},$$
$$D_1 = \mathbf{B}_x^2 - \mathbf{B}_y^2 - \mathbf{A}_x\mathbf{C}_x + \mathbf{A}_y\mathbf{C}_y, \quad D_2 = 2\mathbf{B}_x\mathbf{B}_y - \mathbf{A}_x\mathbf{C}_y - \mathbf{A}_y\mathbf{C}_x,$$
$$E_1 = \sqrt{\frac{\sqrt{D_1^2 + D_2^2} + D_1}{2}}, \quad E_2 = \sqrt{\frac{\sqrt{D_1^2 + D_2^2} - D_1}{2}},$$
$$x_1 = \frac{E_1\mathbf{A}_x - \mathbf{B}_x\mathbf{A}_x + \mathbf{A}_y E_2\, sign(D_2) - \mathbf{A}_y\mathbf{B}_y}{||\mathbf{A}||^2},$$
$$x_2 = \frac{E_2\mathbf{A}_x\, sign(D_2) - \mathbf{B}_y\mathbf{A}_x - \mathbf{A}_y E_1 + \mathbf{A}_y\mathbf{B}_x}{||\mathbf{A}||^2},$$
$$x_3 = \frac{-E_1\mathbf{A}_x - \mathbf{B}_x\mathbf{A}_x - \mathbf{A}_y E_2\, sign(D_2) - \mathbf{A}_y\mathbf{B}_y}{||\mathbf{A}||^2},$$
$$x_4 = \frac{-E_2\mathbf{A}_x\, sign(D_2) - \mathbf{B}_y\mathbf{A}_x + \mathbf{A}_y E_1 + \mathbf{A}_y\mathbf{B}_x}{||\mathbf{A}||^2},$$
$$x_5 = x_1 - x_3, \quad x_6 = \frac{(x_1 - x_3)^2}{4} + x_2^2, \quad x_7 = \frac{(x_1 - x_3)^2}{4} + x_4^2,$$
$$x_8 = -\frac{x_1 + x_3}{2}, \quad x = t - \frac{x_1 + x_3}{2} = t + x_8,$$

where $sign(x)$ denotes the signature of x. The integral γ can be rewritten as

$$\gamma = \frac{1}{||\mathbf{A}||^2}\int_0^1 \frac{dt}{((t - x_1)^2 + x_2^2)((t - x_3)^2 + x_4^2)} = \frac{1}{||\mathbf{A}||^2} \times I,$$

where

$$I = \int_{x_8}^{1+x_8} \frac{dx}{OP}.$$

Next, let

$$M = \frac{-2x_5}{2x_5^2(x_6 + x_7) + (x_6 - x_7)^2},$$
$$N = \frac{2x_5^2 - x_6 + x_7}{2x_5^2(x_6 + x_7) + (x_6 - x_7)^2},$$
$$L = \frac{-2x_5^2 - x_6 + x_7}{2x_5^2(x_6 + x_7) + (x_6 - x_7)^2},$$
$$O = x^2 - x_5 x + x_6, \quad P = x^2 + x_5 x + x_7,$$

then

$$I = \int_{x_8}^{1+x_8} \left(\frac{Mx+N}{O} - \frac{Mx+L}{P} \right) dx = I_1 - I_2,$$

where

$$I_1 = \left[\frac{M \log O}{2} + \frac{2N + Mx_5}{\sqrt{4x_6 - x_5^2}} \arctan \left(\frac{2x - x_5}{\sqrt{4x_6 - x_5^2}} \right) \right]_{x_8}^{1+x_8},$$

$$I_2 = \left[\frac{M \log P}{2} + \frac{2L - Mx_5}{\sqrt{4x_7 - x_5^2}} \arctan \left(\frac{2x + x_5}{\sqrt{4x_7 + x_5^2}} \right) \right]_{x_8}^{1+x_8}.$$

Using the same notations, we can rewrite β_i^0 as follows

$$\beta_i^0 = \frac{2||\mathbf{B}||}{||\mathbf{A}||^2} \times I^0,$$

where

$$I^0 = \int_{x_8}^{1+x_8} \left(\frac{Mx+N}{O} - \frac{Mx+L}{P} \right)^2 dx = I_1^0 + I_2^0 - 2I_3^0,$$

in which

$$I_1^0 = \int_{x_8}^{1+x_8} \left(\frac{Mx+N}{O} \right)^2 dx,$$

$$I_2^0 = \int_{x_8}^{1+x_8} \left(\frac{Mx+L}{P} \right)^2 dx,$$

$$I_3^0 = \int_{x_8}^{1+x_8} \frac{(Mx+N)(Mx+L)dx}{OP}.$$

Let

$$M_1 = 2MN + M^2 x_5, \quad N_1 = N^2 - M^2 x_6,$$

$$M_2 = 2ML - M^2 x_5, \quad N_2 = L^2 - M^2 x_7,$$

$$P = \frac{M(Mx_5 + N + L)}{2x_5}, \quad Q = \frac{M(Mx_5 - N - L)}{2x_5},$$

$$R = NL - Qx_6 - Px_7,$$

then

$$I_1^0 = M^2 I_{11}^0 + I_{12}^0, \quad I_2^0 = M^2 I_{21}^0 + I_{22}^0, \quad I_3^0 = P I_{11}^0 + Q I_{21}^0 + R\gamma,$$

where

$$I_{11}^0 = \frac{2}{\sqrt{4x_6 - x_5^2}} \left[\arctan \frac{2x - x_5}{\sqrt{4x_6 - x_5^2}} \right]_{x_8}^{1+x_8},$$

$$I_{12}^0 = \left[\frac{-M_1}{2O} \right]_{x_8}^{1+x_8} + \frac{2N_1 + M_1 x_5}{4x_6 - x_5^2} \left(\left[\frac{x - x_5/2}{O} \right]_{x_8}^{1+x_8} + I_{11}^0 \right),$$

$$I_{21}^0 = \frac{2}{\sqrt{4x_7 - x_5^2}} \left[\arctan \frac{2x + x_5}{\sqrt{4x_7 - x_5^2}} \right]_{x_8}^{1+x_8},$$

$$I_{22}^0 = \left[\frac{-M_2}{2P} \right]_{x_8}^{1+x_8} + \frac{2N_2 - M_2 x_5}{4x_7 - x_5^2} \left(\left[\frac{x + x_5/2}{P} \right]_{x_8}^{1+x_8} + I_{21}^0 \right),$$

and γ is the solution obtained in Equation (17).