A Real-Time Rendering Technique for View-Dependent Stereoscopy Based on Face Tracking

Anh Nguyen Hoang[1], Viet Tran Hoang[1], and Dongho Kim[2]

[1] Soongsil University
{anhnguyen,vietcusc}@magiclab.kr
[2] 511 Sangdo-dong Dongjak-gu, South Korea
cg@su.ac.kr

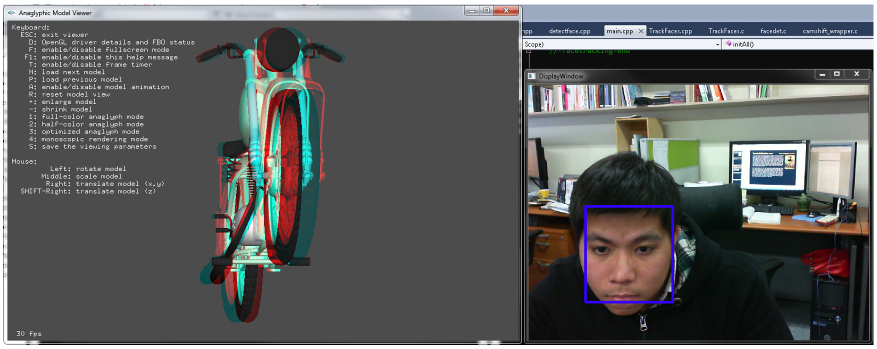
Abstract. In our research, we propose and implement a virtual reality system with the common and widely used devices such as 3D screen and digital webcam. Our approach involves the combination of 3D stereoscopic rendering and face tracking technique in order to render a stereo scene based on the position of the viewer. Our approach is to calculate the offset values of face position to assign to the virtual camera position relatively. We employ a technique to change the typical symmetric frustum into asymmetric to achieve the head-coupled perspective. With our system, the rendered scene observed by human eyes remains realistic and the viewport can be seen as the physical window in the real environment. Therefore, the right perspective can be maintained regardless of viewer position.

Keywords: View-dependent, asymmetric frustum, stereoscopy, face tracking.

1 Introduction

Virtual reality came into existence during the early 1960s. It has been developed and applied widely in recent years because of the development of computer in both hardware and software. A virtual reality system at least consists of these kinds of devices: a computer, input devices for position or gesture tracking, and output devices for displaying information to the user. For a simple virtual reality system, it is typical to implement using a common desktop computer with a digital webcam to track the viewer's face and adjust the 3D rendered scene appearing in the system window. Virtual reality is often used to deliver the visual contents in various application forms which are mostly associated with 3D stereoscopic environment. Therefore, using 3D displays such as interlaced screens or anaglyph with red-cyan glasses would be a simple and appropriated approach for our system.

Stereoscopic rendering has been researched and implemented comprehensively recently. The techniques for stereo display have been applied in common usages such as the mass production of 3D screen with polarized glasses or the 3D Vision



```
Keyboard:
  ESC: exit viewer
    D: OpenGL driver details and FBO status
    F: enable/disable fullscreen mode
   F1: enable/disable this help message
    T: enable/disable frame timer
    N: load next model
    P: load previous model
    A: enable/disable model animation
    R: reset model view
    +: enlarge model
    -: shrink model
    1: full-color anaglyph mode
    2: half-color anaglyph mode
    3: optimized anaglyph mode
    4: monoscopic rendering mode
    S: save the viewing parameters

Mouse:
         Left: rotate model
       Middle: scale model
        Right: translate model (x,y)
  SHIFT-Right: translate model (z)
```

Fig. 1. Main interface of system. Stereoscopic rendering and face tracking.

equipment from NVIDIA. Using those kinds of technique as well as the 3D stereoscopic rendering fundamentally well-known theories, we are able to produce and transfer the 3D stereoscopic rendered scene to the viewer in real-time. It is required to keep our system as simple and applicable as it should be, we are not employing the expensive and high-technology devices, the cheap and accessible devices are employed instead. This is one of main principles in our system. Using the common devices for representing an interactive system between the viewer pose and 3D stereoscopic contents involves the high challenges in the technique to improve the performance of the face tracking as well as stereoscopic rendering.

View-dependent rendering is the terminology describing the changes of rendered scene regarding to the view of observer. In our system, view-dependent rendering is used to adjust the virtual camera position to display a scene dynamically with the viewer's face. View-dependent rendering was used mostly in terms of perspective change when the viewer is looking through the window. The virtual scene will be changed and displayed interactively. In many researches, the basic and simple techniques are being used to track the head position and by controlling the virtual camera position relatively to the viewer, the scenes are changed but they do not actually reflect the right human perspective in real environment. In these cases, the frustum defined by the graphic API will retain the normal symmetric shape and it therefore cannot present a right perspective to human perception.

In our system, we also present a tracking technique based on optical flow to track the viewer's face. In computer vision, face tracking is always an important and interesting research area. Face tracking in our approach is understood as the ability to retrieve the position of a human head relatively to the computer screen. Although it seems to be the challenges in computer vision, the recent works of researches have been introducing variety of face tracking processes with the high accuracy in real-time. We do not make a novel technique. Instead, we employ a high accuracy technique with a modification to enhance the tracking speed. More particularly, the face tracking technique will make the interactive recognition and response to the human perspective changes. The Fig. 1 gives a basic illustration for our system. As the result, when the human head is tracked and the frustum is in the arbitrary form, the rendered scene will appear with the right perspective to human view and the observed scene will look at most realistic.

2 Related Works

There are a number of previous researches which have the same purpose in implementation of system using view-dependent rendering. P. Slotbo [1] introduced fundamental theories of rendering a 3D stereoscopic scene. He also implemented system for 3D interactive and view-dependent rendering. The proposed concept in his research was implemented using two inexpensive off-the-shelf web-cameras and a low-end desktop computer. He used the information from cameras as the input to track viewer's position for setting up a view-dependent rendering. There were some makers used to detect the interactions of the user. The developed system showed the abilities to render the scene in real-time but non-stereo mode. With stereoscopy enabled, the system ran slowly in only 12 fps. It also tracked the viewer position but indirectly.

In another related study, Jens Garstka and Gabriel Peters [7] calculated a view-dependent 3D projection of a scene which was projected on flat surfaces. They implemented a system that enabled the single viewer to explore the scene while walking around. They introduced a novel approach for head tracking using depth-images retrieved from a Microsoft KINECT 3D sensor. Such device had done most of hard works in detection head's orientation. The delays in adaptation of projected scene remained as a limitation. Their system is the first right step of expandability to 3D stereoscopic view-dependent rendering.

Sebastien [4] presented a non-intrusive system that gave the illusion of a 3D depth and interactive environment with 2D projectors. His research involved the head-coupled perspective to give the user a 3D scene which was projected onto a flat surface. The projected image was such that the perspective was consistent with the viewpoint of the user. Kenvin [2] discussed characteristics of head-coupled stereo display; he also dealt with the issues involved in implementing head-coupled perspective correctly. His research gave a first and fundamental theory on head-coupled stereo display. Buchanan [8] proposed a view-dependent rendering set-up for home computer use. In his research, view-dependent rendering used a parallax effect to give the illusion of depth. His face tracking method was based on the Lucas-Kanade algorithm.

3D stereoscopic rendering technique has been known and implemented recently on many researches. Paul Baker [6] introduced a right method to stereoscopic rendering with an off axis setting for two offset cameras. His studies have provided straightforward and fundamental theories to 3D stereoscopic rendering. In 2011, Samuel Gateau from NVIDIA [3] presented that of technique with an overview but concise and comprehensive information. The concrete implementation for such system can be found in the course BYO3D [12] by MIT media laboratory. We utilize this work in our system as a part of handling the 3D stereoscopic rendering.

3 Technique Background

3.1 3D Stereoscopic Rendering

Stereoscopic rendering has been well studied and implemented widely by many researchers in computer graphics field. With the development of display technology and the power of Graphics Card, implementing a 3D stereoscopic render system in real-time is currently not a challenge. The systems involved that of technique always aim to create the correct illusion of 3D depth by presenting a left and right image to the human eyes. Human perception of depth will in turn proceed the visual information taken from two eyes. Therefore, if the left and right image are correctly sent to human eyes, the strong sense of depth can be presented to the viewer as seeing the real world.

There are many methods for viewing a 3D stereoscopic rendered frame. In our system, we use two methods to present the stereo images to the viewer. We employ the common and traditional red and cyan glasses for a rapid 3D display which is known as anaglyph, we also use another method called row-interlaced stereoscopy using 3D display screen with glasses. Anaglyph is an old approach but it is used to illustrate the basic principle of transmitting stereo images. Anaglyph does not represent the colours faithfully. The interlaced display can preserve the image colours but it has the restriction in view in position. The viewer must look the screen in a straight direction and only moves the head horizontally to observe the scene, otherwise the 3D stereo effects will no longer be available.

In the standard graphics API, the frame is rendered and viewed by projecting a virtual 3D scene onto the computer screen as a 2D image. A perspective projection performs perspective division to shorten and shrink objects. View frustum defines those parts of the scene that can be seen from a particular position which is camera position. In our approach with stereoscopy, we consider two cameras presenting two eyes and a single projection plane. The object will be projected differently based on the position of camera and the visibility of object will be considered as well. It can be thought of as a window through which we can see the world. It is obvious that when the viewer moves, the scene will change in order that there are some parts of world being seen from one position but not from other position, and the objects in scene are observed from different angles.

We want the system to render the scene view-dependently. It means that the scene will always have the right perspective to the viewer position. When the viewer moves, the 3D scene should be rendered in such a way that the viewer is observing the real objects. The rendered scenes will change in real-time but the scene perceived by the human eyes must remain stable. Using perspective projection, the scene will be scaled based on the depth, this projection will then be able provide a depth cue. Therefore, such kinds of projection are called on-axis because the points in frustum are chosen to be the points on the near plane along the z-axis which is perpendicular to the screen plane. That projection is implemented by most of graphics API. However, using it will result in various distortions of virtual scene

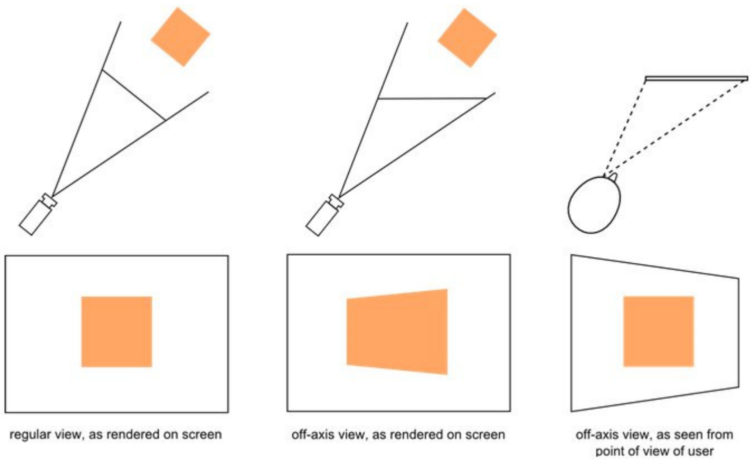

regular view, as rendered on screen
off-axis view, as rendered on screen
off-axis view, as seen from point of view of user

Fig. 2. The concept of off-axis view

because the rendered frames are observed monocularly from an incorrect viewpoint in case they should be seen in a straight pose towards the screen.

In order to implement that aspect, we use head-coupled perspective [2] which means that we directly control the camera position by attaching it relatively to the human face position. Furthermore, we also want to achieve the most correct perspective of a 3D stereoscopic rendered scene, we employ an off-axis view which is illustrated by an asymmetric frustum. By using this frustum in the system, the distortion of object arising when the viewer looks at the screen from different angles will be reduced at most. The Fig. 2 illustrates the distortion appeared with different viewing angles [10].

3.2 Face Tracking

When applying a head-coupled perspective into the real-time rendering system, there is always a tracking technique involved because the system must know where the user's eyes are located. The user eyes will be considered as two virtual cameras and their positions will be assigned to the virtual camera's positions respectively. Those techniques are used with many researches on view-dependent rendering because they are straightforward and easy to implement.

In our approach, we employ the stereo display methods which rely on the glasses. The viewers must wear glasses so that they can perceive the 3D illusion of the virtual scene. Wearable glasses will affect the accuracy of eyes detection and tracking process because the features used to detect eyes are no longer true and distinctive enough. Thus, the result is not trust-able for tracking eyes positions and assign to the virtual cameras. Addressing this problem, we choose to track a human face position using digital camera. The eyes position can be computed correctly because it is true that the eye's position is always remained relative to the face position. In addition to that issue, the algorithm used in detection the human face is robust and adaptable with many changes. For instance, the viewer can wear a anaglyph or polarized glass without any negative effects on the first face detection step. Based on that observation, we use a robust face detection framework implemented by OpenCV called Viola-Johns[5]. This framework describes a approach using machine learning to detect visual objects. It combines four concepts which are Haar features, an integral image for rapid feature detection, a machine-learning method and a cascaded classifiers[5].

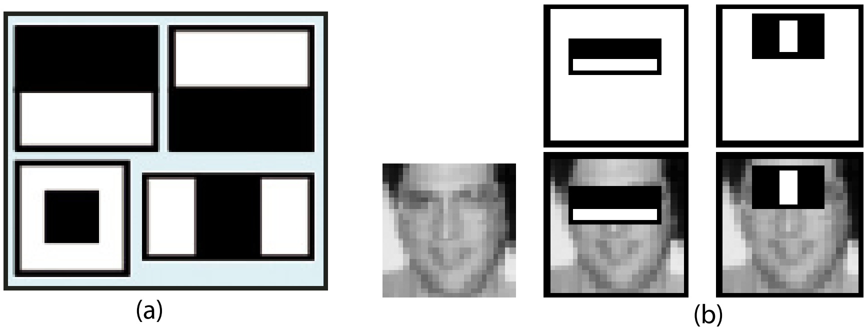


Fig. 3. The Fig. (a) shows the examples of Haar features in OpenCV. The Fig. (b) shows the first two Haar features in the Viola-Jones cascade.

In their detection framework [5], the detection features, which are the combination of sequence rectangles, are based on Haar wavelets. The sequent rectangles form the better method to visual detection tasks. Because of the difference from original Haar wavelet, the features they use are called Haar-like features. The Fig. 3 shows some examples of the Haar features used in OpenCV and the first two Haar features which are used in the original Viola-Jones cascades. To check whether the Haar-like features are being presented in the image, they simply use subtraction between the average dark-region and light-region pixel value. The differences will be compared with a threshold achieved during a machine learning process to decide the presence of those features in image. They use the technique called Integral image. The integral value is calculated for each pixel. This value is recursively computed by adding the sum of all the pixels from its top left pixels. The process starts from the top left pixel of the image and traverse to the right down [11]. Thus, the features are detected efficiently in every location with some scales.

They use a machine-learning method called AdaBoost to decide the Haar-like features and specify the threshold value. AdaBoost method selects a small number of critical visual features from a larger set and makes an extremely efficient classifiers. AdaBoost combines many classifiers which are assigned the weight to create one effective classifier. They use a training face set as the input for AdaBoost learning method, the threshold at each filter level is set to a value at most that face examples in the training set can pass. This method is illustrated

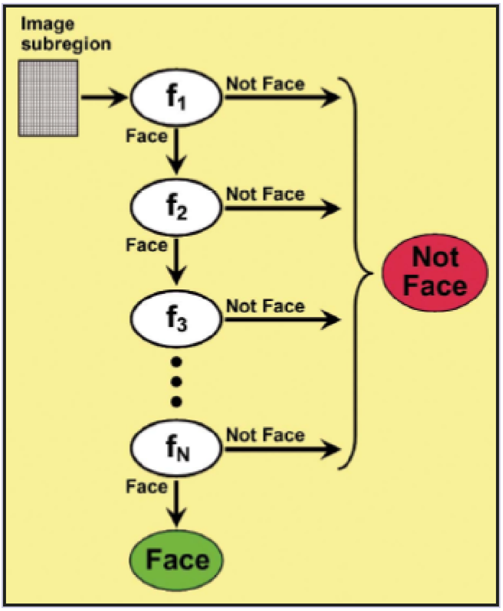


Fig. 4. The classifier cascade

in the Fig. 4 [11]. The image sub-region will pass through the filters, if that region can pass all the filters in the chain, it will be classified as a human face.

After the first stage of face detection, we use another technique to track the face. There are many tracking techniques based on the movement or the colour histograms from the object we want to track. Using the techniques based on the colour of object is fast, effective and rather widely implemented. However, in our system when the viewer wears a glass, especially the red-cyan glass, the incorrect tracking may occur. Tracking by using histogram can potentially miss the face in the frame captured by camera due to the presence of some regions having the same colours with the face such as the neck or some regions in environment around the viewer. The inaccuracy as well as the failure of face tracking breaks our system stability. Therefore, we involve a tracking technique based on the movement of object, that technique is called Lucas-Kanade (LK) optical flow [9].

The LK optical flow technique is typically used to track the optical flow of a video. In our research, this technique will track the location of some specific points in the face detected across multiple frames which the digital camera is capturing. The basic idea of the LK algorithm rests on three assumptions. A pixel from the image of an object in the scene will retain its appearance when moving from frame to frame; this is known as brightness constancy. The motion of a surface region changes slowly in time relatively to the scale of motion in the image. The point belonging to the same surface will have the similar motion with the neighbouring points in a scene.

The LK optical flow technique will find the points of interest which are located on the face of the viewer. Those points are in turn used to track the motion the face so that we can decide the position to assign it to the virtual camera used in stereoscopic rendering. In case that the interest points do not track the face feature precisely, the stage of face detection will be invoked to relocate the face and compute interest points.

4 Implementation

4.1 Stereoscopy with Asymmetric Frustums

We implement a stereoscopic real-time rendering system using OpenGL. It is developed based upon fundamental and solid background theory on stereoscopy made by many researchers in computer graphics field. We utilize a stereoscopic rendering application [12] from the course "Build your own 3D display" in SIGGRAPH 2011 to apply it into our research and finalize with the best performance based on GPU rendering. The application gives the basic illustration of rendering a scene off axis in stereoscopy.

In the system, we render the scene by locating two virtual cameras relatively to the eye positions and the distance between two virtual cameras are computed by eyes separation value. The pseudo code in below illustrates the steps in order to create asymmetric frustum to each virtual camera in OpenGL. Two virtual cameras positions will be computed by shifting main camera a half eyes separation distance to the left and right alternatively. The width and height of the window viewport are also computed in centimetre. The *screen.pitch* will return the size of a pixel in centimetre. We continue to compute other values to form the parameters to construct the asymmetric frustum.

OpenGL initializing asymmetric frustum pseudo code

```
compute x value of left/right camera position alternatively;
depthRatio = camera.near/camera.z;
halfWidth = window_width * screen.pitch/2;
halfHeight = window_height * screen.pitch/2;
left = -depthRatio * (x + halfWidth);
right = -depthRatio * (x - halfWidth);
bottom = -depthRatio * (camera.y + halfHeight);
top = -depthRatio * (camera.y - halfHeight);
set glFrustum() based on six standard frustum values;
```

In anaglyph stereo rendering, we use a fragment shader to specify three colour modes of anaglyph which are full-colour, half-colour and optimized anaglyph. The fragment shader will compute the output colour according to the user-selected mode. We compute the anaglyph's RGB values (r_a, g_a, b_a) from the RGB values of the left (r_1, g_1, b_1) and right images (r_2, g_2, b_2)[13]. We also use a fragment shader for row-interlaced rendering mode. For each view of camera, the shader evaluates the texture coordinate and extract the view mask value. The fragment colour will then be assigned for the interlaced image.

4.2 Face Tracking with OpenCV

In our system, we set up a digital webcam by mounting it into the top of the screen where it lies exactly in the middle point of the top screen edge. The viewer position is restricted to the distance from 40 centimetres to 100 centimetres in order that the camera can capture a face with the best quality for detection task.

The frame as individual image extracted from the webcam will be used as the input for the face detection. The viewer face will be detected as soon as the frames are outputted from the camera. However, there is a restriction in enabling the tracking step. We delay tracking until the viewer's face moves into the centre position of the sequence frames taken by camera in both vertical and horizontal direction. Otherwise, we must adjust the camera vertically so that the face appears in centre. Using such restriction, the relationship between face and virtual camera's position can be proportional to each other because we assume that the starting position of face is to give a look towards the screen in the straight orientation. In addition to this issue, we are able to calculate the centre point of two eyes because it also has the relative position to the face. Finally, we can retrieve the spatial changes from centre of eyes and assign it to the virtual cameras.

We use the implementation in OpenCV for those theories which have been discussed in section 3 - Technique background. OpenCV already implemented the Haar Cascade classifier in order to use not only for face detection but also for object detection generally. The face detector will examine each image location and classify whether that location contains a face. The classification uses a scale in size 50 x 50 pixels to scan faces and it runs through the image several times to search for faces within a range of scales. OpenCV provides an XML file name `haarcascade_frontalface_default.xml` which is the output after the learning process on faces data. The classifier uses that file for decision on which location is a face in the image. The face detector task is under form of an OpenCV method called `cvHaarDetectObjects`. This method will use the classifier as a parameter. The classifier is represented by `CvHaarClassifierCascade` class in OpenCV.

As discussed in the previous section, we use the algorithm called Lucas-Kanade optical flow to track the face which is already detected in the detection step. The first thing to do with this algorithm is getting the feature we want to keep track in frame by frame. In our case, that is the face. We invoke the OpenCV method `cvGoodFeaturesToTrack` to get the interest points in the face region we have from the detecting phase. We then use those points as the good features for tracking. After that, we continue to invoke the method `cvCalcOpticalFlowPyrLK`. This optical flow function makes use of the good tracking features and indicates whether the tracking is proceeding well [9]. By using the combination of these two functions, we can obtain the effective tracking. Our system keeps running in real-time for both face tracking and stereoscopic rendering.

5 Discussion and Result

Our paper presents a technique for real-time rendering a stereoscopic scene which involves the view-dependent aspect. The view-dependent rendering of a normal 3D scene was studied and implemented by many researchers. However, the studies on that of stereoscopy are still interesting research field. Some implementations have been introduced while the challenges on building the real-time system still remain attractive because the computational cost for both stereoscopy and face tracking is extremely high.

We also present an off axis frustum or asymmetric frustum by which the perspective will be reflected as much realistic as the scene should appear in a real environment. These techniques would be effective in case of implementing a virtual reality system using the popular and simple devices. The user will need only display devices for 3D stereoscopy and they will be able to look at the real scene by only sitting front of the working place.

We develop and run our system on Windows 7 Ultimate version with service pack 1. Our test platform is a CPU 3.7GHz Intel Core i7, NVIDIA GTX 480 graphics card as well as 16GB of RAM. For the stereo display, we use a 27-inch 3D screen with the polarized glass. Our system runs fast and smoothly with an average of 30 FPS. We perform many changes in viewer position. The 3D object appears stable to the viewer perception as a real object is floating out of the screen or located behind the screen.

In the future, we will improve the performance of the system. With the 3D stereoscopic scene changed by human view, the system would interest the learner in case it is applied for education. We also would like to involve the hand gesture tracking system in order to enable the interaction between the observer and the virtual 3D objects rendered by our system.

6 Conclusion

In our research, we present and implement a 3D stereoscopic rendering system in real-time based on the viewer's face position. We make the combination of 3D stereoscopic rendering and face tracking techniques. By improving them, we achieve a system having a good performance. By using our system, the user can perceive the realistic scene in 3D and the observed scene therefore can be considered as the scene in reality. The combination of those techniques helps us implement a widely applicable system.

Acknowledgments. This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology (No. 2012M3C4A7032182).

References

1. Slotsbo, P.: 3D Interactive and View Dependent Stereo Rendering. M.Sc. thesis at Technical University of Denmark (2004)
2. Kevin, A.: 3D task performance using head-coupled stereo displays. M.Sc thesis at The university of Bristish Columbia, pp. 14–24 (1993)
3. Samuel, G.: 3D Vision Technology Develop, Design, Play in 3D Stereo. NVIDA (2011)
4. Sbastien, P., Vincent, P., Antoine, L., Marc, D.: An Interactive and Immersive System that dynamically adapts 2D projections to the location of a users eyes. In: Proceedings of International Conference on 3D Imaging (2012)
5. Paul, V., Michael, J.: Rapid Object Detection using a Boosted Cascade of Simple Features. In: Proceedings of Conference on Computer Vision and Recognition (2001)
6. Paul, B., Peter, M.: Stereoscopy: Theory and Practice. In: Workshop at VSMM. Queensland University of Technology (2007)
7. Jens, G., Gabriele, P.: View-dependent 3D Projection using Depth-Image-based Head Tracking. In: 8th IEEE International Workshop on Projector Camera Systems (2011)
8. Buchanan, P., Green, R.: Creating a view dependent rendering system for mainstream use. In: 23rd International Conference on Image and Vision Computing New Zealand, IVCNZ 2008, pp. 1–6 (2008)
9. Gary, B., Adrian, K.: Learning OpenCV. O'Reilly Media publisher (2009)
10. Opera development article, `http://dev.opera.com/articles/view/head-tracking-with-webrtc/` (visited on February 20, 2013)
11. Cognotics article, `http://www.cognotics.com/opencv/servo_2007_series/index.html` (visited on February 22, 2013)
12. Build your own 3D displays course, `http://web.media.mit.edu/~mhirsch/byo3d/` (visited on January 20, 2013)
13. Anaglyph Methods Comparison, `http://3dtv.at/Knowhow/AnaglyphComparison_en.aspx` (visited on January 15, 2013)