Van Mai Nguyen Thi
SProj weekly report
Week 11: Nov 15 – Nov 21, 2014

Multiplanar paper: Ashdown M., *A Flexible Projector-Camera System for Multi-Planar Displays*

H_{ij} = homography from i to j, for $i, j \in \{s\ (screen), p\ (projector), c\ (camera)\}$

In the last meeting we figured out that we need all three homographies: H_{CS}, H_{PS} and H_{PC}, and not just the projector-camera homography like we had thought before. While the H_{PC} is sufficient to generate an anamorphosis for the camera, we will need the other homographies if we consider actual viewers at some known positions instead of cameras.

- - - - - - - - - - - -
<u>PlanarAnamorph program:</u>

Last week, we found a problem with projected images being scaled up/down, while the homographies were computed for the original un-scaled images. The reason why I scaled the images was so that they fit inside the projector screen (projector desktop), and especially so that the image used for calibrating is projected without other desktop elements that would make it hard to detect correct features of the calibrating image.
In order to solve this, I tried to implement full screen view within my program. However I was not able to find an implementation of full screen view for openCV in Python that would work on Mac. I might try later with some Python libraries other than the openCV. For now, I am using the Present mode in Processing, which allows me to display and save an image that is exactly the size of the projector screen. In the same program I have an option of displaying an image. I use this feature to display the anamorphic image after it has been created in my main python program.

Another problem that I did not foresee was again related to image sizes. This time, it is about the mismatch between the sizes of the first camera image $cam1$ (used for calibration) and the second camera image $cam2$ (this one is warped to create the anamorphic image). After implementing the full screen view in Processing, I was able to generate $cam1$ in the size of the projector screen. For $cam2$, I purposefully chose a smaller size so that the resulting anamorphic image fits inside the projector screen. However, after projecting the anamorphic image, the camera saw it as a parallelogram rather than a rectangle. That is because: diagrams on p. 11 in my notes.
The solution to this is using the same size for the both camera images, and placing the smaller image that we want to warp in the center of that image.
It is also convenient to keep the dimensions of both projector images the same (in fact, all my camera and projector images are the same size) because then we know that the anamorphic image will fit perfectly in the projector screen.

Van Mai Nguyen Thi
SProj weekly report
Week 11: Nov 15 – Nov 21, 2014

The current algorithm can still be improved by

- adding an option for automatic placement and resizing of the image that we want to warp inside the $cam2$, and
- writing a parent program that controls and runs all parts of the program. Currently I have to call different programs individually, which is not practical.

I will also state the convention we will use to identify the various images in out system:
$proj1 =$ the first image that I project in order to calibrate the camera
$cam1 =$ capture of projected $proj1$
$cam2 =$ image that I want to see in the camera; this is warped to create anamorph
$proj2 =$ anamorphic projection of $cam2$