HW6: Synthetic Aperture Imaging

Yiming Dai, YDF9097

1. Capture an unstructured light field

We used our phone for this part to capture a video. We moved the camera in a zig-zag motion. The scene we design has a few objects at different depths. After capturing the video, we speeded up the video to make it not too long.

2. Register the frames of video using template matching

(1) Load data into Matlab

Once we have captured a video, I use Matlab to register each frame of captured video and generate synthetic aperture photos. I used the built-in Matlab VideoReader object to read the saved .mp4 file into a Matlab array. The number of frames is 167.

I stored each colored frame in a 4-dimensional matrix. Since registration should be performed on grayscale video frames, I also converted each frame of video to grayscale and stored each of them in a 3 dimensional matrix.

The first grayscale image is as follows:

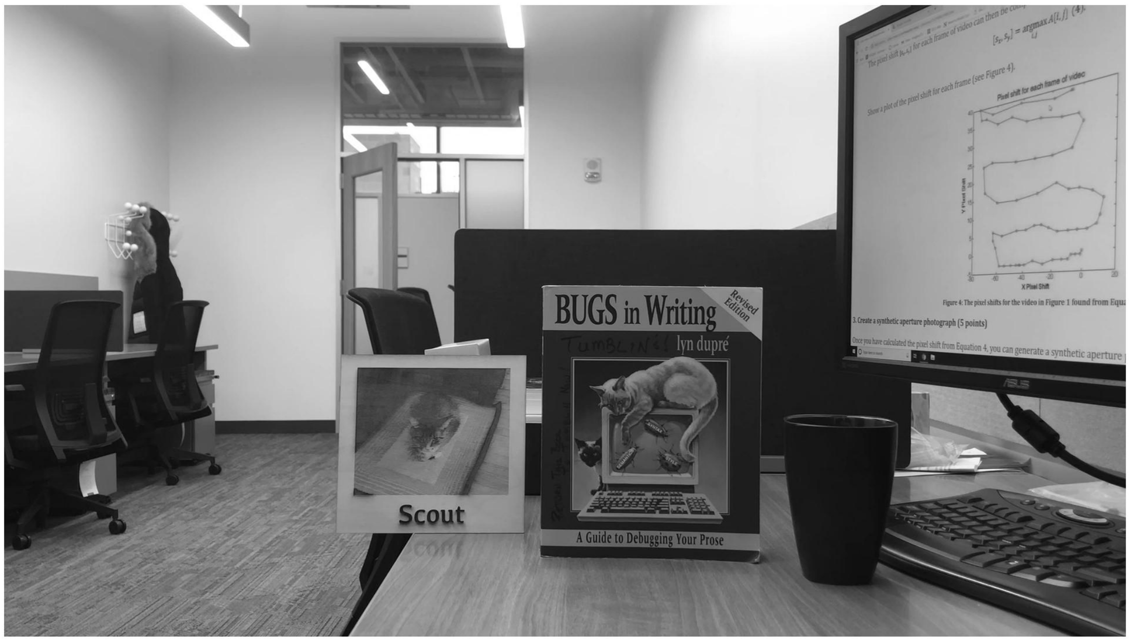

Figure 1 1st frame

(2) Select template and window

I tried several templates and chose a 50*50 pixels size template, which located on the book letters part. I displayed the template of 1st frame and the last frame, which were almost the same, to determine the maximum shift of the template. And I selected a search window of 500*500 pixels.

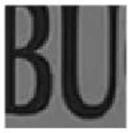

Figure 2 template of 1st frame

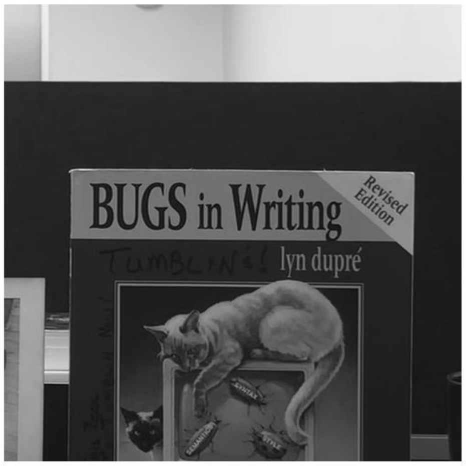

Figure 3 search window

(3) Perform autocorrelation of the template.
My template $t[n,m]$ is T*T = 50*50 pixels and my window $w[n,m]$ is W*W = 500*500 pixels. I can compute the normalized correlation function $A[i,j]$ using the equation:

$$A[i,j] = \frac{1}{N \cdot P} \sum_{n=1}^{T} \sum_{m=1}^{T} t[n,m] \cdot w_{ij}[n,m]$$

where w_{ij} is a block of pixels taken from the window w that is centered on the pixel location (i,j)

$$w_{ij}[n,m] = w^2 \left[n + i - \frac{W}{2}, m + j - \frac{W}{2} \right]$$

and the normalization factors are computed as

$$N = \sum_{n=1}^{T} \sum_{m=1}^{T} t^2[n,m]$$
$$P = \sum_{n=1}^{T} \sum_{m=1}^{T} {w_{ij}}^2[n,m]$$

The pixel shift $[s_x, s_y]$ for each frame of video can be computed as:

$$[s_x, s_y] = argmax_{ij} A[i,j]$$

I can find the maximum A's location of each frame and store the x and y coordinates in 2 separate 1-d vector. Then I can use the builtin Matlab function nlfilter to perform the sliding window operation of Equation 1

Thus I can show a plot of the pixel shift for each frames. The result is as follows:

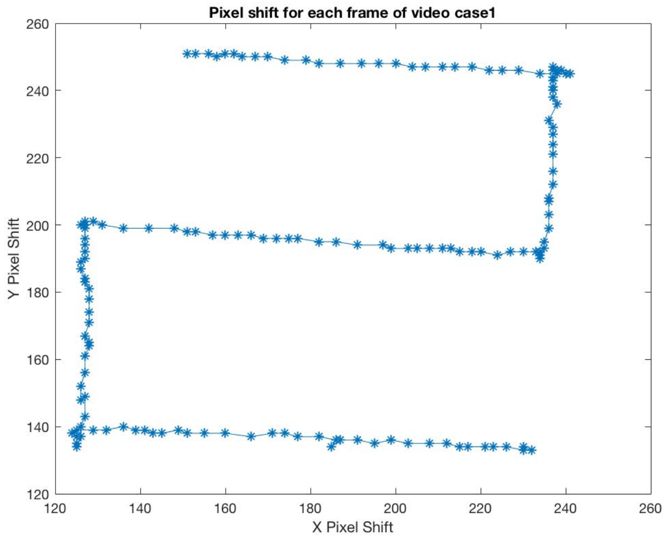

Figure 4 pixel shift for each frame of video

Problems I encountered:
1. At first, I did not quite understand the relationship between template and search window. After discussing with my teammates, I found out that the template we chose was from the first frame of the video, then in the searching area of each next frame, we can use the algorithm to find the most similar block and set its center as the next searching start location.
2. But the method of looping is really complicated and it costs much time to run the code. One of my teammate found a function called C = normxcorr2(template,A), which computes the normalized cross-correlation of the matrices template and A. The resulting matrix C contains the correlation coefficients. The principle of the function is the same as the 'looping' method, and it is quite simple and it saves time.

3. Create a synthetic aperture photograph

After calculating the pixel shift, I shifted the images of each frame of video in the opposite direction and then summing up the result. I calculated my synthetic aperture photograph as

$$P[n.m] = \sum_{i=1}^{N} f_i[n - s_x^i, m - s_y^i]$$

where $f_i[n, m]$ is the ith video frame, and $[s_x^i, s_y^i]$ is the shift for that frame.
By using builtin Matlab functions maketform and imtransform, I could generate a synthetic aperture photograph.
At first, I use gray images to test my code. The result is as follows.

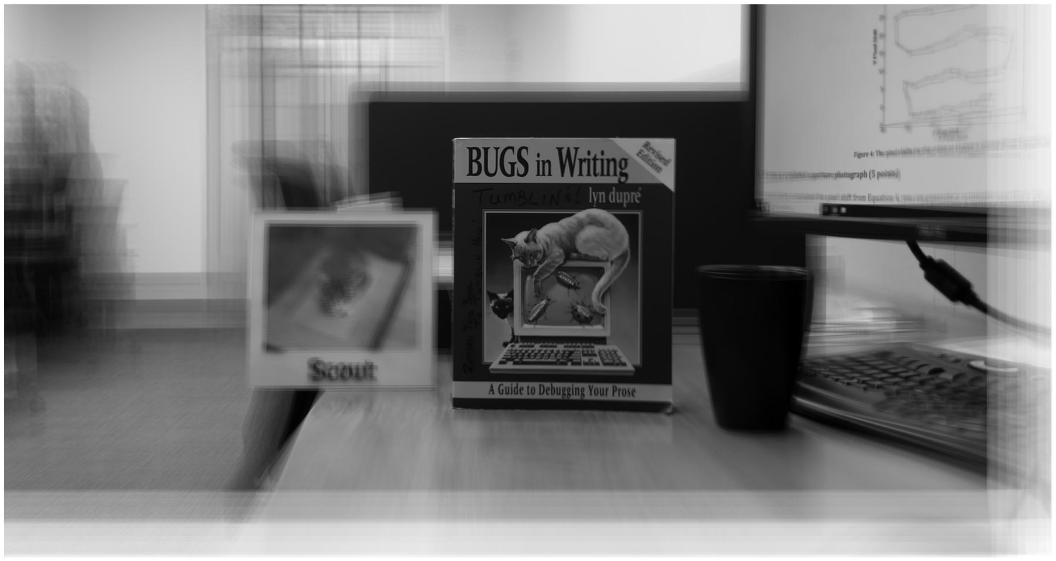

Figure 5 gray synthetic aperture photograph

Then I can implement the function by using colored images. The result is as follows.

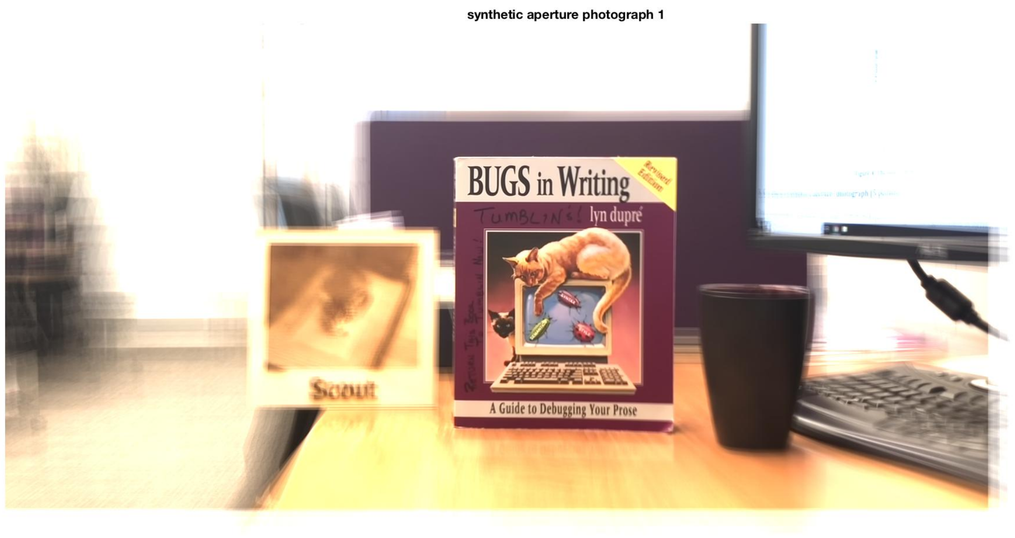
synthetic aperture photograph 1
BUGS in Writing
lyn dupré
TUMBLINE!
Revised Edition
A Guide to Debugging Your Prose
Scout

Figure 6 synthetic aperture photograph

As we can see from the above result, the surrounding region of the template is very clear and the part outside the region is blurred.

Problems I encountered:
The two builtin Matlab functions maketform and imtransform is a little complicated. I searched them on the Internet but could not implement them well. I turned to TA for help and solve the problem.
When using colored images for the code, I have learned that we can use the colored image matrix to implement the function. At first I simply add all photos together, but this could not work. TA told me to use a 4-D matrix to store all photos and 'mean' the matrix before displaying the result photo. The result was quite dark. Then I multiplied the value of pixel by 10, the final result was great.

4. Refocus on a new object
In this part, I refocused on a new object. I selected a new template from the first frame of video to be centered on a different object. Then I repeated previous parts.
The template, searching window and synthetic aperture photograph are as follows:

Figure 7 template of new object

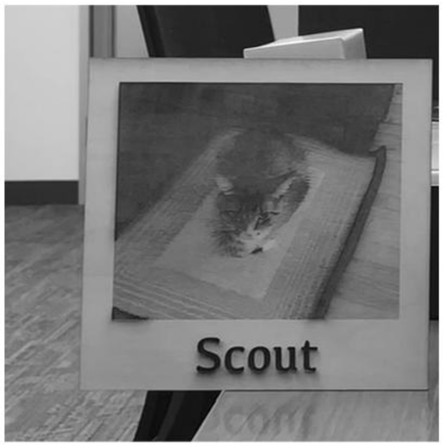

Figure 8 searching window of new object

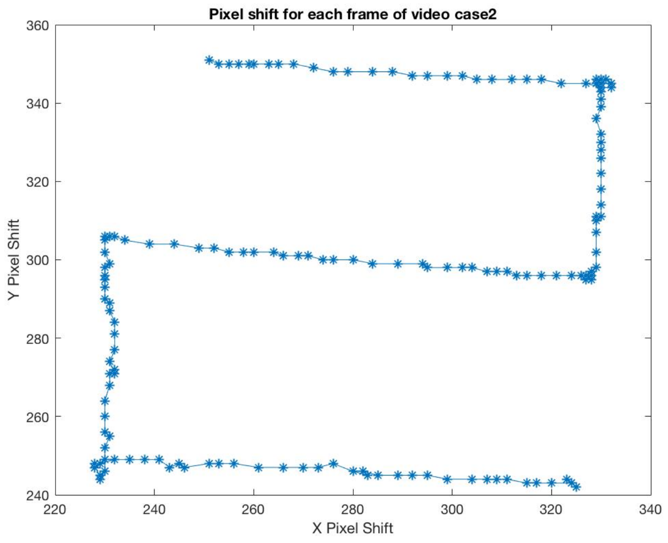

Figure 9 pixel shift for each frame of video synthetic aperture photograph 2

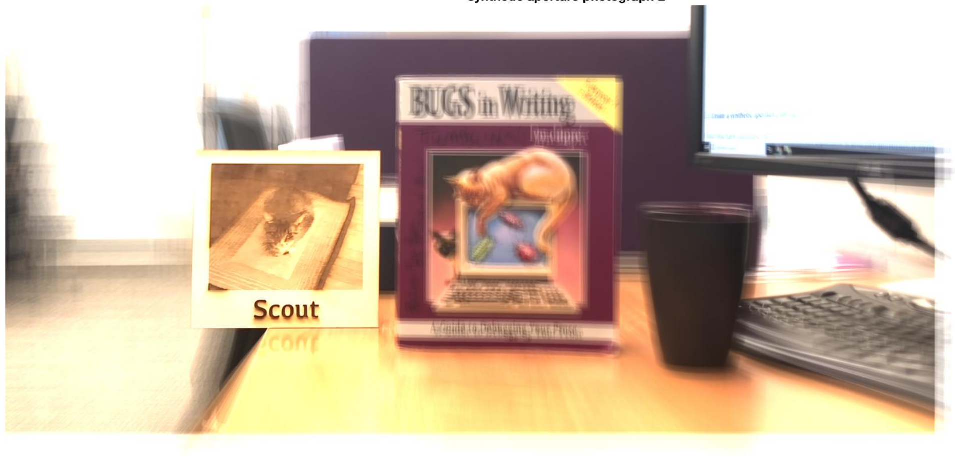

Figure 10 synthetic aperture photograph