Integrated Google Maps and Smooth Street View Videos for Route Planning

Chi Peng
National Taiwan University
Legault@cmlab.csie.ntu.edu.tw

Bing-Yu Chen
National Taiwan University
robin@ntu.edu.tw

Chi-Hung Tsai
Institute for Information Industry
brick@iii.org.tw

***Abstract*—In this paper, we provide a system that takes start and end points as the input, and will automatically connect to the Google Maps with Street View to achieve the route planned result and scenery along the route. Finally, it will generate a smooth scenic video from the starting point to the destination, which combines the Google Maps to provide better route recognition to users. The users can watch and interact with the video as if they are driving a car through the planned route. Our system is fully automatic but still provides interaction mechanisms for the users to walk around in the scenic video.**

Keywords-Google maps; street view; scenic video; route planing; blending; boundary cut

I. INTRODUCTION

With the development of technology, route planning software is popular these days. Users can now plan their driving directions by using online route planning services. Most of the time they only need to input their starting point and destination, and then they can get an electronic map which describes the full route from the starting point to the destination. However, they could not know the scenery along the route before they drive on it, so when they actually drive along the way, it is sometimes hard to distinguish the route in the street they are not familiar with.

To map the virtual map and the real scenery, Google provides a service called Google Maps with Street View[1], by which users can watch the scenery in the street as if they are driving on it. In addition, they can also plan their route by Google Maps, and then watch along the route with the Street View to get a better feeling about the view. However, because Google Maps with Street View only provides a rather sparse point-to-point system to view the street, the process of viewing from the starting point to the destination could not present smoothly, but in a way more like jumping from a point to another point.

To provide a much better service while simplifying the problem, some methods are provided to use their own street view data to make smooth transitions. However, those data are hard to obtain by ordinary users. Moreover, it is also inconvenient to use Google Maps with Street View to watch a route, because the users need to keep pressing the "forward" button to jump from one panorama image to the next one step by step. Hence, in this paper, we provide an easy-to-use system based on pubic but not so good Google Maps with Street View data to provide a smoother guiding video. By using our system, the users can interactive with the scenic video of the planed route easily. Thus, our system can provide a better feeling of the scene of the planned route and the users can interact with our system to watch the part they are interest, or skip those with less importance.

In summary, the main contribution of our work is that we introduce a system which only takes some simple inputs from the users and accesses the data from Google to generate a visually smooth route guiding video. Combined this video with Google Maps, we provide a guiding system to let the users familiarize themselves with their planned routes before actually driving on them, thus makes them easier to follow the pre-planned route.

II. RELATED WORK

Typically, large collections of images are usually shown as a page of thumbnails or a slideshow, which is a practical but not engaging way, since it is lacking of the feeling of "being there", which is critical to street view images. The "Photo Tourism" [8] is a successful example, which puts all images of a given scene in a common 3D space using bundle adjustment. Users then can browse the image collection by moving in the 3D space freely. However, when browsing from one image to another, the transition in the system is not smooth enough to keep on the feeling of really moving into the next image.

Photorealistic "Virtual Space" [7] connects visually similar images together to create a virtual space which users are free to move from one image to the next one using intuitive 3D controls such as move left/right, zoom in/out and rotate. It displays the images in correct geometric and photometric alignment with respect to the current photo, so the transition between the images is smooth. The infinite zoom (in) effects are visually "walking into" the images, which inspired our work.

In [2], Chen *et al.* combined the street view video with a map to enhance the path finding ability of users. However, in this system it requires intensive panorama images along the planned route to create a smooth video tour. Recently, Kopf *et al.* [5] combined the images in the street view system by stitching the side views, which means standing on the street and looking to the left or right, of a certain street together to generate a long street slide for users to quickly browse this street. While it is an excellent way to view the side scene of a certain street, it keeps the users look to the side of the street,

[1] http://maps.google.com/help/maps/streetview/

which is usually not the case while actually driving or walking.

To combine images, matching and blending can smoothly integrate several related images together, which have already been used to generate panorama images [1]. In our system, first we use the image matching technique, SIFT, to match each tile of the panorama image, and then use the technique of minimum error boundary cut [3] to stitch them together. The minimum error boundary cut treats the difference of the junction area between two images as an intensity map, and then calculates the shortest path, with intensity as the cost, to cut through the intensity map, so that the stitched images will have a less obvious seam between the two original images. Finally, we use Poisson image editing [9] to blend the boundary of the two images. The idea of Poisson image editing is approaching the Poisson partial differential equation with Dirichlet boundary conditions which specifies the Laplacian of an unknown function over the domain of interest, along with the unknown function values over the boundary of the domain.

III. SYSTEM OVEWVIEW

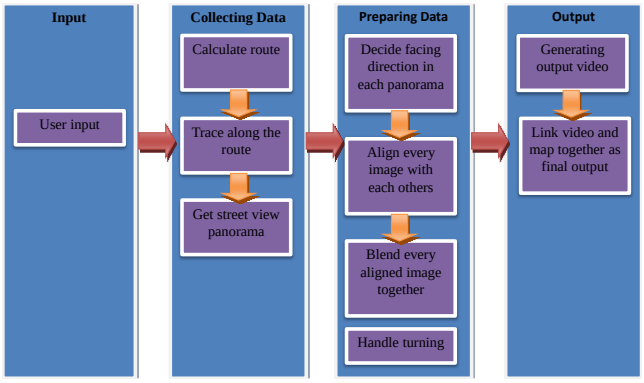


Figure 1. System overview.

The basic steps of our system are shown in Figure 1, which are:

1) User input: Let users input the starting point and the destination of their trips.

2) Calculate route: Connect to Google Maps to calculate the route of the trip, and get a flowchart of this planned route.

3) Trace along the route: Use Google Street View to trace all the way from the starting point to the destination according to the flowchart, and record every panorama ID along the way.

4) Get street view panorama: Download all panorama images and their information according to the panorama ID.

5) Decide facing direction in each panorama: Decide the looking directions of those panorama images, and save them. Thus, for each panorama image, we have one image which is the basically one the users will see in the final video.

6) Align every image with each others: Scale each image taken in Step 4 and align it with the image before it, since "moving forward" is somewhat like "zoom in".

7) Blend every aligned image together: Blend all images while stitching them to reduce the perspective discontinuity visually. By this way, we can link all images together.

8) Handle turning: Turning is different from moving toward, and needs to be handled specially when we stitched the images of the turning point together to create the turning effect.

9) Generate output video: Zoom in from the first image toward the second one which stitched in it while alpha blending the two images at the same time. After we zoom in to the second image, do the same thing with the third image, etc. Some special cases are also needed to be handled like turning at this time. After this step, the route guiding video is generated.

10) Link video and map together as the final output: Link the scenic video with Google Maps to create our final direction guiding system.

IV. GOOGLE MAPS WITH STREET VIEW

A. Data from Google Street View

We can use the following URL to download the panorama image from Google Street View:

http://cbk1.google.com/cbk?output=tile&panoid="PANO_ID"&zoom="Z"&x="X"&y="Y"&cb_client=api&fover=2&onerr=3&v=2

where "PANO_ID" is the panorama ID of this particular panorama image, and "Z" is used to indicate the resolution (here we use 3). Because Google stores each panorama as several image tiles, we need to download all tiles and stitch them together to reconstruct the original panorama image. While "Z" equals to 3, "X" is ranged from 0 to 6, and "Y" is ranged from 0 to 3, so there are totally 28 tiles, and each tile is a 512 x 512 image.

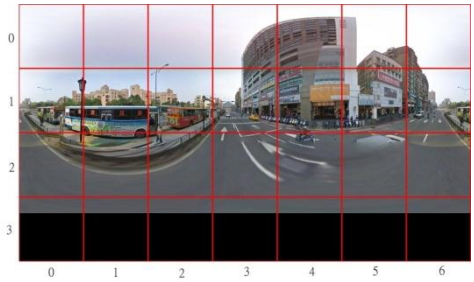


Figure 2. Combined panorama image with each tile separated by red lines.

After stitching these 28 image tiles together, we can reconstruct the panorama image we downloaded with the panorama ID as one single image, but as shown in Figure 2, this panorama image has some repeat parts between the X = 0 and X = 6 tiles, and there is nothing below the Y = 3 tiles.

After some experiments, we found that the width of the repeat part is 256 pixels, and the height of the black line is 384 pixels. This holds true for all panorama images downloaded from Google Street View with zoom = 3. After cropping those parts, we can get our panorama image.

We can also use the following URL to download the XML file which describes the information of the panorama image:

http://cbk1.google.com/cbk?output=xml&panoid="PANO_ID"&cb_client=api&pm=0&ph=0&v=2

where "PANO_ID" is the panorama ID of this panorama image. In this XML file, it records the information of the corresponding panorama images, which are:

- <lat>,<lng>: The latitude and longitude position of this panorama image.
- <pano_yaw_deg>,<tilt_yaw_deg>,<tilt_pitch_deg>: How this panorama image fits to a spherical texture.
- <annotation_properties>: It records the <pano_id> and <link yaw_deg>, which are the ID and direction of its neighboring panorama images.

B. *Data from Google Maps*

Similar to Google Street View, we can access the route planned by Google Maps directly by the following URL:

http://maps.google.com/?saddr="START_LOCATION"&daddr="END_LOCATION"&output=kml

where "START_LOCATION" is the starting point, and "END_LOCATION" is the destination. By assigning "output=kml", Google Maps will return the route information as an XML file, which will be easier for us to make use of.

In the XML file, <Placemark> is important to us, because it records every important key step in the planned route. The key steps are the vital part of this route, like where the starting location and the destination are, and where to make a right or left turn. In each of these key steps, we will pay special attention to the nodes described below:

- <name>: The description of this step, which also states that the route is making a turn at this point or not.
- <longitude>,<latitude>: The longitude and latitude position of this key step.
- <heading>: The degree of the direction of the next panorama image. If this key step is the turning point, we need to consider it to calculate the next panorama image along the route.

By the information we gathered now, we can trace the whole route from the starting location to the destination, and download all panorama images and their XML files.

V. STITCHING PANORAMA IMAGES

A. *Preparing Data*

Since panorama images downloaded from Google Maps with Street View are all spherical panoramas, first we need to calculate the normal viewing scene from them. In the implementation, we use Direct3D to draw a sphere around the camera and apply the spherical panoramas as a texture image on it. Notice that the camera needs to be adjusted by the three values of <pano_yaw_deg>, <tilt_yaw_deg>, and <tilt_pitch_deg> we recorded before to make it correct and coincidence with the <heading> value. Tracing through the planned route, we transform every panorama image to pictures with the camera facing our heading direction.

Figure 3. The output images at a turing point.

As for the turning point, first we take the image with the heading direction opposite to where we came from as the starting image. Then, because we are using a field of view of 60 degree in our system, we take the next image after turning 60 degree and stitch it with the initial one. Then, we continue turning the camera until just one final turn of the camera before we are facing the direction of the next panorama, where the final turning degree may not be another full 60 degree. Thus, while stitching it with all previous images, we need to overlay the previous images, and the overlay region is inverse proportional to the turning degree of the last turn. Figure 3 shows an example of the camera movement at a corner.

B. *Image Alignment and Matching*

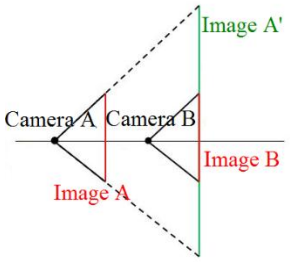


Figure 4. The zooming correspondence between Image A and B.

Now we have every single image along the planned route, next we are going to link them together one by one. We assume that the effect of "moving forward" can be simulated by the effect of "zoom in". For example, if we stand on Image A and moving forward to Image B, then there should be a way to scale Image B down and match it

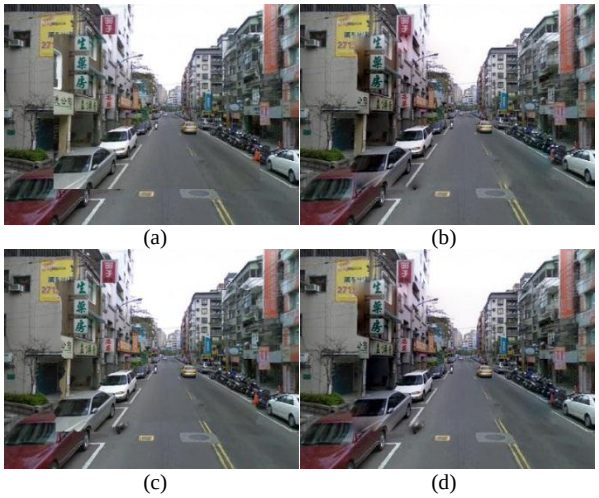
to somewhere in the middle of Image A. Figure 4 shows the zooming correspondence between the two images.

To align Image A and Image B, Scale-invariant feature transform (SIFT) [6] is used. Then, we use RANdom SAmple Consensus (RANSAC) [4] to find the transform function to match the Image B into the middle of Image A. Under the current data accuracy, we found that doing the translation and scaling are sufficient. More complex transformation might only yield worse result. However, since the SIFT features are not completely match between these two images, sometimes the result is not what we expected. With some experiments, we found that by setting an upper bound (0.7) for scaling, the result will be more stable.

Figure 5 (a) shows the pasting result. Notice that the seam between the images is still obvious, and this will be smoothed in the next section. Due to the sudden change of perspective, some information is lost, but most of the scene is still intact thus will not affect the overall feeling of the scene.

C. Image Blending and Stitching

In order to paste one image to another in a seamless way, Poisson blending is usually used. Figure 5 (b) shows the simple result, although it is not so good. To further reduce the seam between the images, the minimum error boundary cut [3] is used. By this algorithm, we can find the best crop line between the images to make the artifact of blending even less obvious. Because we paste one image into the middle of the other, we only need to do the minimum error boundary cut around the four sides of the pasting and pasted images as shown in Figure 5 (c). By using the Poisson blending to further smooth the boundary, a much more smooth result can be achieved as shown in Figure 5 (d).



(a) (b)

(c) (d)

Figure 5. (a) Embeding one image to another by SIFT features. (b) Poisson blending the two images in (a). (c) Embeding with minimum boundary cut. (d) Embeding with minimum boundary cut and Poisson blending.

Now we know how to combine two adjacent images with each other. To link all the images along the planned route, we simply start from the last image, and backtrack every image and do all the works above with them one by one. With so many images stacked together, the seams between the images become visible due to too many seams appear in one image, and they are too close to each other as shown in Figure 6 (a).

To counter this problem, before pasting one image to its previous one, we only keep track of the scaling and translation between them, but not pasting them immediately. Once the scaling change is bigger than a pre-defined threshold, we then do the pasting. In practice, we found that the threshold works well with a value of 0.6, which means the image will be scaled at least 60% before it is pasted. Figure 6 (b) shows the result and comparison.

(a) (b)

Figure 6. (a) Embedding all images together. (b) The result of removing some images before pasting.

Finally we will cope with the turning point. While we are backtracking and matching images, if we encounter the turning point, we can just match the image to the end camera position of the turning point. After that, we crop the starting camera position to create another image, thus we can continue the image string by matching it with the image which is just before the turning point. Due to the fact that the perspective change during the turning point might be more serious (some buildings may block the view at the corner), we will not skip the images adjacent to the turning point.

VI. SYSTEM INTEGRATION

A. Generating Video

Now we are going to generate the guiding video from this image string created before. Basically we will keep enlarging the first image toward its pasted second image, and make the part where contains the second image do an alpha blending with the real second image. When the part of second image contained in the first image covers the whole scene, we will switch to the second image and do the same thing with the third image. Minimum error boundary cut makes the seams between the images less obvious even when the weight of the next image is larger, which means we are close to switch to the next image. Continue doing this until we go through the image string, then we will have the output guiding video of the planned route.

At the turning point, we similarly zoom in toward the next pasted image. Although we can stand still while turning, the aim of our system is to simulate as driving a car. Hence,

we will keep moving forward and turning at the same time. We also need to keep track at which frame we switch the images, this information will be useful in the next part.

Note that while generating the video, we only take the center 80% size of each zoomed image as the output video. Because of the boundary cut, we need to make sure when we switch to the next image, that image must be able to cover the whole frame without holes caused by the boundary cutting.

B. *Integrating with Google Maps*

Finally, we combine the video with Google Maps to create a guiding system which users can easily navigate through their planned routes. To use this system, the users only need to input their starting location and the destination, and then the system will do all the above processes to give them the output as shown in Figure 7. In Figure 7, the left part is our guiding video, and the users can watch the video and pause anytime they want. The slider below the video also provides a quick navigation along the route, so the users can navigate the whole path at their own pace. The "View" button will bring up a "Street Viewer" window which contains the nearest panorama image of the current frame of the guiding video. The users can rotate the camera of the Street Viewer in case they want to watch the scene more closely.

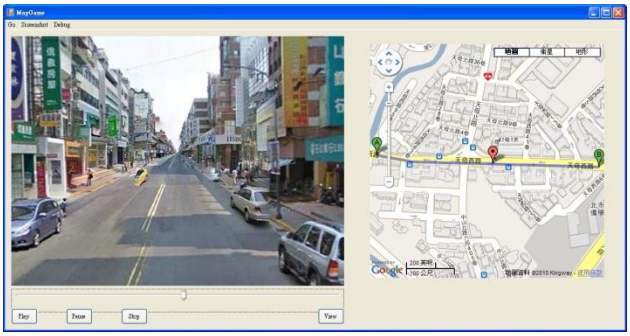


Figure 7. The screenshot of our system.

The right part of the GUI is the Google Maps integrated with our video. Utilizing the coordinates we recorded with every panorama image and the information about which frame we switch to the next image in the video, we can calculate the current position by interpolating the coordinates of the current image and the next one. Combining the guiding video with the Google Maps, the users can get visual feeling of any point within the planned route, and at the same time link every point on the map with its actual scene with ease. Thus, our system can provide a driving experience to the users even before they actually go to the spot themselves.

VII. DISCUSSION

Here we compare our system with the original Google Maps with Street View as well as a recent system [2] also dealing with the street view as shown in Table I.

TABLE I. THE COMPARISON BETWEEN SYSTEMS

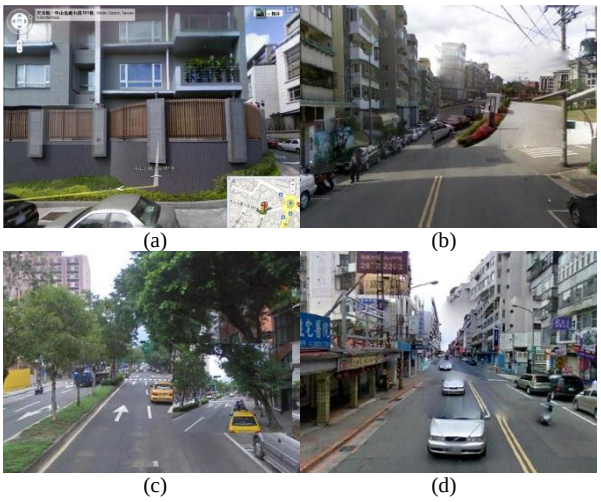
	Google Maps with Street View	Integrated Videos and Maps for Driving Directions [2]	Our System
Moving Smoothness	X	O	O
Video Smoothness	X	O	O
User Interaction	O	X	△
Easy Data Access	O	X	O
Route Recognition	△	O	△

- Moving Smoothness: How camera moves from one panorama image to the next one. Google Maps with Street View suffers from the flaw stated before, and [2] counters this problem by using intensive panorama images, while our system uses the techniques described in the previous sessions to solve it.
- Video Smoothness: In Google Maps with Street View, users need to press the "next" button again and again to move from the starting location to the destination step by step, while [2] and our system both have automatic playing function.
- User Interaction: Users in Google Maps with Street View have fully control ability of where to go and where to look at, while in [2], users can only follow a predefined route and viewing. Although our system also makes the users follow a predefined route, they can watch their surroundings freely by pressing the "view" button.
- Easy Data Access: Google Maps with Street View is opened for any user, and our system also uses the same data. However, [2] uses its own panorama images, which are not obtainable by ordinary users.
- Route Recognition: Our system shares the same map and scene from Google Maps with Street View, but [2] highlights important landmarks on the map and provides a better way for the users to recognize their planned routes.

Still, there are some cases that our system cannot work very well with. The first case is that some information about the street view we obtained from Google Maps with Street View is not very accurate. For example, sometimes the neighboring panorama is not recorded correctly in the XML file, as shown in Figure 8 (a). Since the relation between these panorama images are wrong, our system could not generate a reasonable output. Another problem caused by the original data is that the next image might have a biased view without any reason, and this problem is not recorded in its XML file. As shown in Figure 8 (b), in this case the alignment will be completely wrong due to the different viewing angles. The above two failure cases are due to the error of the original data obtained from Google Maps with

Street View. That means to use the real data to build a useful system needs a lot of efforts to deal with the original data error.

Another failure case comes from the street trees, because it is very hard to find the corresponding feature point pairs of the street trees in two corresponding images due to the tree leaves. If the image is covered by too much trees, our system may not be able to handle it well as shown in Figure 8 (c). In Figure 8 (d), it shows two problems: one is the cars moved with the scene which will appear in several images repeatedly. Since the street view data is sparse, it is hard to detect the moving objects and remove them. The other problem is the curved road. Due the distance between each panorama the curved road could not be sampled perfectly. The difference between the sampled curved road and the real road will cause the seams and inconsistence as shown in Figure 8 (d).



(a) (b)

(c) (d)

Figure 8. Failure cases. (a) Original data error. (b) Inconsistancy of viewing direction. (c) Alignment error caused by trees. (d) Curved road and vehicles in the scene.

VIII. Conclusion and Future Work

In this paper, we provide a guiding video system which is intuitive to use and requires data comparatively easy to obtain from Google Maps with Street View. Up now, the methods to deal with the moving between panorama images in the street view video is simply to increase the number of panorama images, which is usually unobtainable by ordinary users. Our system only takes normal density of panorama images to create the smooth guiding video. We combined image alignment and blending techniques to simulate moving forward by the zoom in effect, and generate a visually smooth guiding video. Finally, we integrate this video with Google Maps to create a guiding system which users can familiarize themselves with the scene of their planned routes before they actually drive on them.

Our system processes all the panorama images after we apply them to a sphere texture and take them as a screenshot, but theoretically we could process the panorama images directly. With proper cropping and transformation, it is possible to create the image string with the original panorama images. Then, we can generate a fully interactive virtual space with all the real panorama images, and move freely in it smoothly.

Another possible improvement with our system is the alignment part. SIFT cannot cope with trees well. The image alignment is vital to the performance of our system, thus further research in this part could yield better result than the current system. For now, our system took about five seconds to download one panorama image from Google Maps with Street View, about five seconds to align it with other images, and about five seconds to do the Poisson blending. Further research will be needed to reduce the time complexity.

Finally, indicating landmarks in our system could also help with route recognition. Integrating labels of landmarks in both of the map and the guiding video system may make users to know which scene they should pay particular attention to, although this idea still requires further research.

Acknowledgment

This paper was conducted under the III Innovative and Prospective Technologies Project of the Institute for Information Industry which is subsidized by the Ministry of Economy Affairs of Taiwan and also partially supported by the Excellent Research Project of the National Taiwan University under NTU98R0062-04.

References

[1] M. Brown and D. G. Lowe, “Recognising Panoramas,” Proc. Intl. Conf. on Computer Vision 2003, pp. 1218-1225.

[2] B. Chen, B. Neubert, E. Ofek, O. Deussen, and M. F. Cohen, “Integrated Videos and Maps for Driving Directions,” Proc. Symp. User Interface Science and Technology 2009, pp. 223-232.

[3] A. A. Efros and W. T. Freeman, “Image Quilting for Texture Synthesis and Transfer,” Proc. ACM SIGGRAPH 2001, pp. 341-346.

[4] M. A. Fischler and R. C. Bolles, “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography,” Comm. of the ACM, vo. 24, no. 6, pp. 381-395.

[5] J. Kopf, B. Chen, R. Szeliski, and M. Cohen, “Street Slide: Browsing Street Level Imagery,” ACM Trans. Graphics, vo. 29, no. 4, 2010, Article no. 96, (Proc. SIGGRAPH 2010).

[6] D. G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints,” International Journal of Computer Vision, vol. 60, no. 2, 2004, pp. 91-110.

[7] J. Sivic, B. Kaneva, A. Torralba, S. Avidan, and W. T. Freeman, “Creating and Exploring a Large Photorealistic Virtual Space,” Proc. IEEE Workshop on Internet Vision 2008.

[8] N. Snavely, S. M. Seitz, and R. Szeliski, “Photo Tourism: Exploring Photo Collections in 3D,” ACM Trans. Graphics, vo. 25, no. 3, 2006, pp. 835-846, (Proc. SIGGRAPH 2006).

[9] P. Pérez, M. Gangnet, and A. Blake, “Poisson Image Editing,” ACM Trans. Graphics, vol. 22, no. 3, 2003, pp. 313-318, (Proc. SIGGRAPH 2003).