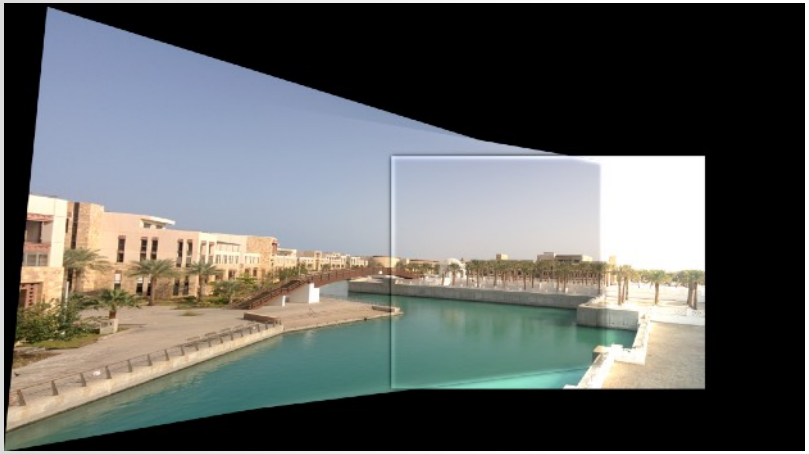
Computational Photography
Assignment #5
Panoramas

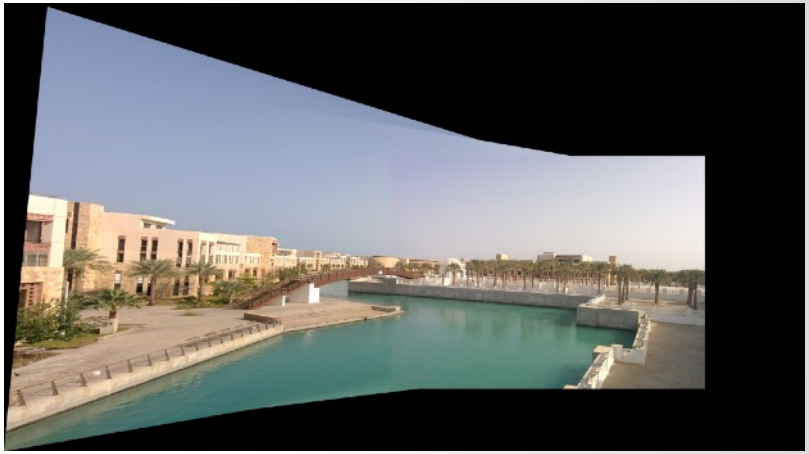
Nitish Sanghi
Spring 2019

Results - Test Images Main Blending Function

Results - Test Images Other Blending Function



Pyramid Blending as implemented in Assignment 4



Pyramid Blending tweaked to handle non-image area 0 pixels

Setup and Panorama Type

- Describe your setup for your camera. In addition, make sure to include relevant camera parameters (like ISO, etc.)

I used a nikon D3500 mounted on a tripod for stability. The lens used is AF-P Nikkor 18-55mm. The camera was for an exposure time was 1/5 s, f-number 3.5, and ISO 800. The settings were selected to ensure the sensors gets enough light for a clear photo. The camera was focused using the tree in the middle of image 1.

- Describe the camera movement that you used when you took your pictures.

In order to make panoramic images, it is essential that there is "sufficient" overlap between images. The camera tripod mounting fixed had the ability to rotate in place. I tried 3 different rotation angles (~10 degrees, ~7.5 degrees and ~5 degrees). The pictures taken using 5 degrees seemed to have enough overlap to prevent distortion and undesired stretching artifacts in the images.

- What kind of panorama did you make?

Based on the the reference text [1][2] I made a "Rotational Panorama". Since the camera did not translate but was in pure rotation the panorama type was Rotational Panorama.

Original Input Images

Image 1

Image 5

Image 9

The scene of interest is the view from my office door. The scene is comprised of a few trees, a building under construction, a parking lot with painted white lines for parking spots, a few cars in the background on the left and a car in the lower right corner. The time of day is around 6:30 pm and hence low light. The building under construction has lots of wooden frames and metal support structures which act as good features for matching and stitching. The scene also has a temporal component which affects the panorama: Image 5 has a car turning in the lot with the headlights on.

Results - Original Images

The original images composition was blended in a few different ways and the best result is presented here. The number of matches was set 1000 for these result. This result was obtained by blending the images using 2 “alpha” parameters per pair. One alpha is a mask with constant element values and the other “alpha ” is a ramp down function in the masked area. Results obtained using other method illustrated on next slide.

More Results - Original Images Other Blending Functions

Pyramid Blending as implemented in Assignment 4

Using the regular pyramid blending scheme the panorama obtained had image edge halos primarily due to boundary condition violation apart from visible seams of different images, and color intensity issues.

Pyramid Blending tweaked to handle non-image area 0 pixels

The regular pyramid scheme was amended to attempt to satisfy the boundary conditions. The images being stitched were wighted linear combinations of the input panoramic images.

warpCanvas()

- Why multiply x_min and y_min by -1 in the warping function?

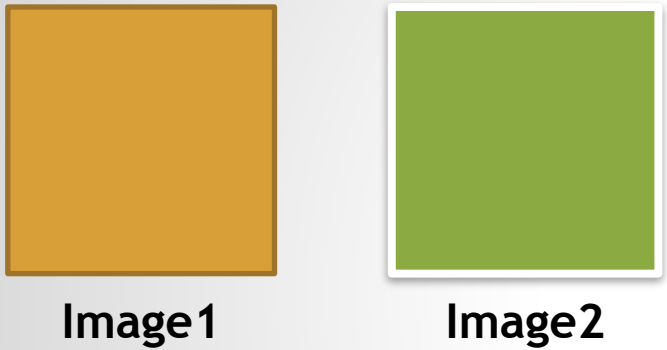
Using the findhomography() function and matched keypoints between the images, a 3x3 transformation matrix is found relating pixel coordinates of image1 and image2. getBoundingCorners() used the homography transformation matrix to transform image1 corners to space of image2 keeping image 2 fixed in its space. getBoundingCorners() further identifies the bounding coordinates of the a canvas/frame which will completely bound the transformed image1 and image2. Identifying the x_max and y_max is straight forward but x_min and y_min values might fall into 2 categories. The most trivial case is x_min or y_min are (0,0) independently it will not matter whether whichever or if both are 0 is multiplied by -1 or not. This scenario will happen when either x_min or y_min are corner coordinates of image2 or are also coordinates of image1 post homographic transformation.

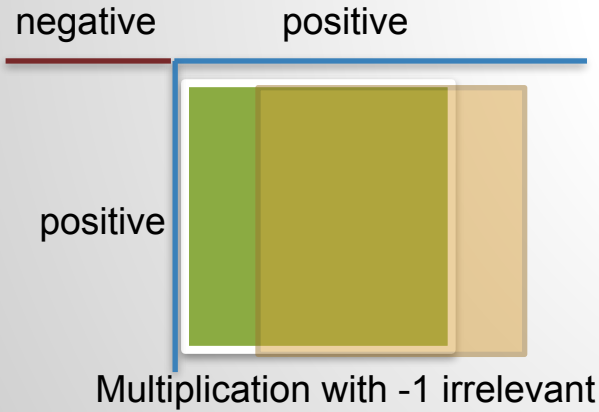
The second case is when x_min or y_min are negative in which case they fall outside the coordinate frame of image2 multiplication by -1 becomes important. This scenario occurs when some transformed coordinates of image1 are negative with respect to image2.

warpCanvas()

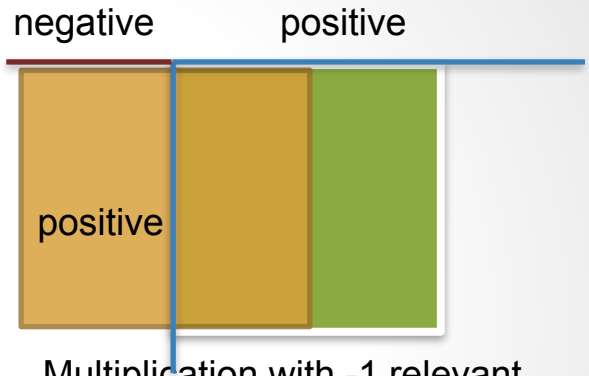
If some coordinates of image1 are negative then there are pixels of image1 which are falling out of the frame of image2. To rectify this the canvas/frame which holds transformed image1 and image2 needs to be enlarged so that all image1 pixels have +coordinates. Multiplying x_min or y_min or both which ever is the case with -1 and creating a translation transformation matrix for image1 enables us to move the pixels with negative pixels to the positive coordinate frame. This is done by matrix multiplication of homography matrix and translation matrix to create a compound transformation matrix which when used with the warpprespective() function warps image1 to an image frame that can hold both image1 and image2.

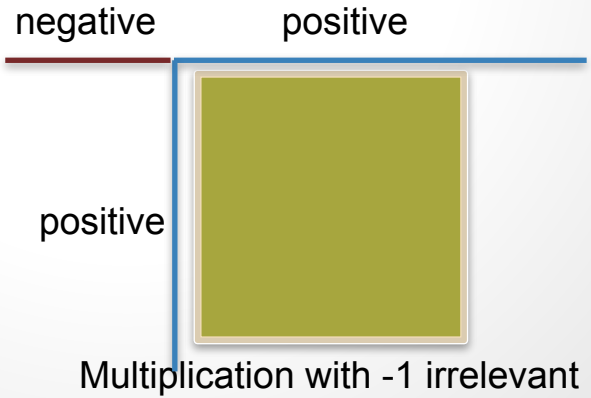
warpCanvas() coordinate systems representation





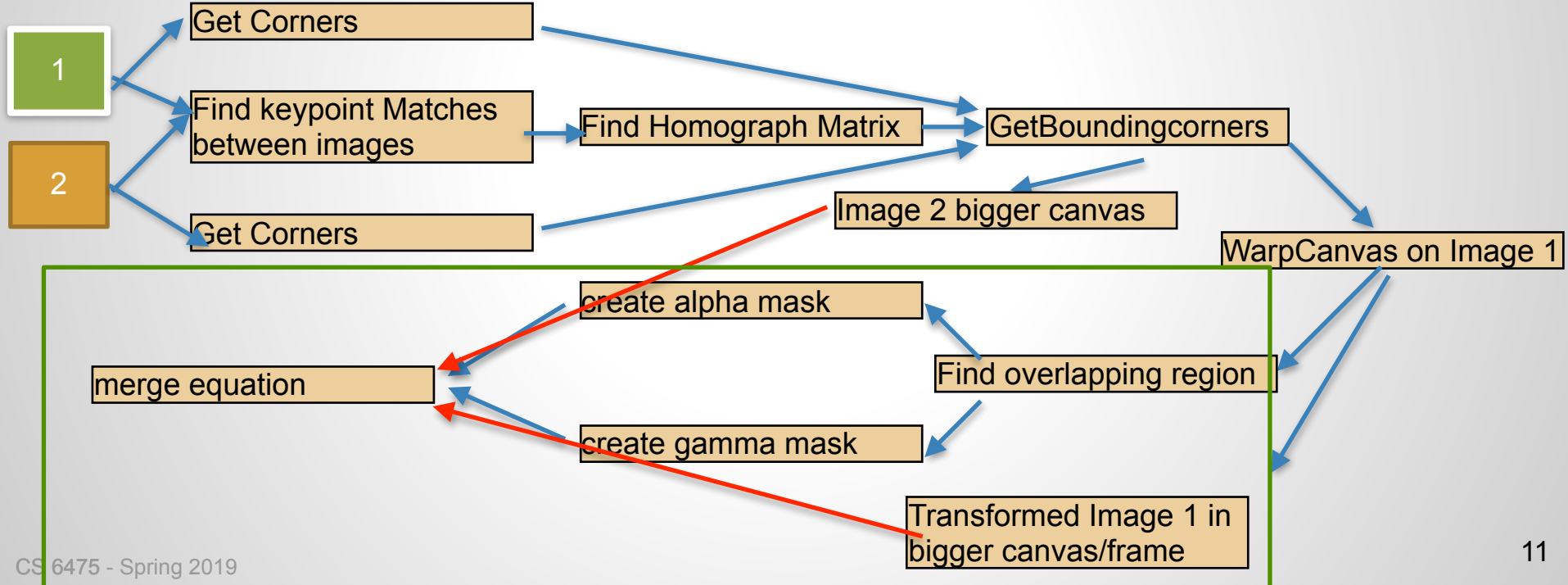
*Opacity added for clear illustration only





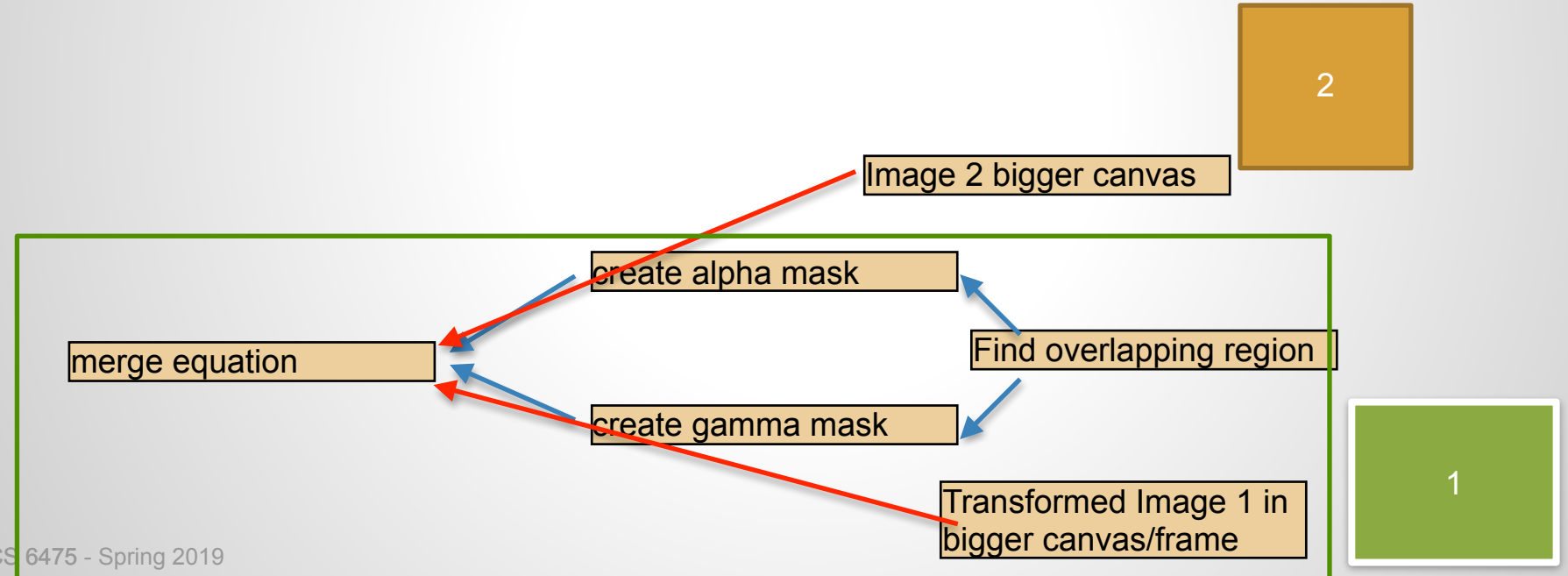
blendImagePair()

- Discuss your implementation for **blendImagePair()**.
  The blend function used for bonnie submission implements steps as illustrated in figure. The function snippet enclosed in the green rectangle will be elaborated.



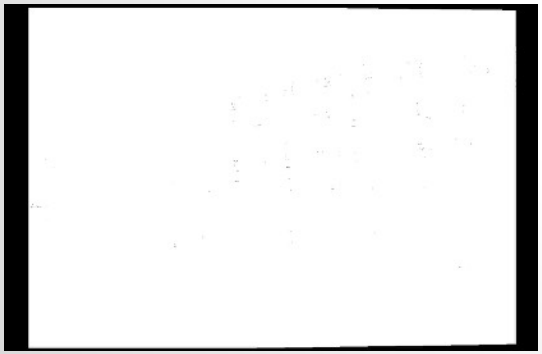
blendImagePair()

Gamma and alpha are masks which are used to merge the images excluding some portions of the images and merging the common portions.



blendImagePair(): Gamma

Gamma and alpha are masks which are used to merge the images excluding some portions of the images and merging the common portions.
Gamma mask captures the overlapping region with each pixel value set at 1.0.
The Gamma mask is inverted to find region of no overlap. This region will not be affected by blending. Inverted Gamma = 1 - Gamma. Both images when multiplied element-wise will contribute their respective non- blended regions. Since the Canvas is bigger than both images there will be some non-blended areas from both. The red boundary around 1 - Gamma is only to distinguish the image from the slide background.



Gamma

1 - Gamma

blendImagePair(): Gamma Implemented

x

=

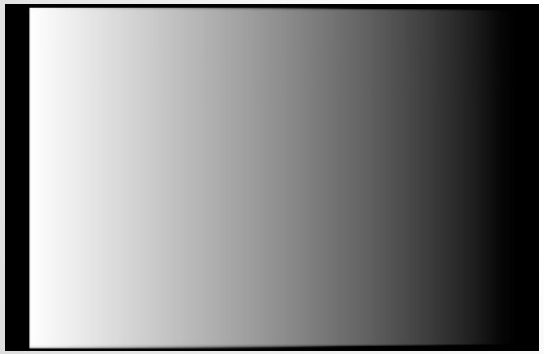
x

=

Images Element-wise x 1 - Gamma Contributions to delta panorama image

blendImagePair(): Alpha

Alpha mask is used to determine the contribution of images to the overlapping region. For the left image alpha is used (image1 in our case) and for the right image 1-Alpha The red boundary around 1 - Alpha is only to distinguish the image from the slide background. The white portion on the left side of 1- Alpha will not contribute as the left image has no valid pixels there. The Alpha function linearly goes from 1 to 0 from left to right thus reducing its contribution as it goes.



Alpha

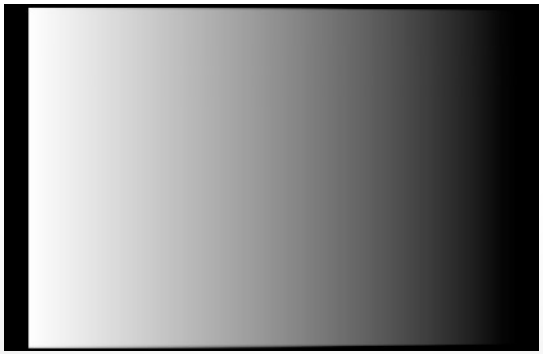
1 - Alpha

blendImagePair(): Alpha Implemented

x

=

x



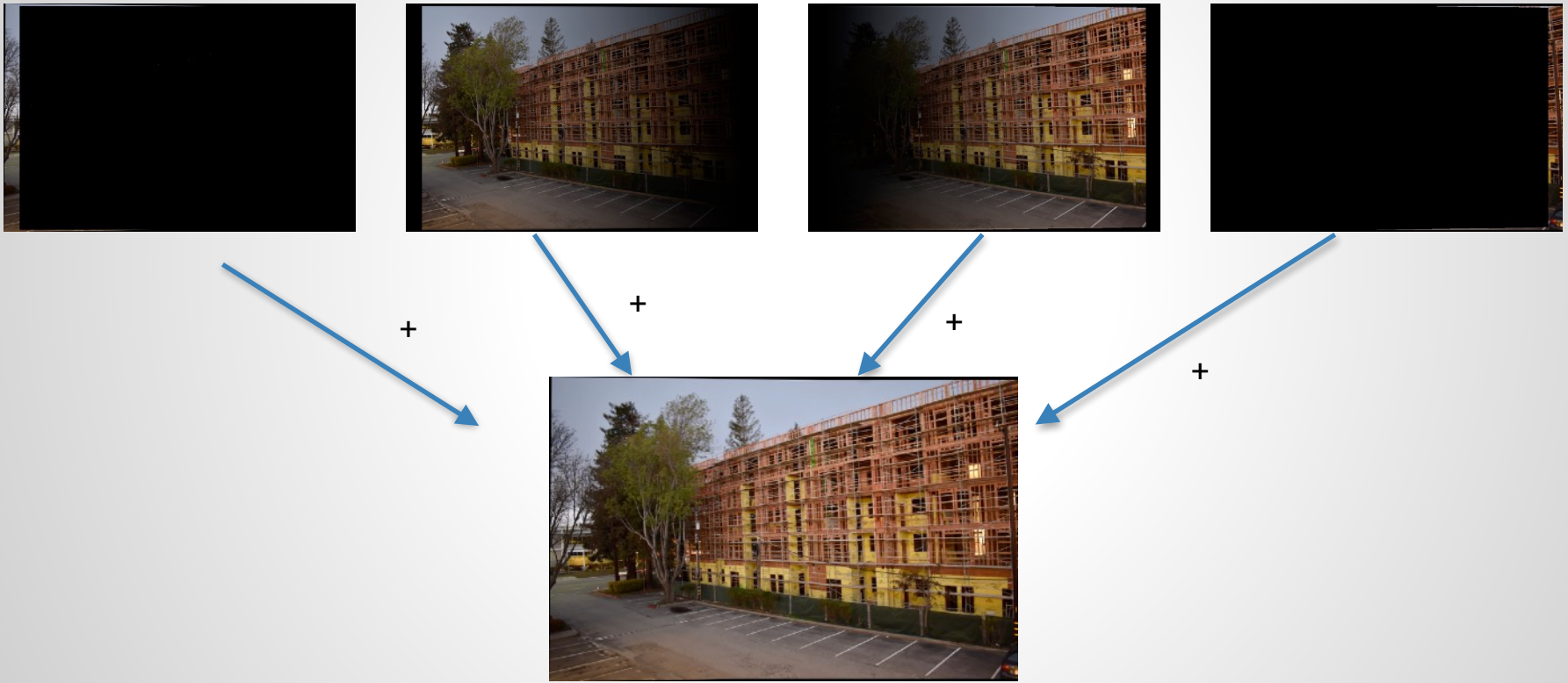
=

Images

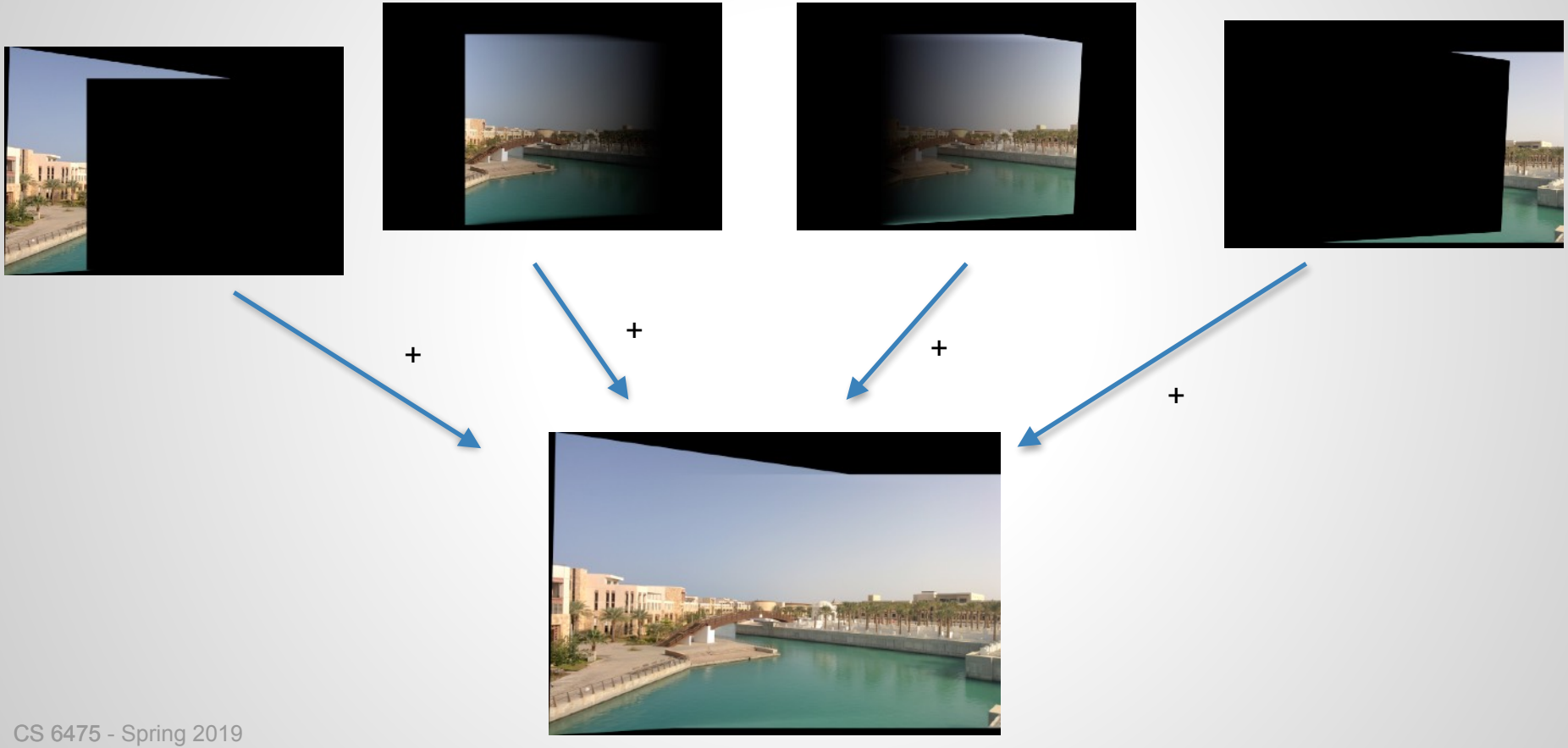
Element-wise x

Contributions to delta panorama image

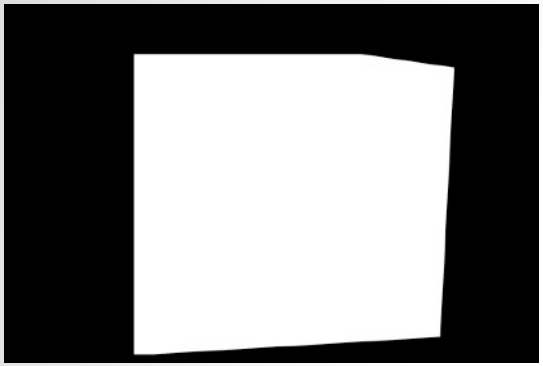
blendImagePair(): Blend



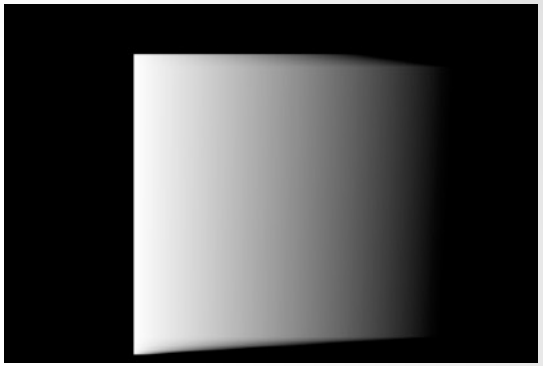
blendImagePair(): Blend Sample Test images



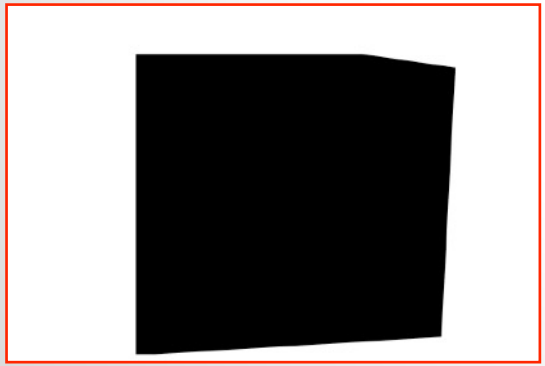
blendImagePair(): Gamma And Alpha Sample Test images



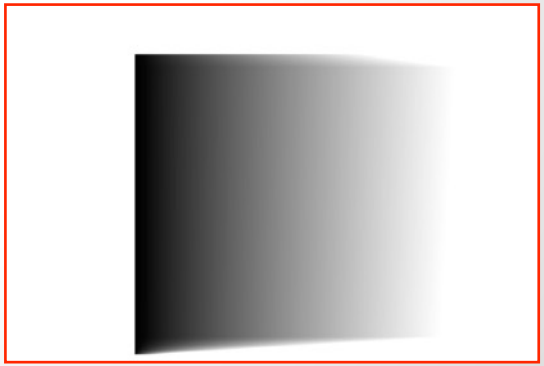
Gamma



Alpha



1 - Gamma

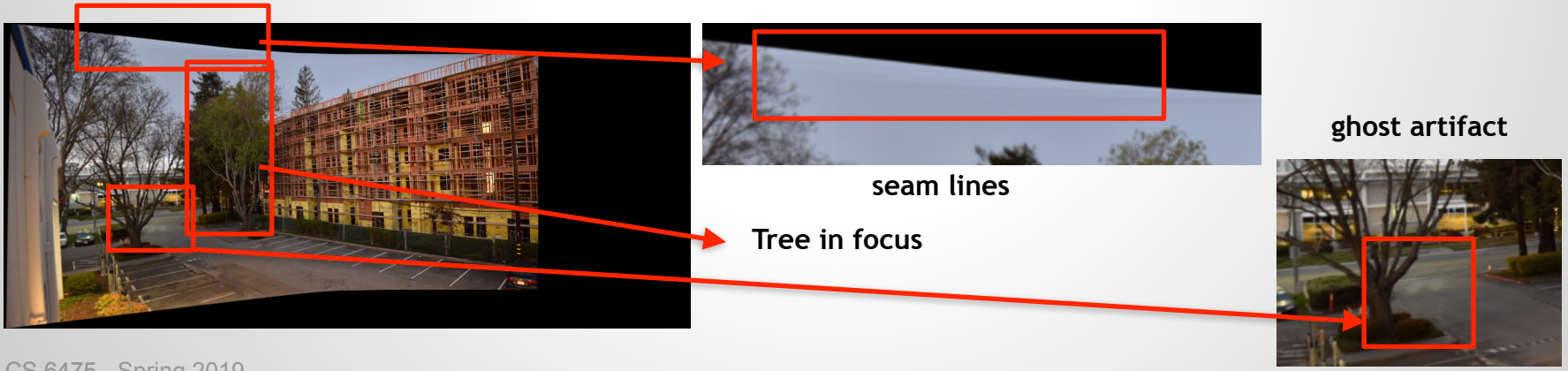


1 - Alpha

Discussion & Analysis

- Consider your blend function and answer the following questions - What worked well? What did not work well? Were there any problems you couldn't solve? If you had more time, what would you do to improve upon your existing blend implementation?

Multiple blend functions were implemented taking inspiration from assignment4, lecture, and szeliski. The function selected for bonnie submission was based on alpha blending which I have referred as alpha-gamma function. The function blended the images well in the horizontal dimension i.e. along the width but along the height dimension a slight seam lines is visible which run along the width. Straight and parallels lines have blended well and the focal object i.e. the tree is in focus. There are minimal distortions and the horizontal blend is seamless. The image can be slightly cropped along the seams on the edges of the panorama to get a better final image.



Discussion & Analysis

Image 1 no car present

Image 5 car present which causes ghost artifact

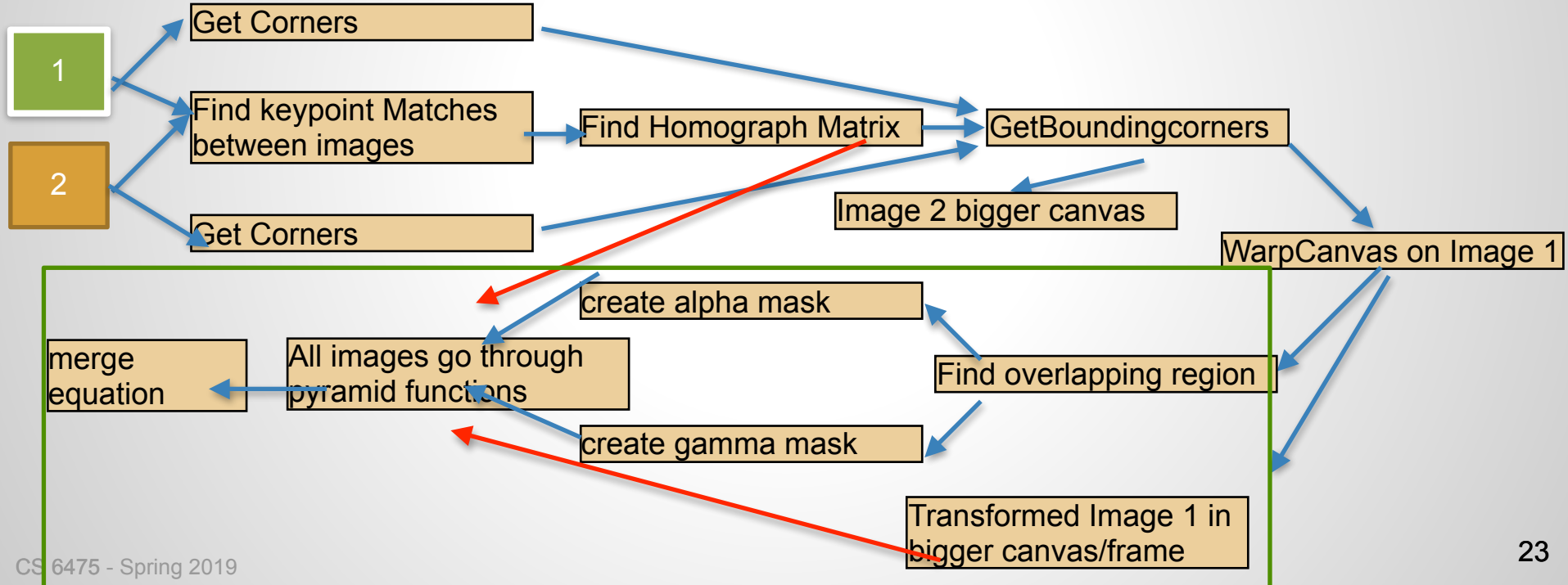
There is at least one ghost artifact due to the fact that there was a temporal component to the images. Even though I could not identify a ghost artifact due to blending there might be one. The blending in the vertical direction did not come out well. This is due to intensity differences and optical phenomena along the edge location of the images. Function to manage intensity difference like gain compensation might help. Also de-ghosting using a technique like poisson blending might help or it might actually give a more robust blend. The blending function required rotation per image to be small ~5 degrees for smooth horizontal blend. Which reduce the over all image size with respect to the number of images used. An improvement to the blend function to handle overlaps as low as 20% will make the algorithm more robust.

Discussion & Analysis

- If you started the project over again, what could you do differently and how would you go about doing it? (Do not say there is nothing you could do differently. There's always a way to improve.)

A few things can be done differently to improve the project. The functions need to be rewritten to speed up computation to handle large images or large number of images. Currently, I adjusted the size of the images and the number of images to reduce compute time to a minute or so. I attempted to use different blending techniques but ended up blending length-wise. Implementing a function or algorithm to make blending direction invariant or implement blending in vertical direction would be another thing. I spent a lot of time attempting to implement pyramid blending even though I knew the boundary conditions were not easy to handle. I would attempt to implement poisson blending which upon review of the paper "Poisson Image Editing" by Perez et al seems appropriate for blending together the edges seamlessly.

Above and Beyond*: Blending Function Pyramids*



Above and Beyond*: Blending Function Pyramid 1*

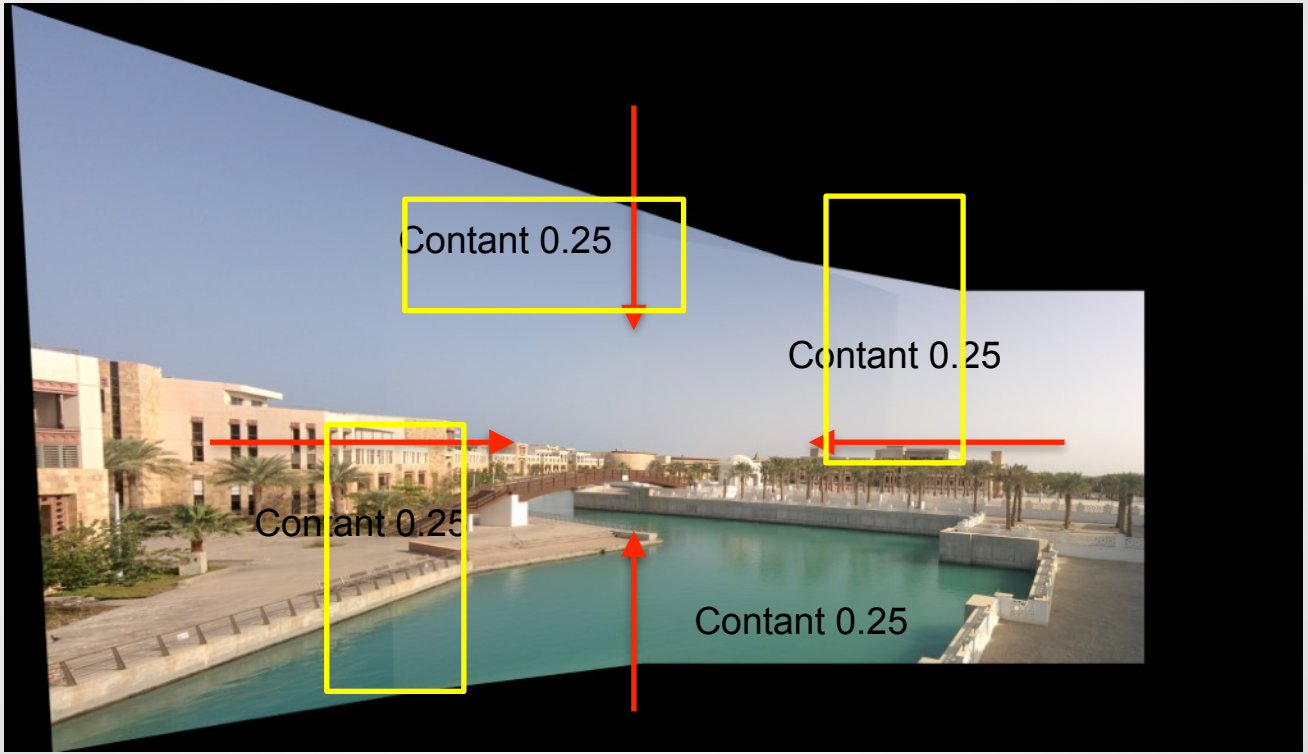
Pyramid blending applied on images of the left independently and then combined
The resultant image as artifacts due to boundary violation.

Above and Beyond*: Blending Function Pyramid 2*

Pyramid blending applied on images of the left as augmented with masks and then combined. The resultant image as ***weaker*** artifacts due to boundary violation.

Above and Beyond*: Blending Function .25 alpha*

Quater Alpha Function: regular linear alpha averaging. Alpha is split equally into 4 components for left, right top and bottom contribution. Smooth blending on half lengths in each direction. Artifacts highlighted



Resources

[1] Udacity lecture 5-03

[2] Szeliski Chapter 9

[3] Assignment4 blending functions