CGI2017 manuscript No.
(will be inserted by the editor)

Robust Upright Adjustment of 360 Spherical Panoramas

Jinwoong Jung · Beomseok Kim · Joon-Young Lee · Byungmoon Kim · Seungyong Lee

Abstract With the recent advent of 360 cameras, spherical panorama images are becoming more popular and widely available. In a spherical panorama, alignment of the scene orientation to the image axes is important for providing comfortable and pleasant viewing experiences using VR headsets and traditional displays. This paper presents an automatic method for upright adjustment of 360 spherical panorama images without any prior information, such as depths and Gyro sensor data. We take the Atlanta world assumption and use the horizontal and vertical lines in the scene to formulate a cost function for upright adjustment. In addition to fast optimization of the cost function, our method includes outlier handling to improve the robustness and accuracy of upright adjustment. Our method produces visually pleasing results for a variety of real-world spherical panoramas in less than a second and the accuracy is verified using ground-truth data.

Keywords spherical panorama · upright adjustment · camera calibration

1 Introduction

Spherical panoramas (360 panoramas) are images with 360 degrees of horizontal and 180 degrees of vertical field of views. In the past, 360 panoramas used to be niche, typically created by skilled professionals: first carefully taking a set of images and then stitching them using special software or service. Recently, this has been changing as mobile panorama applications have been released, and the use of 360 panoramas is accelerated as new 360 cameras become available. Moreover, as virtual reality (VR) headsets are getting popular and VR medias appear to be strong among VR applications, 360 camera development is further accelerated in both low-cost and high-end flavors from large camera makers and start-up companies.

360 panoramas can be seen in traditional monitors or mobile displays with special software that allows a user to change the viewing direction. 360 panoramas can also be viewed with a VR headset that allows a user to look at any direction by simply turning the head. With a VR headset, a user typically feels presence inside the scene where the panorama image was taken, even when the panorama is a single mono image, similarly to when we close one eye, we do not lose much of depth perception in typical scenes. Such presence at the scene is a new photography experience to most users, and therefore, delivering high-quality VR experience is an important part of 360 panorama imaging.

In typical 360 panorama images taken with a low-cost camera, e.g., Ricoh Theta S ($350), wavy horizons and slanted objects due to slight camera tilts and rolls are common, as shown in Figure 1. By design, 360 cameras do not provide stable grips or viewfinders while users often take lots of photos with hand-held cameras. VR experience of such mis-oriented panorama images would be unpleasant: users may feel like falling down or leaning toward the ground in the scene (Figure 2). The correction of this mis-orientation will greatly improve the experience as the user now feels standing straight in the scene. This kind of correction is called *upright adjustment* [10].

Jinwoong Jung, Beomseok Kim, Seungyong Lee
Department of Computer Science and Engineering,
POSTECH, Pohang, South Korea
E-mail: {ca00229, beomseok0203, leesy}@postech.ac.kr

Joon-Young Lee, Byungmoon Kim
Adobe Research, San Jose, CA, USA
E-mail: {jolee, bmkim}@adobe.com

Fig. 1 Our method automatically uprights a single 360 spherical panorama image (left) by aligning the perceived upward direction of the scene to the vertical axis of the panorama image, producing the result image (right) that is much more pleasant to look at. To achieve this, we detect lines, find great circles, analyze vanishing points, and finally solve for optimal 3D rotation.

Fig. 2 A spherical panorama viewed with a VR headset could be unpleasant if it is tilted (left). Users may feel like falling down or leaning toward the ground. If this tilt is corrected, the user would feel standing straight (right), greatly improving the viewing experience.

With upright adjustment, monitor or mobile display viewing experiences can also be improved for spherical panoramas. Grasping of the scene in the whole panorama view mode benefits from a straight horizon with minimal distortions and non-slanted objects near the horizon. In the narrow field of view mode, when a user scrolls the view horizontally to the left or right, the horizon stays around the middle line of the image, rather than continuously fluctuating up and down.

In this paper, we present an automatic method for upright adjustment of 360 spherical panorama images. We take the Atlanta world assumption [12] for the scene, and find horizontal and vertical great circles that represent the input spherical panorama. We formulate a cost function for upright adjustment by imposing the desired properties on the great circles from vertical lines and the vanishing points from horizontal lines. Then, we compute the optimal rotation to resolve the mis-orientation of the camera by minimizing the cost function, where a rotation is represented as an update of the north pole position of the sphere. To make our method more robust and accurate, we also present effective outlier handling methods for removing misclassification of a horizontal line as a vertical one, as well as reducing the influences of less-accurate great circles. Various experimental results show that our method successfully achieves upright adjustment of spherical panoramas in a fast, accurate, and robust way.

The main contributions of our work are summarized as follows:

- Analysis of the upright adjustment problem of spherical panoramas, providing a simple but effective formulation with a cost function
- Fast method for optimizing the cost function to find an optimal rotation that resolves the mis-orientation of the camera
- Effective outlier handling methods to improve the robustness and accuracy of upright adjustment.

2 Related Work

Images captured by casual users often appear to be distorted due to the tilt and roll of a camera, and the distortions degrade the visual quality of images. To improve the perceptual quality of such images, Gallagher [4] proposed one of the pioneering work that automatically corrects camera tilt in an image. This method estimates the vertical vanishing point of an image and adjusts the rotation by placing the vanishing point on the y-axis. Recently Lee et al. [10] introduced a set of criteria for upright adjustment of photographs and proposed an optimization framework that estimates a homography to satisfy the criteria. To reduce warping distortion and avoid cropping due to a rigid transformation during upright adjustment, He et al. [6] presented an optimization-based warping method that enables the rotation of horizontal and vertical lines without cropping. Image editing tools, such as Adobe Photoshop, also provide not only an automatic upright tool but also manual adjustment tools like 3D image plane rotation to correct distortions and transform the perspective in an image. With these techniques and tools, we can obtain visually pleasing upright adjustment results for most of the conventional camera images.

However, applying the upright adjustment method [10] for ordinary images to 360 panoramas is cumbersome

and would not be the best. It may require first taking a perspective crop from the input panorama and then computing the homography that uprights the crop, followed by figuring out the rotation matrix corresponding to the homography. With this approach, the upright adjustment result would depend on the crop, and finding the best crop is not a simple problem. Furthermore, there is no reason to crop and give up information available in every view direction in a 360 panorama. Our method directly handles upright adjustment of 360 panoramas, taking advantage of all information in the input for robust and accurate results.

There have been studies that focus on enhancing the visual quality of wide angle views. Cylindrical and spherical projections are typically used to show 360 panorama images on a flat 2D display surface, but they introduce distortions like curving straight lines. To reduce the distortions, Kopf et al. [9] presented locally-adapted projections, where the user specifies regions where a perspective-like projection is desired, and then a mapping is computed to minimize distortions while being planar in the specified regions. Instead of cylindrical projection, Wang et al. [14] proposed a new representation method for panoramas using cube unwrapping. This method estimates vanishing points to rectify the orientation and reproduces the final cube unwrapping with aesthetic and information-preserving criteria. For fisheye images, Wang et al. [13] introduced an adjustment technique to correct the rotation angle of a camera and complete the missing parts. However, these methods adjust images to display wide-angle views on a flat 2D surface while our method provides upright adjustment for displaying on a VR headset.

The analysis on 360 panorama images has been actively studied in the field of robot vision using omnidirectional cameras [11,7,2,3]. In particular, Bazin et al. [2] considered vanishing point extraction and rotation estimation from omnidirectional images, and Kamali et al. [7] presented stabilization results from omnidirectional videos. Recently, Kopf et al. [8] solved omnidirectional stabilization using a deformable rotation motion model from feature trajectories.

On the other hand, there has been little attention on an automatic upright adjustment of 360 panoramas. Bazin et al. [2] applied vanishing point extraction and rotation estimation to omnidirectional video stabilization, and their method has an initialization step for orienting the first frame, which is the most closely related work to ours. The method achieves upright results by aligning the vertical vanishing points with the upward direction, so it works reasonably well if accurate vertical vanishing points can be extracted. However, this simple solution easily fails when images contain noisy edges or violate the Manhattan world assumption (Figure 8).

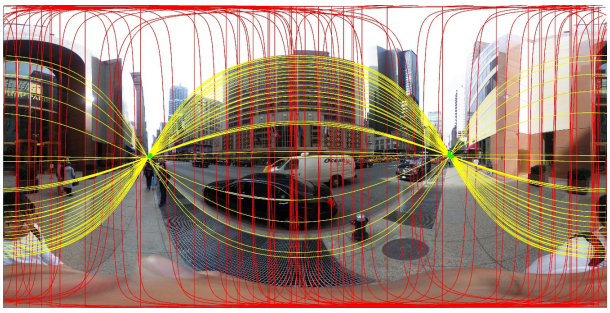


Fig. 3 Horizontal and vertical lines and their vanishing points. We can obtain the red circles by projecting vertical lines, and the intersections of the circles are north and south poles. Following our Atlanta world assumption, there can be multiple horizontal vanishing points along the equator. We express the vanishing points from horizontal lines as green dots.

3 Formulation

For upright adjustment of an input 360 spherical panorama image, we need to find a rotation that resolves the mis-orientation of the camera. Once the rotation has been computed, the adjusted image can be easily obtained by resampling the input image. This differs from the upright adjustment of an ordinary perspective image [10] which introduces distortions and cropping of image boundary due to a homography-based image transformation. In the case of a spherical panorama, an upright adjustment is achieved by rotating an axis in the spherical coordinate, therefore there is no cropping and content loss.

The rotation needed for upright adjustment of a spherical panorama has 2 DOF, i.e., *tilt* and *roll*, as *pan* is not related to the undesirable inclination of the camera. A tilt can resolve the difference between the physical and perceived horizons (eye levels) when a user watches a spherical panorama using a VR headset. Similarly, a roll can handle the mismatch of the physical and perceived straight-up directions. In contrast, a pan simply changes the initial forward direction for viewing with a VR headset and does not affect the perception of horizons and object orientations. We represent our desired 2 DOF rotation as the update of the north pole position on the sphere, where the position is specified by a unit vector. Consequently, our problem is reduced to finding a unit vector $\overrightarrow{P}$ for the updated north pole, whose original position is $\overrightarrow{P}_0 = \overrightarrow{y} = (0, 1, 0)$.

A spherical panorama image usually has no depth information. As a scene prior for structure analysis, we take the Atlanta world assumption [12], where there is one shared vertical direction in the objects and several

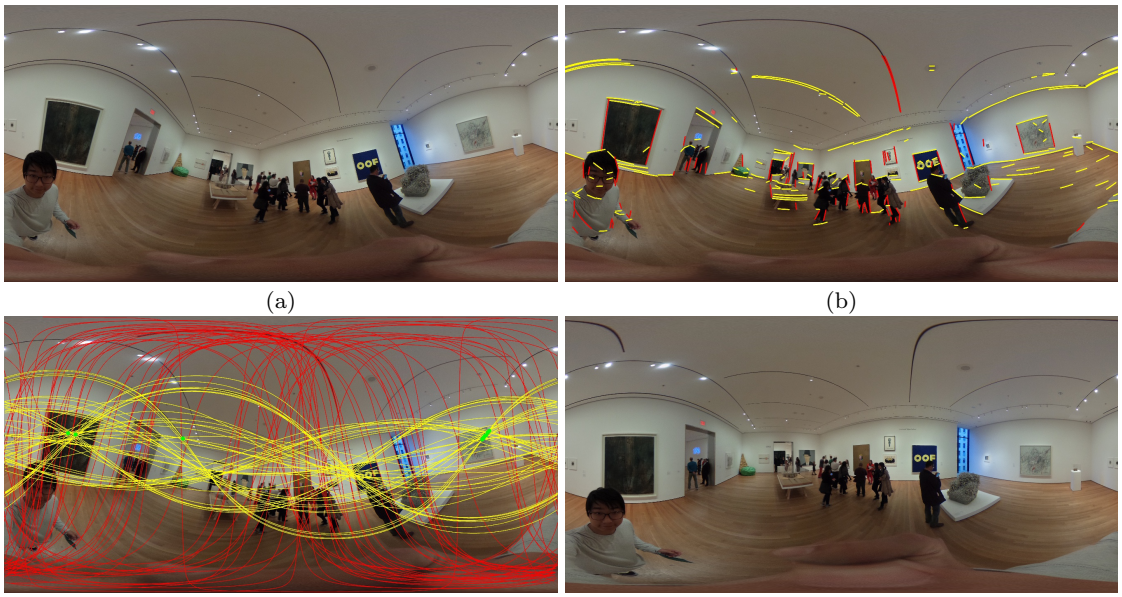


Fig. 4 Overall process. (a) input image. (b) lines detected and classified: red for vertical lines, and yellow for horizontal lines. (c) great circles from the classified lines. Green dots are vanishing points computed from horizontal (yellow) lines. (d) upright adjustment result.

dominant horizontal directions that are orthogonal to the vertical direction. This assumption works well for most indoor and outdoor scenes with man-made objects.

Now consider horizontal and vertical lines in the scene. A horizontal line is parallel to the ground plane, and a vertical line is in the straight-up direction perpendicular to the ground. When a 3D line is projected onto a spherical panorama, it is mapped onto a great circle of the sphere whose center is the camera position. In the ideal case with no tilt and roll of the camera, a great circle corresponding to a vertical line always passes the north pole. In other words, the north pole is a vanishing point of the vertical lines. For horizontal lines, a set of parallel lines forms two vanishing points. That is, the great circles corresponding to the parallel lines meet at two antipoles on the sphere (Figure 3). The vanishing points change with the orientations of parallel lines, but they always lie on the equator of the sphere in the ideal case with no tilt and roll.

For a 3D vertical line l, we represent the corresponding great circle on the unit sphere as a unit vector $\overrightarrow{v}$. We first sample two points, p_1 and p_2, on the line and project the points onto the sphere, obtaining p_1' and p_2'. Let c be the center of the sphere. Then, the cross-product of two vectors from c to p_1' and p_2' gives a unit vector $\overrightarrow{v} = (p_1' - c) \times (p_2' - c)$, which is perpendicular to the great circle. In addition, the vanishing point of a set of parallel horizontal lines is a point on the sphere and can be represented as a unit vector $\overrightarrow{h}$. Recall that both unit vectors $\overrightarrow{v}$ and $\overrightarrow{h}$ should be perpendicular to the north pole vector $\overrightarrow{P}$ if there is no tilt and roll of the camera.

Finally, we formulate the cost function for upright adjustment of a spherical panorama as follows.

$$E(\overrightarrow{P}) = \alpha \sum_i (\overrightarrow{v_i} \cdot \overrightarrow{P})^2 + \beta \sum_j (\overrightarrow{h_j} \cdot \overrightarrow{P})^2 + \lambda(1 - \overrightarrow{y} \cdot \overrightarrow{P})^2, \tag{1}$$

where $\overrightarrow{v_i}$ is the great circle of a vertical line l_i, and $\overrightarrow{h_j}$ is the vanishing point in a horizontal direction parallel to the ground. The last term is for regularization to prevent a drastic change of the north pole from the original position (y-axis). It can effectively prevent unnecessarily large rotations when line detection has failed and produced an erroneous set of lines. α, β, and λ are user-specified relative weights of the terms. We estimate the updated position of the north pole by finding a unit vector $\overrightarrow{P}$ that minimizes Eq. (1). Then, we achieve upright adjustment which resolves the tilt and roll of the camera by aligning the north pole to $\overrightarrow{P}$.

4 Algorithm

Image representation Our 360 spherical panorama image is represented as a 2D rectangular array of pixels. A pixel in an image contains the color value of the view direction (ϕ, θ), where $0 \leq \phi < 2\pi$ and $-\frac{\pi}{2} \leq \theta \leq \frac{\pi}{2}$. Figures 1 and 3 show examples. For notational convenience, we use x and y, instead of ϕ and θ, to represent

the longitude and latitude of the sphere. We assume the input image is equipped with no depth information (Figure 4(a)).

Line segment (arc) detection To find great circles corresponding to 3D lines in the scene, we first detect line segments from the input spherical panorama image. However, a 3D line segment in the scene is mapped onto an arc in a spherical panorama image, and it would not be effective to directly apply a line detection algorithm to the input panorama image. Thus, we first resample the spherical panorama image onto the cubemap and apply LSD algorithm [5] to the cubemap images, although any line detection algorithm (e.g. EDLines [1]) can also be used. Each cube face is grayscale image of the size 256×256 regardless of input image resolution. The detected line segments on the cubemap are converted to arcs on the spherical panorama. The top and bottom images of the cubemap would have little information for upright adjustment, and we use only four side images for line detection, where the vertical FOV of each side image is increased to $\frac{2}{3}\pi$ from the ordinary $\frac{\pi}{2}$ in order to cover more areas. In this paper, for convenience, we use a *line segment* and an *arc* interchangeably to represent the projection of a 3D line segment in the scene onto the spherical panorama image.

In a given spherical panorama captured with a misoriented camera, 2D line segments mapped from 3D vertical and horizontal lines would appear slanted reflecting the camera orientation. We define the *slanted angle* of a 2D line segment as the angle from the x-axis of the image. If the slanted angle of a detected line segment is less than γ_1, it is classified as horizontal. If the angle is between γ_1 and γ_2, the line segment is discarded. If the angle is greater than γ_2, the line segment is classified as vertical. We use $\gamma_1 = \frac{\pi}{6}$ and $\gamma_2 = \frac{\pi}{3}$ for all examples in our experiments. Figure 4(b) shows the detected horizontal and vertical line segments.

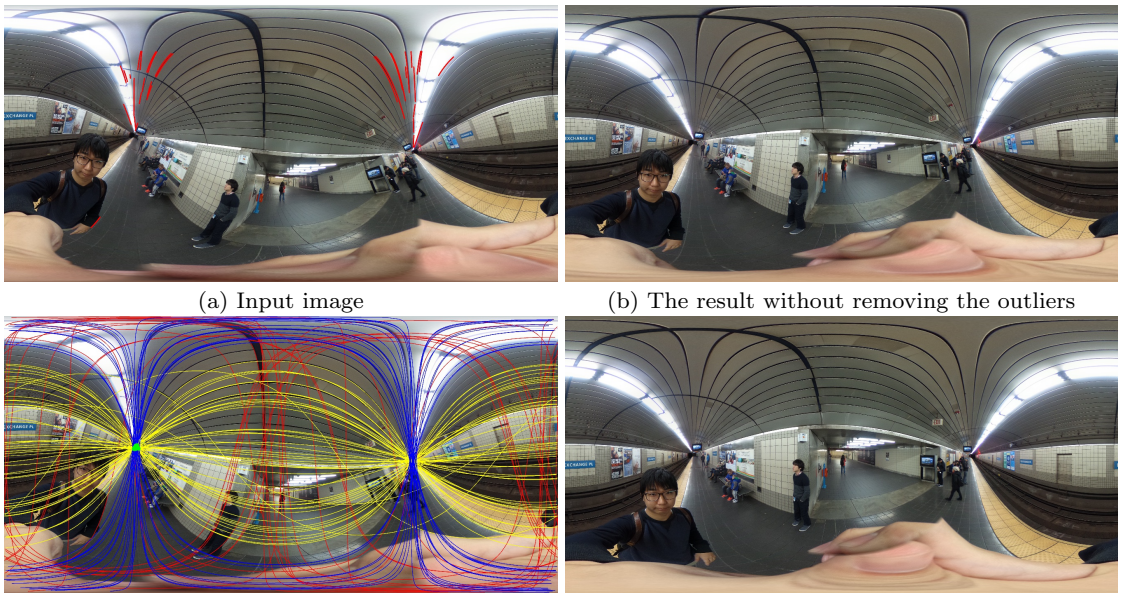
Great circle detection To recover the great circles of the corresponding 3D lines in the scene, we apply spherical Hough transform to the detected horizontal and vertical line segments. For a line segment, the two endpoints are points on the sphere and determine a great circle, which can be represented by a unit vector, as described in Section 3. We quantize the unit hemisphere into the resolution of $m \times n$ and accumulate the unit vectors of the great circles produced by horizontal and vertical line segments, where the lengths (a.k.a. spatial degree) of line segments are used for the weights of the accumulation. In our implementation, we use $m = 360$ and $n = 90$. After accumulating all line segments, we select top $t_1\%$ horizontal and top $t_2\%$ vertical great circles using the accumulated weights, where the maximum numbers of the selected great circles are limited by k_1 and k_2, respectively. We use $t_1 = t_2 = 10$ and $k_1 = k_2 = 50$. Figure 4(c) visualizes the detected great circles overlaid on the input image.

Vanishing point detection Using the great circles recovered from horizontal line segments, we find vanishing points $\overrightarrow{h_j}$ used for Eq. (1). We again quantize the unit hemisphere into the resolution of $m \times n$, and rasterize the great circles to find intersections. The weight of a great circle accumulated in the spherical Hough transform is added to the quantized cells in the rasterization process. We finally choose top $t_3\%$ non-zero cells containing the largest weights to determine vanishing points from parallel horizontal lines, where k_3 is the maximum allowed number of chosen cells. We use $t_3 = 10$ and $k_3 = 30$. In Figure 4(c), the intersections of great circles from horizontal lines show the detected vanishing points.

Optimization Although it is quadric in terms of $\overrightarrow{P}$, optimization of Eq. (1) could be complicated due to the unit vector constraint on $\overrightarrow{P}$. We can find a geometric solution that optimizes Eq. (1) on the surface of the unit sphere. That is, each term in Eq. (1) gives a great circle constraint on the target position of $\overrightarrow{P}$. Then, we can compute the intersections of the great circles from the terms in Eq. (1) and take the weighted average of the intersection points, e.g., using quaternion arithmetic. However, we found that in most cases simply computing the least-square solution of Eq. (1) and then normalizing the solution would provide accurate enough optimization results for $\overrightarrow{P}$. Figure 4(d) shows the upright adjustment result obtained by optimizing $\overrightarrow{P}$ with this simple approach.

5 Outlier Handling

Misclassified 3D horizontal line removal In Section 4, we use simple thresholding to determine horizontal and vertical line segments. If the slanted angle of a line segment, at the cubemap, detected from the input spherical panorama is greater than γ_2, it is classified as vertical. However, this thresholding could include outliers, where a line segment corresponding to a 3D horizontal line in the scene is misclassified as vertical. This case happens when a 3D horizontal line is oriented toward the view direction in the scene, having rapid depth changes from the viewpoint. Then parts of the projection of the 3D line, around the vanishing points, could



(a) Input image (b) The result without removing the outliers

(c) Visualization of great circles (d) The result with outliers removal

Fig. 5 Outlier handling for misclassified horizontal lines. (a) input image. Red line segments are outliers, i.e., sampled from 3D horizontal lines but classified as vertical by our thresholding. (b) result without removing the outliers from vertical lines. (c) visualization of great circles: red for vertical lines, yellow for horizontal lines, and blues for outliers (vertical lines that pass vanishing points depicted as green dots). (d) result with outliers removed, i.e., only with red and yellow lines.

Fig. 6 Iterating the upright adjustment improves the result quality. (left) the input image, (middle) after the first iteration, (right) after the second iteration. We apply the iteration about three times, and each iteration takes about 45 milliseconds.

appear almost vertical in the cubemap image (also in panorama image) (Figure 5(a)). This misclassification can incur a wrong result of the upright adjustment, as shown in Figure 5(b). Note that this misclassification of 3D horizontal lines is the only case of outliers that could happen with our thresholding. A 3D vertical line in the scene may not be classified as horizontal in the panorama image unless a drastic roll has been intentionally applied to the camera.

To handle this misclassification, we use vanishing points of horizontal lines. If the great circle of a line segment l passes through the vanishing point determined from a set of horizontal lines, line l should be parallel to the line set. Base on this property, we check each line segment classified as vertical by our thresholding to check whether its great circle intersects any of vanishing points computed from horizontal line segments. If the great circle has an intersection, we discard the line segment in the optimization process. Recall that we have already computed the vanishing points of horizontal line segments for Eq. (1), thus this outlier removal can be done with negligible computational overhead. Figure 5(c) visualizes the outliers to be removed, and Figure 5(d) shows that we achieve correct upright adjustment with our outlier removal.

Adaptive weight normalization Due to the 3D lines which are not parallel (perpendicular) to the ground plane but classified horizontal (vertical) line, some of the detected line segments and consequently some of the recovered great circles could be less accurate than others. To reduce the influences of these less accurate great circles in the optimization process, we revise Eq. (1) as follows:

$$E(\overrightarrow{P}) = \alpha \sum_i \omega_i^v (\overrightarrow{v_i} \cdot \overrightarrow{P})^2 + \beta \sum_j \omega_j^h (\overrightarrow{h_j} \cdot \overrightarrow{P})^2,$$

$$\text{s.t.} \quad \omega_i^v = e^{-(\overrightarrow{v_i} \cdot \overrightarrow{P_{old}})^2 / 2\sigma_v^2}, \ \omega_j^h = e^{-(\overrightarrow{h_j} \cdot \overrightarrow{P_{old}})^2 / 2\sigma_h^2},$$

$$\sigma_v^2 = \frac{1}{n_v} \sum_i (\overrightarrow{v_i} \cdot \overrightarrow{P_{old}})^2, \ \sigma_h^2 = \frac{1}{n_h} \sum_j (\overrightarrow{h_j} \cdot \overrightarrow{P_{old}})^2, \quad (2)$$

where ω_i^v and ω_j^h are weights for vertical great circles $\overrightarrow{v_i}$ and horizontal vanishing points $\overrightarrow{h_j}$ respectively,

and $\overrightarrow{P_{old}}$ is the current north pole. In principle, upright should be a slight adjustment for correcting the camera orientation, and need not drastically change the user's original intention of camera angles. We reflect this principle on the weights by giving a high weight ω_i^v to a vertical great circle passing nearby the current north pole, i.e., $\overrightarrow{v_i} \cdot \overrightarrow{P_{old}} \simeq 0$. Similarly, a high weight ω_j^h is assigned to a horizontal vanishing point $\overrightarrow{h_j}$ around the equator of the sphere. Weights ω_i^v are normalized by the variance σ_v^2, which measures the average divergence of vertical great circles from the current north pole. ω_j^h are similarly normalized by σ_h^2, which is the average divergence of horizontal vanishing points from the equator. n_v and n_h are the numbers of detected vertical great circles and horizontal vanishing points, respectively.

With Eq. (2), we iteratively update the north pole $\overrightarrow{P}$ using the result of the previous iteration as $\overrightarrow{P_{old}}$, where $\overrightarrow{P_{old}} = \overrightarrow{y}$ at the beginning. If the current north pole $\overrightarrow{P_{old}}$ is not correct, e.g., in the first iteration of a mis-oriented input spherical panorama, many of vertical great circles $\overrightarrow{v_i}$ are diverged from $\overrightarrow{P_{old}}$, and the variance σ_v^2 becomes large. Then, the relative differences among the weights ω_i^v are reduced so that vertical great circles $\overrightarrow{v_i}$ have similar influences on the optimization using Eq. (2). Consequently, $\overrightarrow{P}$ is determined by the majority of vertical great circles. On the other hand, when $\overrightarrow{P_{old}}$ is almost correct, many vertical great circles pass nearby $\overrightarrow{P_{old}}$, and σ_v^2 is small. In that case, vertical great circles $\overrightarrow{v_i}$ almost passing through $\overrightarrow{P_{old}}$ would have relatively higher weights ω_i^v than other less-accurate or outlier vertical great circles. A similar argument holds for horizontal vanishing points $\overrightarrow{h_j}$ and their weights ω_j^h. With this adaptive weight normalization, our iterative computation of $\overrightarrow{P}$ can quickly converge to an accurate solution. Note that the regularization term in Eq. (1) has been omitted in Eq. (2) as the weight computation using $\overrightarrow{P_{old}}$ provides similar regularization.

Iterative adjustments After we have resampled a cubemap from an input spherical panorama image using the rotation computed for upright adjustment, vertical/horizontal line segments would be classified more correctly. Thus, to obtain a more accurate upright adjustment result, we repeat the optimization process using Eq. (2), where the output of the current iteration becomes the input of the next iteration. As the iteration goes, the great circles from vertical/horizontal line segments become more accurate, and the north pole can be determined more accurately by minimizing Eq. (2), as shown in Figure 6. In our experiments, we set the maximum iteration as 10 times, but our optimization usually converges within four iterations.

Algorithm 1 Upright of spherical panorama

```
procedure EstimateRotationMatrix
    I ← Input image
    R ← Identity matrix
    repeat
        C ← ResampleCubemapWithRotation(I, R)
        [V, H] ← FindGreatCircle&VanishingPoint(C)
        P⃗ ← (0, 1, 0)
        repeat
            P⃗ ← FindPole(P⃗, V, H)                ▷ Eq. (2)
        until P⃗ has not been updated anymore
        R ← MakeRotationMatrix(P⃗)
    until R has not been updated anymore
    return R
```

Algorithm 1 summarizes the overall process of our upright adjustment method for spherical panorama images. We first resample a cubemap from an input image. We then find vertical great circles and horizontal vanishing points using the methods described in Section 4 with our outlier handling which remove mis-classified 3D horizontal lines. Finally, we iteratively update the north pole $\overrightarrow{P}$ using Eq. (2), and compute the rotation matrix R from the converged $\overrightarrow{P}$. The whole process is repeated until R does not change anymore.

6 Results

Parameters and computation time Our method has parameters, α, β, and γ in Eq. (1) and Eq. (2). We empirically validated that adjustment results are not sensitive to these parameters. We used $\alpha = 1.0$, $\beta = 3.0$, and $\gamma = 10.0$ for all experiments in the paper. We implemented our method using C++ and the OpenCV library on a 64-bit Windows PC with an Intel i7-6700K 4.00GHz CPU and 32GB RAM. For an input image of size 5376 × 2688, it takes a few hundred milliseconds (less than one second) to obtain the final rotation matrix R for upright adjustment.

Evaluation To evaluate the accuracy of our method, we used 14 ground-truth images, which were carefully taken with no tilt and roll of the camera using a tripod with bubble level. To diversify our dataset, we applied various rotations to each of the ground-truth images in ten random directions with six different angles. The amount of six angles consists of 5°, 10°, 15°, 20°, 25°, and 30°. As a result, we have 14 (images) ×10 (directions) ×6 (angles) = 840 test images.

For quantitative evaluation, we measured the angular error between an estimated rotation and its corresponding ground-truth rotation in the dataset. We depict the cumulative histogram of angular errors of our upright adjustment in Figure 7. Our method shows consistent performance regardless of the amount of initial

	Pole Estimation	Average Error	GC, VP Detection	Average # of Iterations
ours (basic)	0.039 ms	2.36°	215 ms	5.15
ours (handler)	0.064 ms	1.79°	204 ms	4.89
ours (weight)	0.076 ms	1.36°	185 ms	4.23
ours (handler+weight)	0.056 ms	1.29°	175 ms	4.11
ours (no resample)	0.013 ms	4.55°	46.7 ms	1.00
Bazin et al. [2]	420 ms	3.66°	4942 ms	1.00

Table 1 Numerical comparison of different upright adjustment methods. We used our ground-truth dataset for all experiments. We measured the north pole estimation time, average angular error, great circle detection time (include cubemap conversion), and the average number of iterations. Ours (basic) is the result of Eq. (1), and Ours (handler) is the result of Eq. (1) with misclassified 3D horizontal line removal. Ours (weight) is the result of Eq. (2) without misclassified 3D horizontal line removal. Ours (handler + weight) is our final result with both outlier handling methods. Ours (no resampling) is the result of Ours (handler + weight) before resampling, i.e., the result of the first iteration. The result of Bazin et al.'s method is also included for comparison. Ours (no resample) and Bazin et al. do not resample the input image as no iterative improvement is applied.

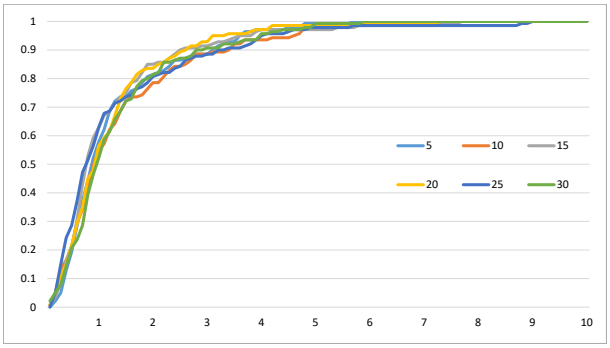


Fig. 7 Cumulative histograms of the angular errors of our method. We acquired ground truth images using a carefully installed camera tripod with bubble level. The images were rotated in random directions by the angles of 5, 10, 15, 20, 25, and 30 degrees to obtain a set of test samples. More than 90% of the results have less than 3 degrees of errors. (x-axis) Error in degree; (y-axis) The proportion of images in the dataset.

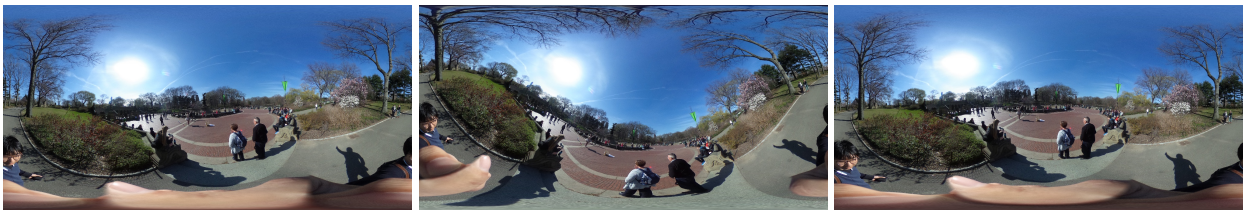
angular errors. For more than 90% of the test images, our method achieves upright adjustment with the errors less than 3°. This result demonstrates the robustness and effectiveness of our method.

Comparison For comparison, we performed the same evaluation on Bazin et al.'s algorithm [2]. For Bazin et al., we used the MATLAB code which is released by the original authors with the recommended parameters. In addition, we performed the experiments on the variants of our method to verify the effectiveness of our outlier handling methods. Table 1 shows the result of each upright adjustment method. For different versions of our method, average errors are reduced as we add our outlier handling methods. With our misclassified 3D horizontal line removal and adaptive weight normalization, each iteration estimates a more accurate north pole direction and it makes our algorithm converge quickly. Therefore, our final method achieves the best performance in terms of both accuracy and total computation time among all variants.

For Bazin et al.'s method, it works reasonably well when input images satisfy the Manhattan world assumption, but it fails when the assumption is not satisfied or an image contains many natural structures. Therefore it totally failed more than 10% of our dataset. Figure 8 shows typical examples of the failure cases of Bazin et al. Regarding the algorithmic aspect of Bazin et al's method, it detects many great circles without vertical/horizontal line clustering and outlier handling, and uses a RANSAC approach to find a solution for rotation estimation, incurring heavy computation. With our line clustering based great circle detection and outlier handling, as well as direct optimization of a quadric cost function, our method is more accurate and computationally efficient than Bazin et al.(Table 1).

Horizontal line constraint Upright adjustment can be achieved if we have two accurate great circles corresponding to 3D vertical lines. However, in practice, it is difficult to directly extract accurate vertical great circles from an input spherical panorama. To resolve this, we have the constraints using both horizontal and vertical line segments in our cost function for upright adjustment. At the first iteration of our upright adjustment, the constraint from the vanishing points of horizontal lines plays the role of a regularizer that guides vertical line segments to determining the correct north pole. As the iteration goes, the vertical line segments become more accurate (Figure 6), and the constraints from horizontal lines help improve the accuracy of the computed north pole rather than regularizing. Figure 9 demonstrates that the accuracy of upright adjustment is improved as the horizontal line constraint is included in Eq. (1).

More example results Figure 11 shows various examples of our upright adjustment of spherical panoramas. Our method works well even when horizontal and vertical line segments are not clearly identified in the input images. The supplementary material includes natural



(a) Input image (b) Result of Bazin et al. [2] (c) Our result

Fig. 8 Failure case for Bazin et al.'s method. It suffers from natural structures like trees, which yield many outlier great circles. On the other hand, our method is more robust against such outliers because our great circle detection using spherical Hough transform can exclude small line segments extracted from tree branches.

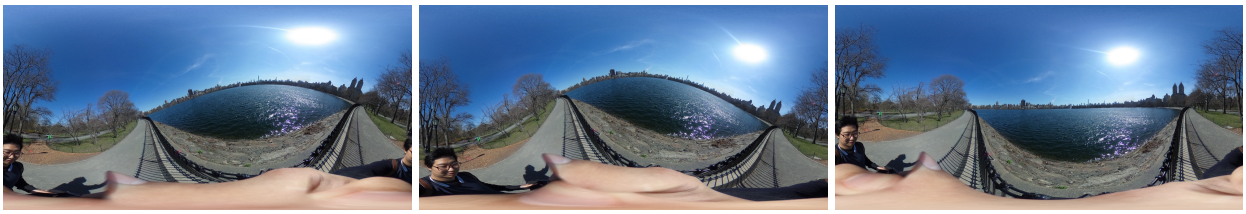


Fig. 9 Effect of the horizontal line term in our cost function. In a scene without vertical lines (left), using only the vertical line term would not succeed (center). This case is handled by including the horizontal line term (right).

scenes with successful upright adjustment results, although our method assumes Atlanta worlds with man-made objects. Successful upright adjustment results for blurred images can also be found in the supplementary material. Furthermore, our algorithm can be extended to slanted video and make it look comfortable and stable adjust using upright adjustment.

Failure cases Figure 10 shows failure cases of our method. In the case of top row, the input image does not follow the Atlanta world assumption. There are many diagonal lines on the ceiling, which hinder our algorithm. In the case of bottom row, the input image contains dense nature structures. There are many short lines on trees, and they cause false positive great circles. If the input image does not satisfy the Atlanta world assumption or has very few positive great circles, our method could fail to achieve accurate upright adjustment results, even though there are several successful examples for such cases presented in the supplementary material.

7 Conclusion

With the increasing popularity of consumer 360 cameras and VR headset devices, it is important to provide comfortable and pleasant experience on consuming spherical panorama contents while user-created 360 contents are prone to be mis-oriented, which makes VR experience unpleasant. In this paper, we have presented upright adjustment for 360 spherical panoramas which can greatly improve VR experience. From noisy edge information in an image, we cluster the horizontal and vertical lines and estimate great circles by spherical Hough transform. Then, we estimate the north pole direction as an upright position of the image through iterative optimization. We handle outlier lines robustly by using vanishing points of horizontal lines and stabilize the convergence by adaptive weight normalization. Our method is computationally efficient and works fast even with our unoptimized implementation. With our method, accurate upright adjustment of 360 spherical panoramas can be achieved to provide comfortable viewing experience. Our method could also be used as a pre-processing step to calibrate the initial orientation in various 360 VR applications, such as 360 image editing and video stabilization.

There are few directions for improving our method. We approximate spherical arcs by clustering line segments with slanted angle, and it causes iterative adjustments accordingly. Direct detection of spherical arcs would make the computation of great circles more robust, which can lead to improving the accuracy of our method. For simplicity, we use rasterization over the unit hemisphere to determine the intersections of great circles for vanishing points detection. Analytic intersection computation of great circles would also improve the accuracy and speed of our method.

Acknowledgements

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant (R0126-17-1078), the National Research Foundation of Korea (NRF) grant (NRF-2014R1A2A1A11052779), and Korea Creative Content Agency (KOCCA) grant (APP-0120150512002), funded by the Korea government (MSIP, MCST).

(a) Input image (b) Our result (c) Expected result

Fig. 10 Our failure cases.

Fig. 11 Our upright adjustment results for 360 spherical panoramas. (top) input images. (bottom) our results.

References

1. Akinlar, C., Topal, C.: Edlines: A real-time line segment detector with a false detection control. Pattern Recognition Letters **32**(13), 1633–1642 (2011)
2. Bazin, J.C., Demonceaux, C., Vasseur, P., Kweon, I.: Rotation estimation and vanishing point extraction by omnidirectional vision in urban environment. The International Journal of Robotics Research **31**(1), 63–81 (2012)
3. Bosse, M., Rikoski, R.J., Leonard, J.J., Teller, S.J.: Vanishing points and 3D lines from omnidirectional video. In: Proc. IEEE International Conference on Image Processing (2002)
4. Gallagher, A.C.: Using vanishing points to correct camera rotation in images. In: Proc. Canadian Conference on Computer and Robot Vision, pp. 460–467. IEEE (2005)
5. von Gioi, R., Jakubowicz, J., Morel, J.M., Randall, G.: LSD: A fast line segment detector with a false detection control. IEEE Transactions on Pattern Analysis and Machine Intelligence **32**(4), 722–732 (2010)
6. He, K., Chang, H., Sun, J.: Content-aware rotation. In: Proc. IEEE International Conference on Computer Vision, pp. 553–560 (2013)
7. Kamali, M., Banno, A., Bazin, J.C., Kweon, I.S., Ikeuchi, K.: Stabilizing omnidirectional videos using 3d structure and spherical image warping. In: Proc. IAPR Conference on Machine Vision Applications, pp. 177–180 (2011)
8. Kopf, J.: 360 video stabilization. ACM Transactions on Graphics (TOG) **35**(6), 195 (2016)
9. Kopf, J., Lischinski, D., Deussen, O., Cohen-Or, D., Cohen, M.: Locally adapted projections to reduce panorama distortions. Computer Graphics Forum **28**(4), 1083–1089 (2009)
10. Lee, H., Shechtman, E., Wang, J., Lee, S.: Automatic upright adjustment of photographs with robust camera calibration. IEEE Transactions on Pattern Analysis and Machine Intelligence **36**(5), 833–844 (2014)
11. Scaramuzza, D.: Omnidirectional vision: from calibration to robot motion estimation. Ph.D. thesis, ETH Zurich (2008)
12. Schindler, G., Dellaert, F.: Atlanta world: An expectation maximization framework for simultaneous low-level edge grouping and camera calibration in complex man-made environments. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition, pp. 203–209 (2004)
13. Wang, X., Cao, X., Guo, X., Song, Z.: Beautifying fisheye images using orientation and shape cues. In: Proc. ACM International Conference on Multimedia, pp. 829–832 (2014)
14. Wang, Z., Jin, X., Xue, F., He, X., Li, R., Zha, H.: Panorama to cube: A content-aware representation method. In: Proc. SIGGRAPH Asia 2015 Technical Briefs, SA '15, pp. 6:1–6:4 (2015)