CSCI 3290 Computational Photography

Final Project – Cardboard

Due Date: 11:59pm on Monday, Dec 28th, 2015

Demonstration Dates: Dec. 29th and 30th*

*** You can submit your work and demo it earlier if you will not be available these days.**

I. Background

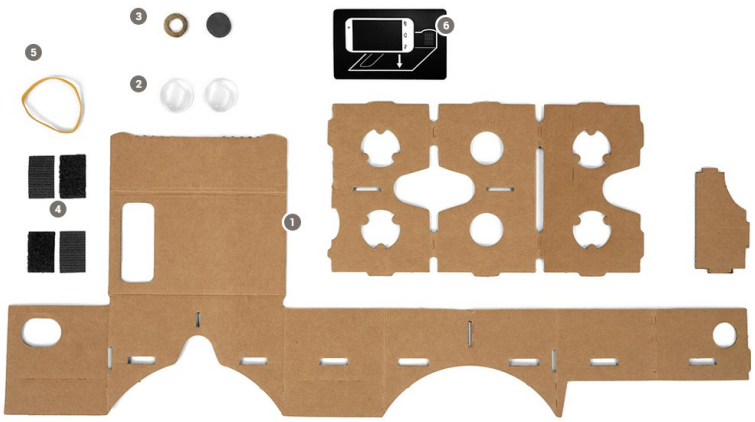
Google has designed an interesting equipment named '**Cardboard**'. It facilitates everyone to experience virtual reality in a simple, fun, and inexpensive way. It is essentially a 3D display with the help of your cell phone.

In this final project, you will fabricate your own Cardboard with given materials for displaying 3D contents on your cell phone. You are also encouraged to express your creativity to establish 3D scene viewers or cool animation effect. The only limitation is your imagination.

This final project will be done in **groups**; each group consists of at most **2** members with at least one usable Android phone (see the requirement in https://cardboard.withgoogle.com/).

II. Build Cardboard Glasses

Step 1: Collecting Materials [1] (Our provided material based on blueprint version 1)



1. Cardboard

   Corrugated cardboard sheet, preferably E Flute (corrugated cardboard comes in a variety of thicknesses called "flutes"). For best results, you should look for strong, thin cardboard (sturdy shoe box rather than moving box). Minimum size: 8.75in (22cm) by 22in (56cm), and 0.06in (1.5mm) thickness. You can get your cardboard from **PARK-n-SHOP** in CUHK campus, collecting printing-paper boxes, asking your family for help, or by ordering extra-large pizza.

   Hint: You can also get finished cardboard from the tutors. But this will result in a lower score in this project (see further below).

2. Lenses

   Lenses that have a **45mm** focal distance might work. Biconvex lenses work best because they prevent distortion around the edges.

   - We provide **a pair of lens** in the material package.

3. Magnets

   One neodymium ring magnet and one ceramic disk magnet. Approximate size: 0.75in (19mm) diameter and 0.12in (3mm) thickness.

   - We provide **a pair of magnets** in the material package.

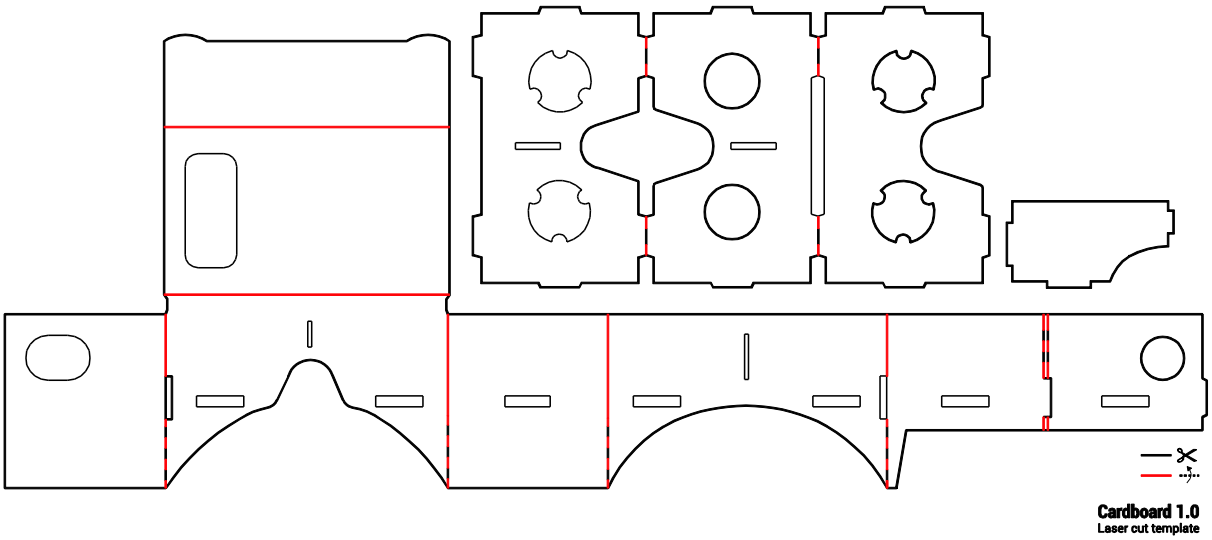
4. Velcro

   Two strips of regular strength adhesive-backed velcro. Approximate size: 0.75in (20mm) by 1.25in (30mm).

   - We provide **2 pairs of Velcro** in the material package.

5. Rubber band

   One rubber band, to prevent the phone from sliding out. Minimum length of 3.2in (8cm). This is not necessary. You can prepare it for your own device.



Step 2: Following Blueprint

The full-size blueprints are in ***cardboard_design_v1.0.zip***

Print it out on papers and cut out the shape on cardboards.

Step 3: Testing Your Product

Goto the website http://g.co/chromevr using your device.

Or download the Android apps: **Cardboard**
(https://play.google.com/store/apps/details?id=com.google.samples.apps.cardboarddemo)

Apple apps:

(https://itunes.apple.com/us/app/google-cardboard/id987962261?mt=8)

See if your Cardboard hardware works.

III. Display Stereoscope Images on Android Phones By Your Cardboard Glasses

We have prepared a simple program *CardboardDemo*, which can display two images or two videos on the left and right halves of the screen in your Android cell phone. You should build your own project based on this demo program.

Step 1: Install Android Studio

For your own PC:

Download and install JDK first (according to your OS):

http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

Download and install Android Studio:

https://developer.android.com/sdk/installing/studio.html

For Room 123 PCs:

Press **Win+R**, type in “cmd”, press **Enter**.

Type in:

> S:

> cd S:\android-studio\bin

> set JAVA_HOME=S:\jdk1.6.0_27

> studio.exe

Now you can open Android Studio.

Step 2: Install Cell Phone Drivers

For your own PC:

Plug-in your cell phone to your device. Your system will automatically install drivers.

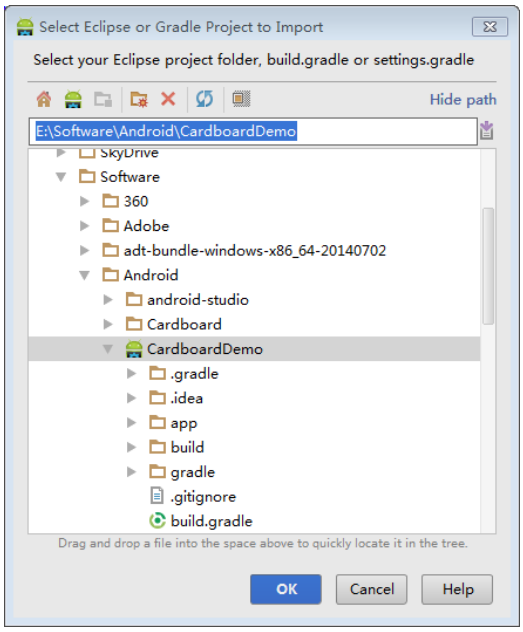
For Room 123 PC:

The lab may not allow you to install drivers. Thus you can only test your code on simulator.

Step 3: Import CardboardDemo Project

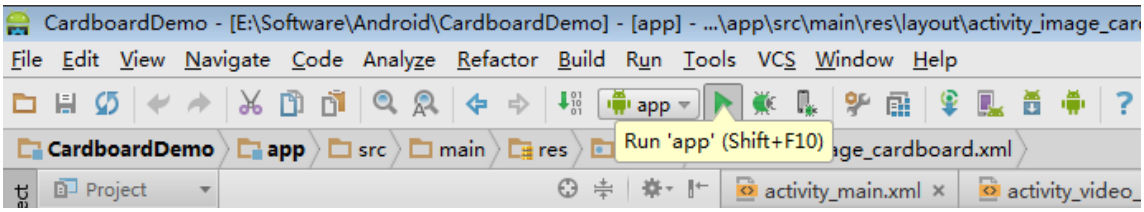
Extract *CardboardDemo* from package *cardboarddemo*.zip.

In Android Studio, press menu “File”-> “Import Project”, select *CardboardDemo* folder and press OK.



Step 4: Build and Run

In Android Studio, press menu “Build” -> “Make Project”



Press “Run” button to run this code.

Hint: You may come across errors for lacking certain packages. A simple way is to install all packages that are missing.

You may encounter lack of SDK errors for Android Studio in Room 123. You need to download Android Studio, and extract it to a folder (for example, **C:\android-studio**). Then follow Step 1 (except the last step) to run **C:\android-studio\bin\studio.exe** of your downloaded version instead of the version in “**S:\android-studio**”

Step 5: Run on Your Phone

Place 2 images in the following position. (/sdcard/ is the root of your SD card of your phone)

/sdcard/ CardBoardDemo/left.jpg

/sdcard/ CardBoardDemo/right.jpg

Then test the app on your phone, you will see:

Press “Start Image Cardboard”. The two images will be displayed on the left and right halves of your screen.

It is similar for displaying videos, where the two videos are as follows.

/sdcard/ CardBoardDemo/left.mp4

/sdcard/ CardBoardDemo/right.mp4

You can change the **image path** in the file:

..\CardboardDemo\app\src\main\java\com\demo\zhouc\cardboarddemo**ImageCardboard**.java

You can change the **video path** in the file:

..\CardboardDemo\app\src\main\java\com\demo\zhouc\cardboarddemo**VideoCardboard**.java

You can change the **panorama image path** in the file:

..\CardboardDemo\app\src\main\java\com\demo\zhouc\cardboarddemo**PanoramaCardboard**.java

Hint: you need to tune the scale and translation of the images and videos to see 3D effect. Change the parameters in **ImageCardboard**.java & **VideoCardboard**.java.

The root of SD card may vary from device to device. If you cannot find ‘/sdcard’, try ‘/storage/sdcard1’, /storage/emulated/0’ or others.

IV. Generate Stereoscope Images using a Single Image & Depth Map (MATLAB)

Stereo image pairs can be generated by giving a single image and a depth map. The algorithm is named as *Depth Image-Based Rendering* (DIBR). Given a 2D image and a depth map, the algorithm will generate a virtual view - what the scene would look like if a camera was placed in a different position. The phenomenon behind the algorithm is parallax: viewpoint difference between two eyes. Parallax makes you feel

- objects close to the viewer move faster,
- objects far away move slowly,
- objects at "infinity" don't move at all.

Given per-pixel dense depth, generating a virtual view is easy.

- For each pixel in the depth map, compute the corresponding disparity.
- For each pixel in the source 2D image, find a new location for it in the new view as "old location + disparity" of that specific pixel.

The relation between **depth** and **disparity** can be determined by triangulation rules. Camera movement b (eye disparity) is assumed to be **strictly horizontal** (along x-direction). Thus the **y-coordinates** are fixed after rendering. The z-coordinate of a point is known from the depth map.

f is the **focal length**. You can set f to 1, but feel free to experiment and see how the value affects the outcome. The value we are interested in is the difference between a point's locations in two different views, i.e. $d = x_L^* - x_R^*$. The map containing d values is called **disparity map**.

First see how you can convert **image space coordinates** x_L^* and x_R^* into **world coordinates** x_L and x_R based on the following Figure. Then identify two triangles that are similar, and use the fact that the ratio between their bases and their heights is the same. Finally, you should be able to solve for *d*, which is disparity related to your depth z.

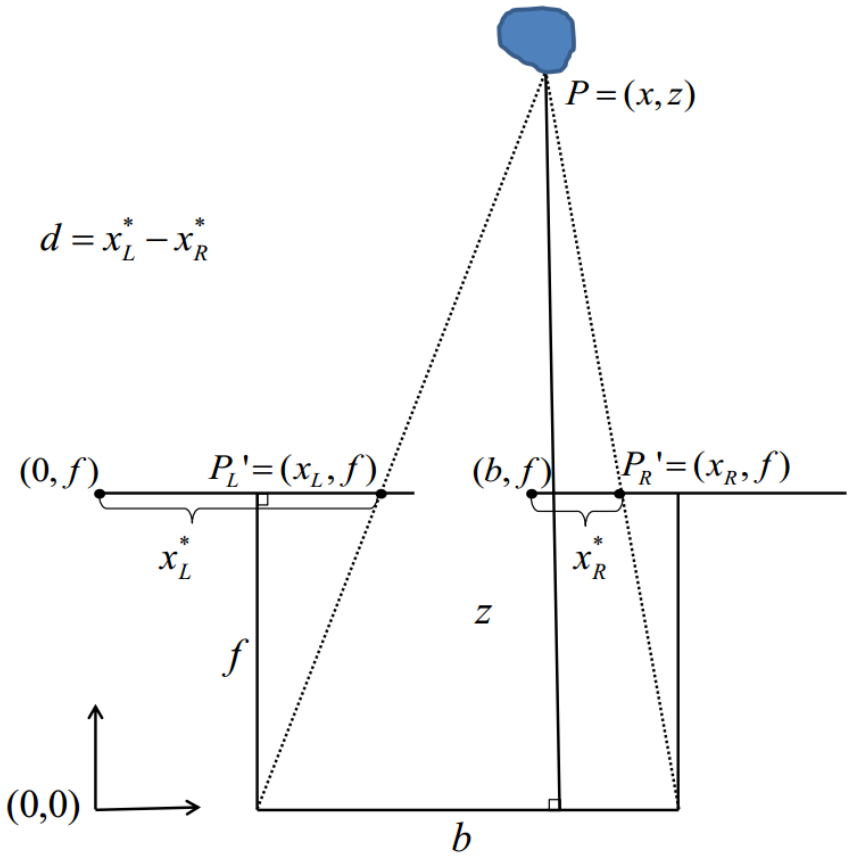


Figure. A dual camera setup for determining the relation between depth and disparity on a (x,z) coordinate system

It is unavoidable that some areas will be revealed while attempting to "peek" behind a close object. This will lead to unknown color. Either keep it black or fill it using neighborhood color by interpolation in MATLAB.

In our project, we provide you **one image** with corresponded **disparity map** for creating another view. You are supposedly compute this new view on PC using MATLAB. Then display the two views using your 3D cardboard display. You can search more examples online or generating one by Photoshop.

Step 1: Convert Depth Map into Depth Values

(If you already have the **disparity map**, jump to Step 3) Given the depth map D, the real depth value of each point from your eyes can be determined as

$$T_{i,j} = \alpha \cdot D_{i,j} + \beta$$

where α, β are two parameters you can adjust when creating two views. (i,j) are the coordinates.

Step 2: Convert Depth Values into Disparity Map

Estimate the values of focal length f, distance between eyes b, distance of each point in the image is $T_{i,j}$. Using the triangulation rules, calculate the disparity map d. $d_{i,j}$ means the pixel at (i,j) position need to be shifted $d_{i,j}$ pixels horizontally.

We need images for left and right eyes, respectively. Thus we have 2 disparity maps d^L, d^R for left eye and right eye respectively. If you already have one **disparity map,** you can regard the image as left eye image, and warp the image using disparity map to right eye.

Hint: the given disparity map may not suit your Cardboard, you will need to scale up/down the disparity values to get better results.

Step 3: Warp images according to Disparity Map

Now we have an image I and a disparity map d. We need to do the warping for the resulting image J. For each pixel in I, we know $I_{i.j}$ need to be shifted to position $(i, j + d_{i,j})$, that is

$$J_{i,j+d_{i,j}} = I_{i,j}$$

J's y-position $j + d_{i,j}$ may not be an integer. Referring to the lecture notes, we need to use back-warping. *interp2()* in MATLAB can interpolate unknown values.

Hint: *interp2()* provides several interpolation algorithms, try them.

Step 4: Display Warped Images on your Device

Copy your image to the correct path of your device, and view your stereo images through your cardboard glasses. If you cannot produce correct effect, tune any available parameters and try to figure out why.

V. Video Handling - Add a 3D logo to a video

In this part, you need to add a floating logo on a video (as the NAT GEO logo below). When you see through the cardboard glasses, the logo appears in front of your video.

Use *VideoReader(), VideoWriter(), writeVideo()* to read / write video in MATLAB.

Step 1: Assign a Depth for the Logo

You can choose any logo. For example

Assign this logo a reasonable depth by yourself.

Step 2: Calculate the Disparity Map

This part is similar to previous task.

Step 3: Get Warped Logo Map for Left / Right Eye

Generate 2 images L, R for left / right eyes respectively; they are of the same size as video frames. This is similar to that of last section. Logos are in the correct position of L, R according to disparity map.

Generate 2 alpha-channel images A_L, A_R for L, R. Make $\mathrm{A}_{\mathrm{i,j}}^{\mathrm{L}} > 0$, if $L_{i,j}$ belongs to the logo; $\mathrm{A}_{\mathrm{i,j}}^{\mathrm{L}} = 0$, if $L_{i,j}$ belongs to the background or transparent part.

Step 4: Using alpha-blending for each view

For left-eye view, we now have three images that are of the same size: I, L, A^L, where I is one frame from the video. The resulting image is:

$$J^L = A^L \cdot L + (1 - A^L) \cdot I$$

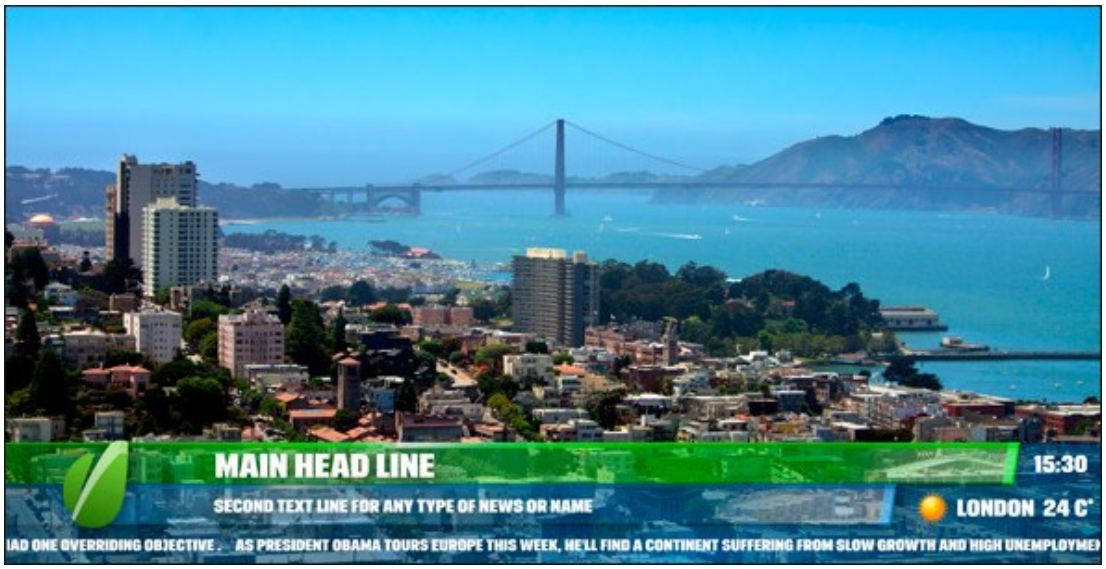
Do the same for right-eye view.

Hint: $A_{i,j}$ can also be floating values between 0 and 1. Tune the values to get the best visual effect.

Step 5: Display your video on your device

Copy the videos into the correct path of your device. View your video with stereo logo through your cardboard glasses. If you cannot see correct effect as expected, tune the parameters of your method and try to figure out why.

VI. Video Handling - Add Synchronized Stereo Subtitles



This is very much similar to adding a stereo logo. Above is an example.

The only difference is that you need to read the subtitles and add correct subtitles to the correct frames.

Hint: The color of the video contains variations. Therefore, the color of your subtitles may appear indistinguishable from the background video. A common solution to handle this is to add a transparent bar as the background of the subtitles, shown in green in the above example. The color of the bar needs to be quite different from color of the subtitles, and does not attract too much attention. Finally use alpha-blending to achieve desired effect.

VII. Panorama image viewing

Cardboard will be a perfect tool to view panorama images, as shown above. In this section, we use cardboard to enjoy panorama images in an immerse way.

We use sensors such as accelerometer and magnetic to determine the orientation of your device, then show the image according to the device orientation. Therefore, when you move your head, you can see scenes 'moving' just like you are there!

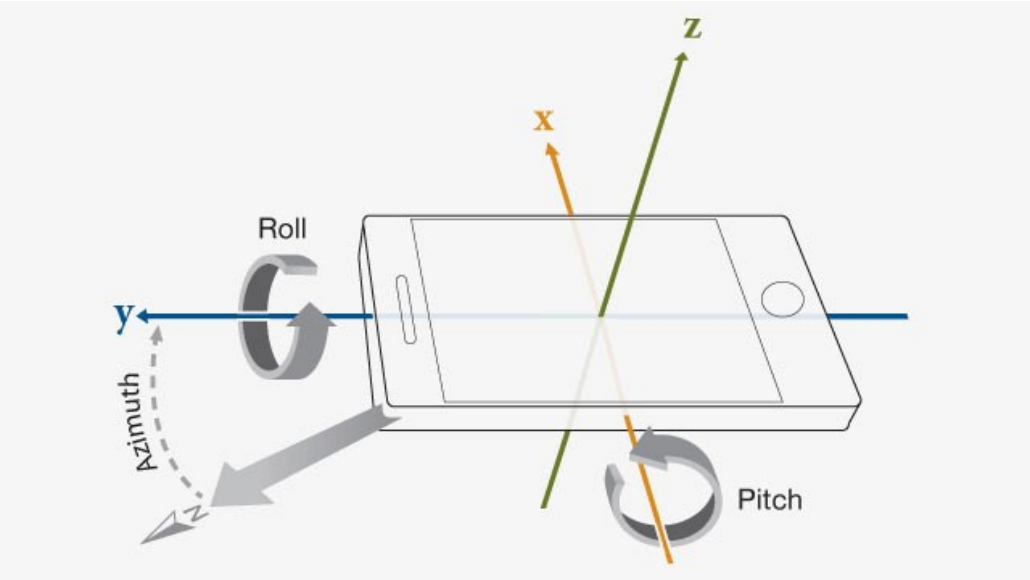
Step 1: Prepare panorama image

In this project, you can use existing images on the Internet.

It's better that you use your own taken photos. (man y photo have Panorama mode, and Photoshop can generate good panorama too)

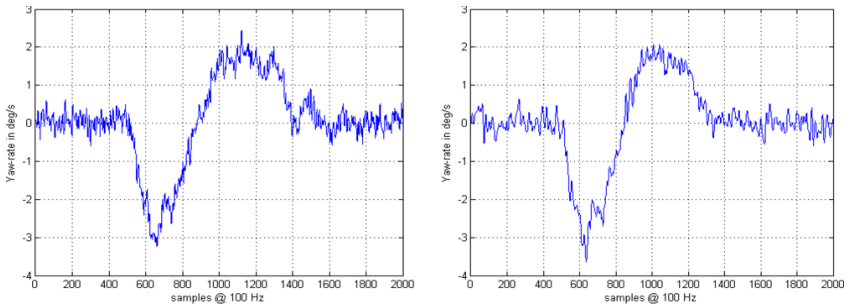
Hint: In this part, generate a pair of stereo panorama is a difficult task. Therefore, showing the same image for two eyes is OK.

Step 2: Get Android sensor data to determine device orientation



We use Android API getOrientation() to get device orientation, which contains three angles: roll, pitch and azimuth. As shown in the figure above.

getOrientation() is based the sensor data of Accelerometer and Magnetic.

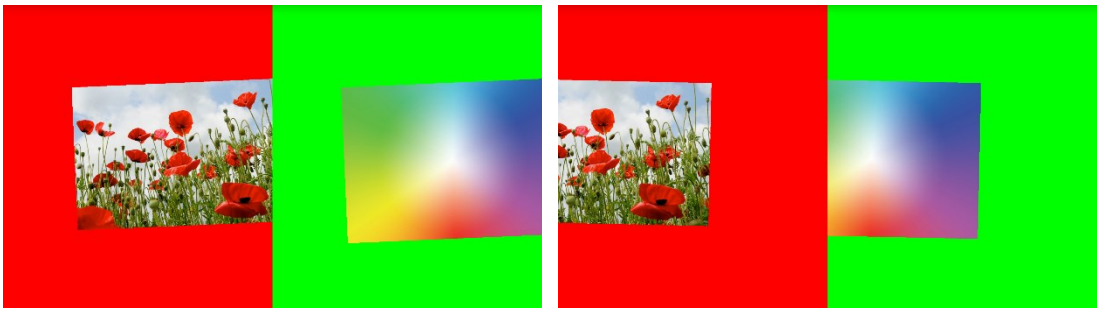


However, raw orientation data is very noisy, a typical plot of this noisy data is as above. If we directly use raw orientation data to control our view, the image will keep vibrating randomly.

A reasonable thought is to first smooth the raw orientation data before applying to the viewport. The simplest way is to apply a mean filter: we keep record n raw samples, and use their average value as a reliable estimate. More advanced and effective filters such as Kalman filters are better, as bonus parts.

Hint: We have finished most of the part regarding sensor interfaces in the given skeleton code. Please refer to the code for details.

Step 3: Relate device orientations with image translations



Look left Look right

When you look left, images should be moved right. When you look right, images should be moved left. As shown in the figure above. You need to determine the relationship between device orientation and image translation. (We also considering in-plane rotation of the images)

Hint: Images seem awkward in above figure, so please use large panorama images to cover the whole region, and carefully set scales and translations in order to view in a natural and comfortable way.

Step 4: Zoom-in / zoom-out with magnets button

Triger magnets to zoom-in the panorama image for a better view.

Triger again to go back to normal size.

Hint: you can refer to Cardboard official sample (cardboardsample.zip) to see how to use magnets.

VIII. Packages we provide

Software:

1. Specification of Final Project
2. Blueprints: *cardboard_design_v1.0*.zip
3. CardboardDemo Project: *cardboarddemo_v2*.zip (for basic part)
4. CardboardSample Project: *cardboardsample*.zip (cardboard official demo)
5. A few related images / logos / videos etc.
6. Links for related resource

Hardware:

All necessary accessories mentioned above.

IX. Scoring

a) Basic Part

1. Fabricate Google cardboard glasses (at most 20%)

   20 %: If you build your cardboard glasses framework by yourself. The glasses need to work smoothly for final demonstration. (See previous Section II)

   OR

   10 %: If you use the full package with cardboard from the tutors or purchased by yourself somewhere.

2. Binocular Stereoscope Images (10%)

   10 %: Build and run the sample project CardboardDemo. Successfully display given stereoscope images. See the correct 3D effect. (Section III)

3. Generate Stereoscope Images using a Single Image & Depth Map (by MATLAB on PC) (10%)

   10 %: Display you own generated stereo images using a depth map or disparity map. See the correct 3D effect. (Section IV)

4. Add a Stereo Logo to a video / Add Synchronized Stereo Subtitles (10%)

   5 %: Add stereoscope logos floating in front of the video. (See previous Section V)

   5 %: Add synchronized stereoscope subtitles to the movie. (See previous Section VI)

5. Panorama Image Viewing (15%)

5 %: Successfully display given panorama images. See the correct 3D effect. (Section VII)

5 %: Use smooth filter to suppress vibration (Section VII)

5 %: Use magnets to zoom-in / zoom-out panorama images (Section VII)

6. Demo and Presentation (10%)

   10 %: Present what you did in the project. Demonstrate your work using the hardware. Less than 10 PPT slides are recommended.

b) Bells and Whistles

1. Up to 6 %: Find at least 3 more stereo image pairs to show 3D. Find at least 3 more videos to add logos and subtitles. (images and videos should **not** be from Google examples)
2. Up to 25 %: Design a friendly user-interface for your Android app.

   You can add any interesting functionality into your user interface. Making it good looking and convenient to use is your target.

   Requirements: allow users to select input images & videos to play, like the YouTube in the CardBoard App downloaded from Google Play Store.

   High scores will be given if you create a UI similar to g.co/chromevr or YouTube in the CardBoard App that can smoothly browse the menu/videos in 360 degrees by rotating head.
3. Up to 10 %: Use the magnet to control your app, e.g. to switch over images, pause & resume movies, etc. (other than in Section VII)
4. Up to 15 %: Change the logos to animation in front of the video.

   In the basic part, we process the videos with logos. In this extra part, you need to change logos to interesting animations in front of the original videos – simply put, the logo animates.
5. Up to 15 %: Zoom-in or zoom-out of the image pairs

   Zoom-in / zoom-out are **no easy** problems. Simply scale-up / scale-down the images will appear unnatural when seeing through cardboard glasses. You need to take **depth** image and **triangulation** rule into consideration.

All results need to be viewed effortlessly by anyone in Cardboard as 3D. **The total score of this project for each student does not exceed 100.**

Other Remarks

- 10% total score increase for students working alone.
- At most **one-day** late is allowed because the demonstration is arranged after submission deadline.

- 20% off for any late submission within one day.
- 100% off for any late submission beyond one day.
- The final project is to be completed in **GROUPs.**
- Absolutely **NO sharing or copying** code! Offender will be given a failure grade and the case will be reported to the faculty.

X. Details of the Submission

Your submission should contain the following things.

a) Compliable codes for the project

b) The Android app demo program run on your smart phone.

Notes: You can use all functions and API provided by Android SDK & cardboard glasses API. You can also use 3rd party libraries. However, you need to list all what you have used and a brief introduction in the README.docx. Otherwise, it will be counted as plagiarism.

You own work **MUST** be the majority in your uploaded codes.

c) About report

NO html report is needed for the final project. But you can choose to prepare at most 10 slides in PowerPoint, detailed below.

d) Demonstration and presentation

It is arranged on **Dec. 29th and 30th in Rm121 and Rm123**. If you need an early demonstration, send your request to tutors as soon as possible.

In the demonstration, you need to introduce what you have done for EVERY basic and bonus point. Then tutors and lecturer will see your corresponding codes and watch results in your provided cardboard. They will grade accordingly.

So a few slides to guide your demonstration will be great. Don't make them more than 10 slides.

e) Handing in

The folder you hand in must contain the following:

- **README.docx** - containing anything about the project that you want to tell the tutors. Give a brief **summarization** of what you've done. **Highlight** every extra bonus you've finished.
- **code/** - directory containing all your code for this assignment

Compressed the folder into <*your student ID*>**-Final.zip** or <*your student ID*>**-Final.rar**, and upload it to e-Learning system.

XI. Reference

[1] https://cardboard.withgoogle.com/

[2] https://developers.google.com/cardboard/get-started

[3] https://developers.google.com/cardboard/api/reference/packages