Ikhlas Attarwala / Fall 2019
EECS 331 / Computational Photography
Group #9 (w/ Yunhao Li, Deanna Dimonte)

Programming a Macro-like effect for Images with Large Depth-of-Fields

Quick note before I begin:

- My results are awesome, but the only sad thing is that I had to turn in the assignment late because I lost 5 hours trying to wrap my head around the final bug (im2double vs uint8). So, to make up for the late submission, I did some extra work.
    - **main1.m** is the beginning of my work on the video '1.mp4'
    - **main2.m** is the same process as main1 but with changes to address my second video, '2.mp4'
    - **main3.m** is like main2.m, but for the video '3.mp4'
    - Each video contains a different capturing process that I discuss below.
    - '*_full*' at the end of an image stands for processing all frames; '*short*' processes ~60-80 frames

This was honestly a very fun assignment, I've done work similar to this before, but I was instead trying to transfer small depth-of-field effects from one image to the details of another image that was taken with a smaller aperture. Anyway, this week I synthesized a new image that appeared to have an expensive, large aperture, but was actually only taken from my phone. The trick was to compute the image from a video and use the motion from the video to calculate the depth of an object by its shifting pace. Upon selecting a template, I was able to refocus the image at that depth.

I began the experiment by cleaning my room for the first time this quarter. I found some boxing gloves, a water bottle and a sketchbook in the process so I used them as the objects I wanted to focus on, because each had some degree of texture or writing that made it possible to focus on it easier. Tilting or rotating the camera while taking a video of the objects adds additional dimensions to compute to offset the motion, so I tried to keep the camera motion as flat and planar as possible by brushing it up against a wall the entire time of the video feed. **Figure 1** shows one of the positions I used so that each object was placed at different distances away from the camera. I took 3 videos, one where I slowly zig-zagged around the plane perpendicular to the scene ('1.mp4'), one where I zig-zagged quicker (this resulted in a little bit more blurrier images) ('2.mp4'), and a third video where I used circular motions to cover as much of the plane as possible ('3.mp4'). Once I uploaded the videos to MATLAB, I converted the images to grayscale for faster processing, and then scaled back how many frames I needed for the assignment, roughly 60-80 frames in total (later I compare the differences between processing the whole video vs with fewer frames). It was a mistake to do it this way, as in order to generate a colorful output at the end, it's wiser to reverse the order by scaling back frames first before converting to grayscale.

The next goal was to manually select a template (a small piece of the image on one of the objects) that I wanted to match in each video. In the previous steps when I was recording, I kept in mind which template I was going to use for this step and made sure not to cover them as the camera was moving around. Template-matching is an important technique for finding small parts of an image in a larger one, and we want to use it so that we can measure the distance an object moves from frame to frame. There are a large multitude of programs, incl. OpenCV, that have packages to template match for you. We did it the old-fashioned way by manually selecting a window that we knew the template would always stay within through all frames, and computed the zero-normalized cross-correlation for all template-shaped boxes within that window. Ollie's website has a few errors so Yunhao and I talked to him after class on Tuesday to show what needed to be changed. This is the proper equation:

$$A = \frac{\sum_{u,v} \hat{I}(u,v)\hat{T}(u,v)}{\sqrt{\sum_{u,v} \hat{I}(u,v)^2 * \hat{T}(u,v)^2}}$$

$$where\ \hat{I}(u,v) = I(u,v) - \bar{I}, and\ \hat{T}(u,v) = T(u,v) - \bar{T}$$

I tried to compute this exactly as the formula is written, and I have before, but for some reason my *nlfilter* function in MATLAB to slide a smaller template-sized window over the larger window wasn't working, so I improvised and wrote a function at the bottom of the page called *ncc* to calculate the x- and y-offset and shift. Other than normalized cross-correlation, you could also use regular cross-correlation or even the sum of squared difference. Be creative!

This leads to **Figure 3**, where I show the computed pixel shifts per frame. On the left, we have the pixel shifts wrt. the template being on the water bottle; in the middle we have the pixel shifts wrt. the template being chosen on the sketchbook; on the right we have the shift of the template on the boxing gloves. The x-axis is on the bottom, and y-axis is on the left-hand side. Below each image is a 3-dimensional surface plot (of a single frame) that represents the intensities and peak of our normalized cross-correlation phase earlier. The peaks show us where the template matched highest and best.

Now that I calculated a list of each frame's template position shifts, from the first frame, I simply shifted the images in the reverse direction and averaged (summed and divided) the result across all frames. This is the step that caused me to submit late because converting the image to a double with *im2double* made my results go nuts, even though I had the answer all along. A friend told me to convert to uint8, and that solved the trick! Because the shift is calculated based off the template we choose and it's relative movement, we now have images that appear to be blurry everywhere else but the template region and its depth. For each of my 3 videos, I calculated the results of a template on each of the 3 objects.

After I was done, I decided to do another experiment and run the program without having scaled back any frames. In almost every comparison, while the template region was still quite visible in images where I cut back on the number of frames, the other regions were far blurrier, and didn't really give the feel of a high-quality camera. When I did incorporate every frame, the images were much better.

Additionally, I compared each of the 3 motions used to record the scene. In '1.mp4', I made an 'S'-like motion, as evidence in **Figure X**. '2.mp4' was an attempt to cover my space in the plane, however I also moved it rather quickly, so I noticed that there was more blurriness from the video itself. Finally, the results of '**main3.m**' on processing '3.mp4' was the best! The motion was wonky and crooked, but the attempt was to make it circular. This seemed to cover the most area on the plane and provide the best output, as evidenced by **Figure X**.

In the future, I should keep more objects around, and I feel like I kept the objects rather far in my last placement, so keeping them closer together would help reduce the image shifting to extreme levels such that they don't appear realistic. Overall this was fun!

Thanks for everything!

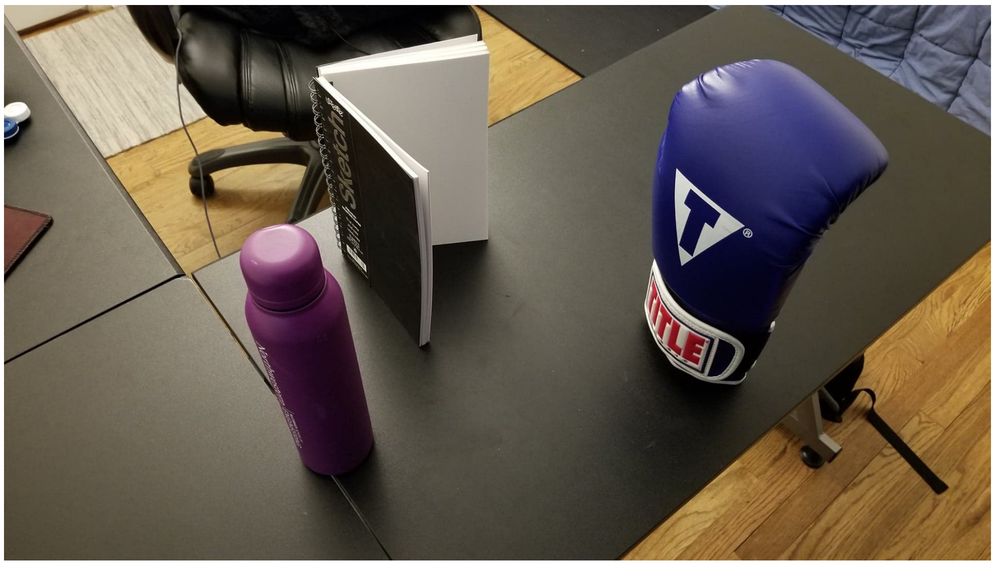

Figure 1: A side-view of the scene to keep objects at different depths (I later changed this layout)

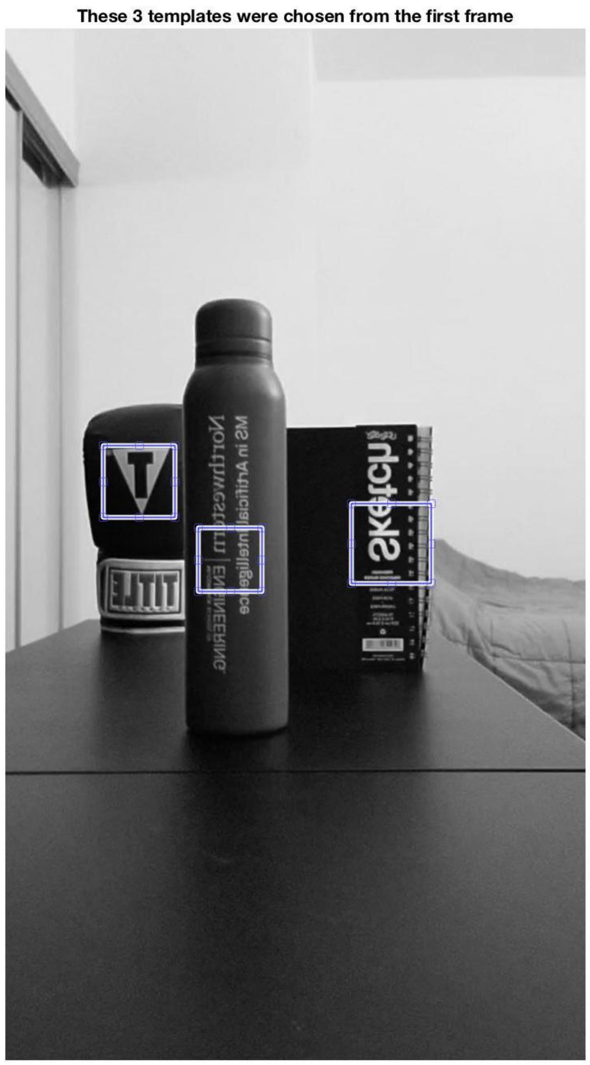

Figure 2: Original image in grayscale with the templates I selected in '3.mp4'

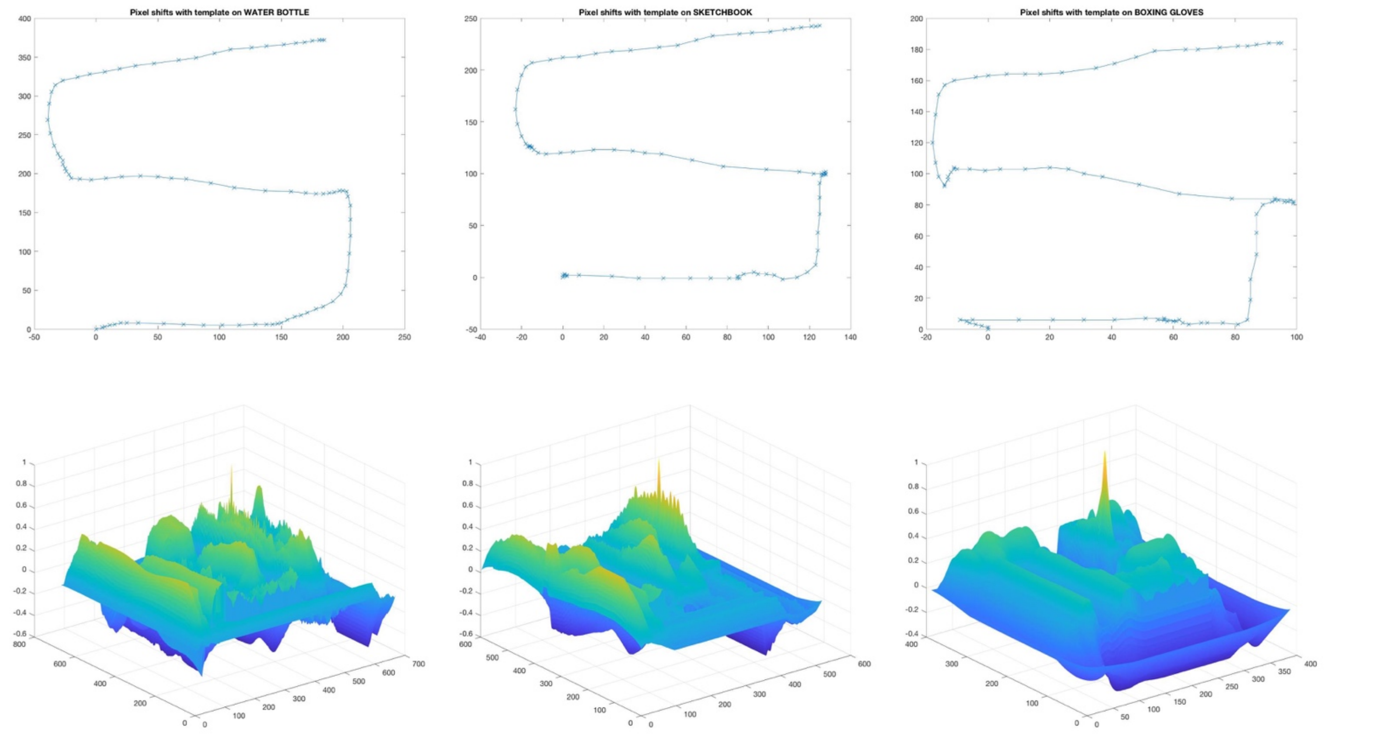

Figure 3: Pixel shifts and surface plots for ‘1.mp4’ (using slow zig-zag motions)
(from left to right: water bottle, sketchbook, boxing gloves)

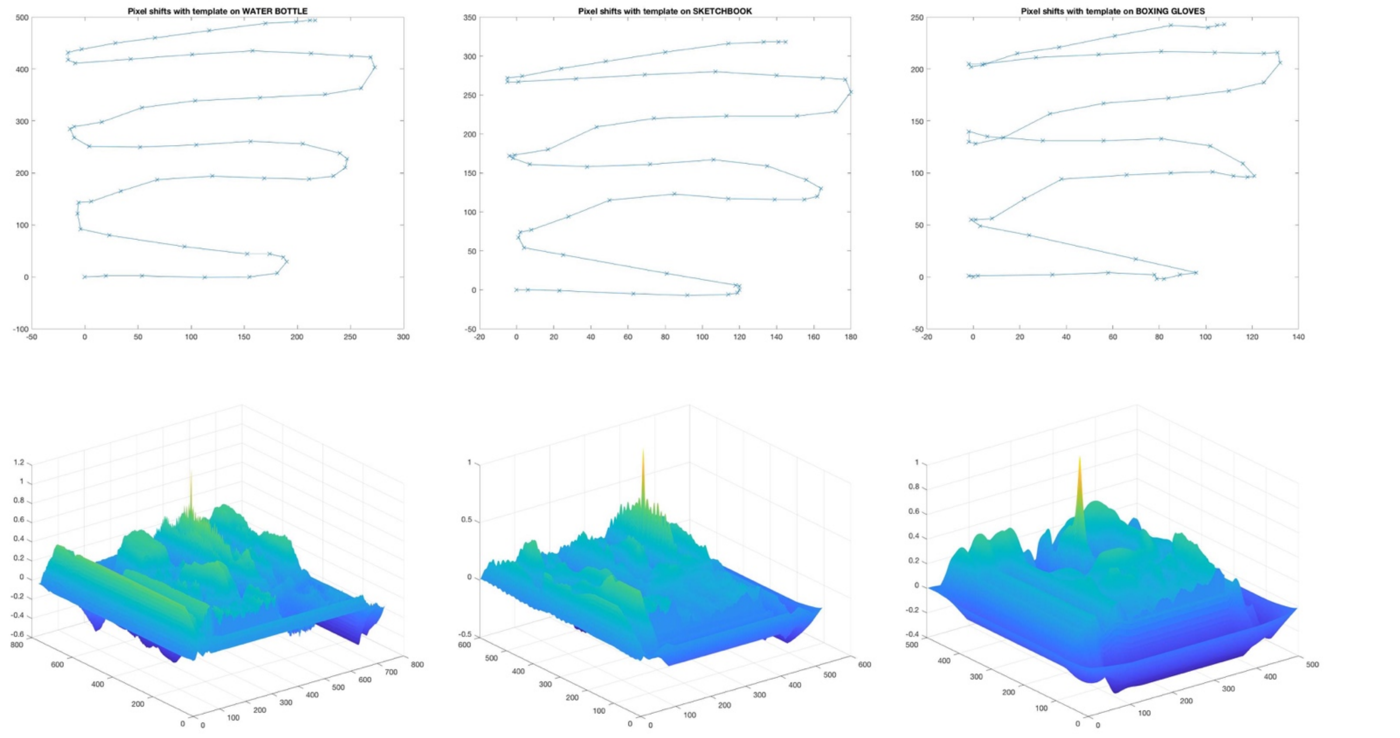

Figure 3: Pixel shifts and surface plots for ‘2.mp4’ (using many zig-zag motions)
(from left to right: water bottle, sketchbook, boxing gloves)

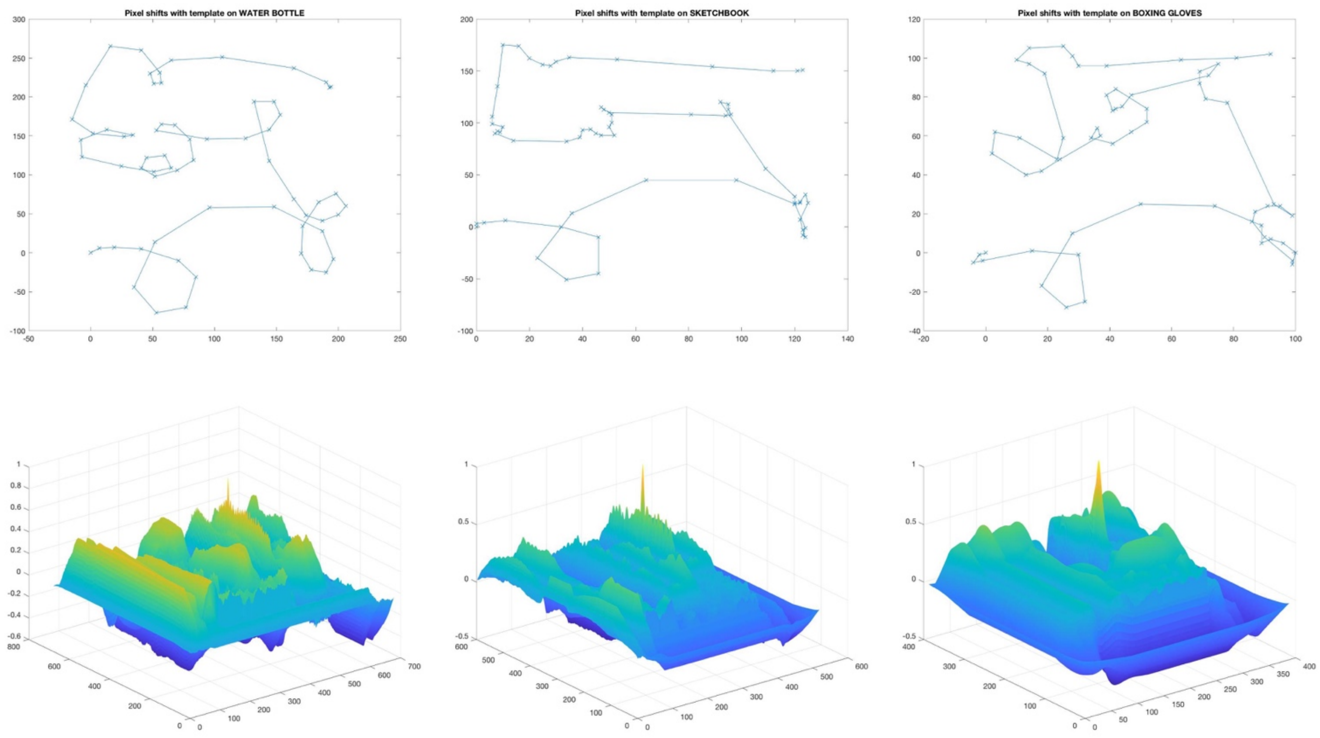

Figure 3: Pixel shifts and surface plots for ‘3.mp4’ using circular motions (had the BEST results!)
(from left to right: water bottle, sketchbook, boxing gloves)

(continue to next page for image comparisons of
full-frame processing vs scaling back to 60-80)
.
.
.

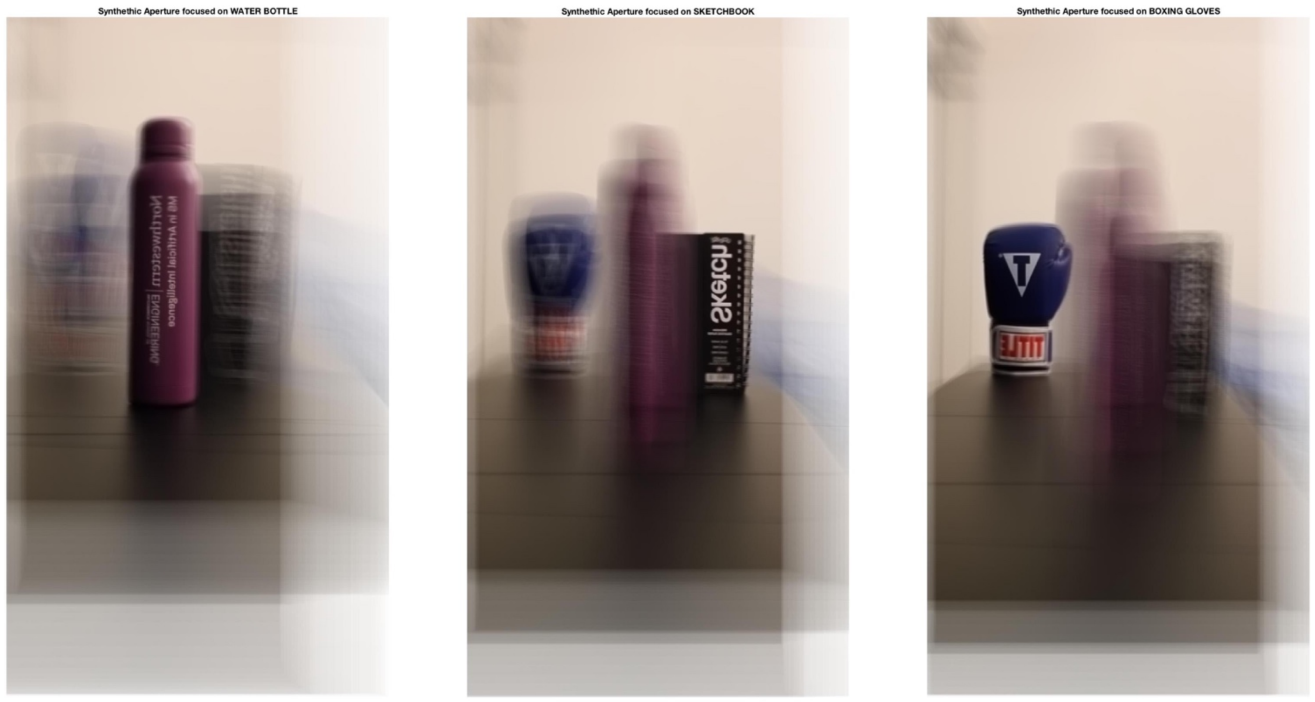
Synthethic Aperture focused on WATER BOTTLE
Synthethic Aperture focused on SKETCHBOOK
Synthethic Aperture focused on BOXING GLOVES

Figure 4: Synthetic Aperture results on vid '1.mp4' while scaling back to approx. 80 frames

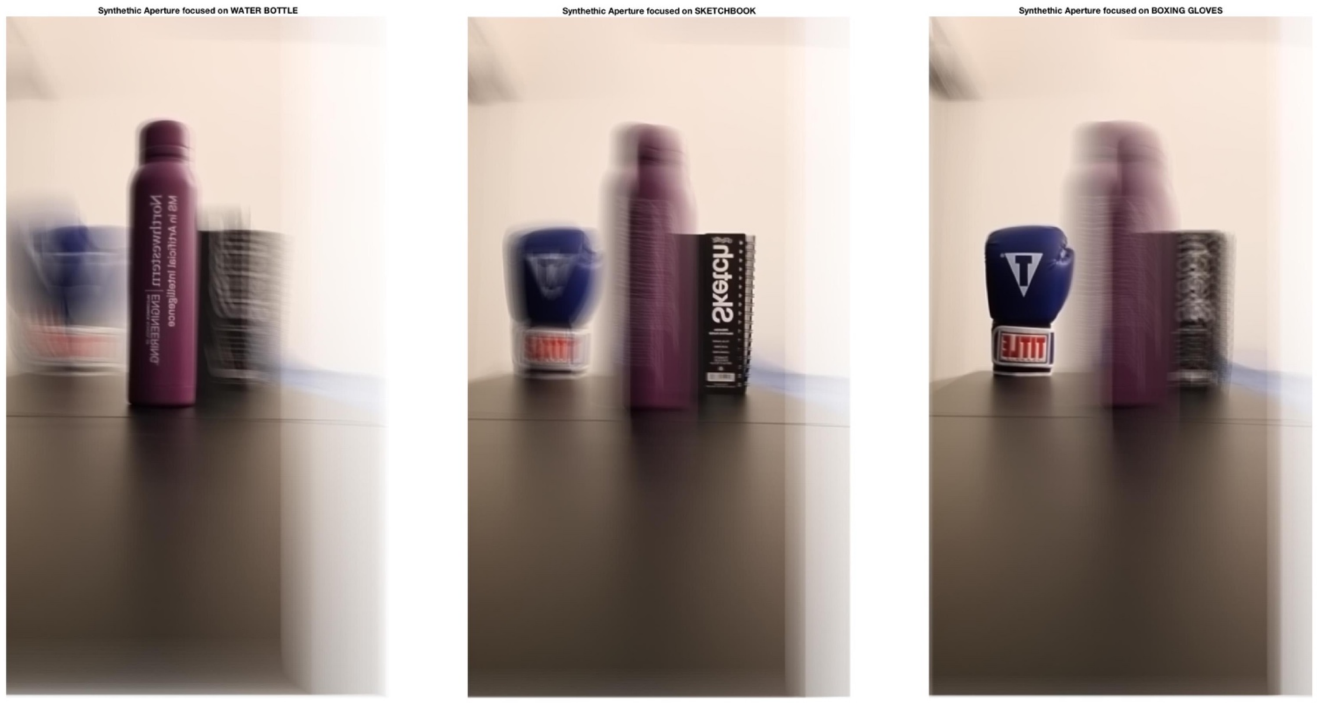
Synthethic Aperture focused on WATER BOTTLE
Synthethic Aperture focused on SKETCHBOOK
Synthethic Aperture focused on BOXING GLOVES

Figure 5: Synthetic Aperture results on vid '1.mp4' while keeping ALL frames in processing (less blur around depths out of focus, keeps it more realistic)

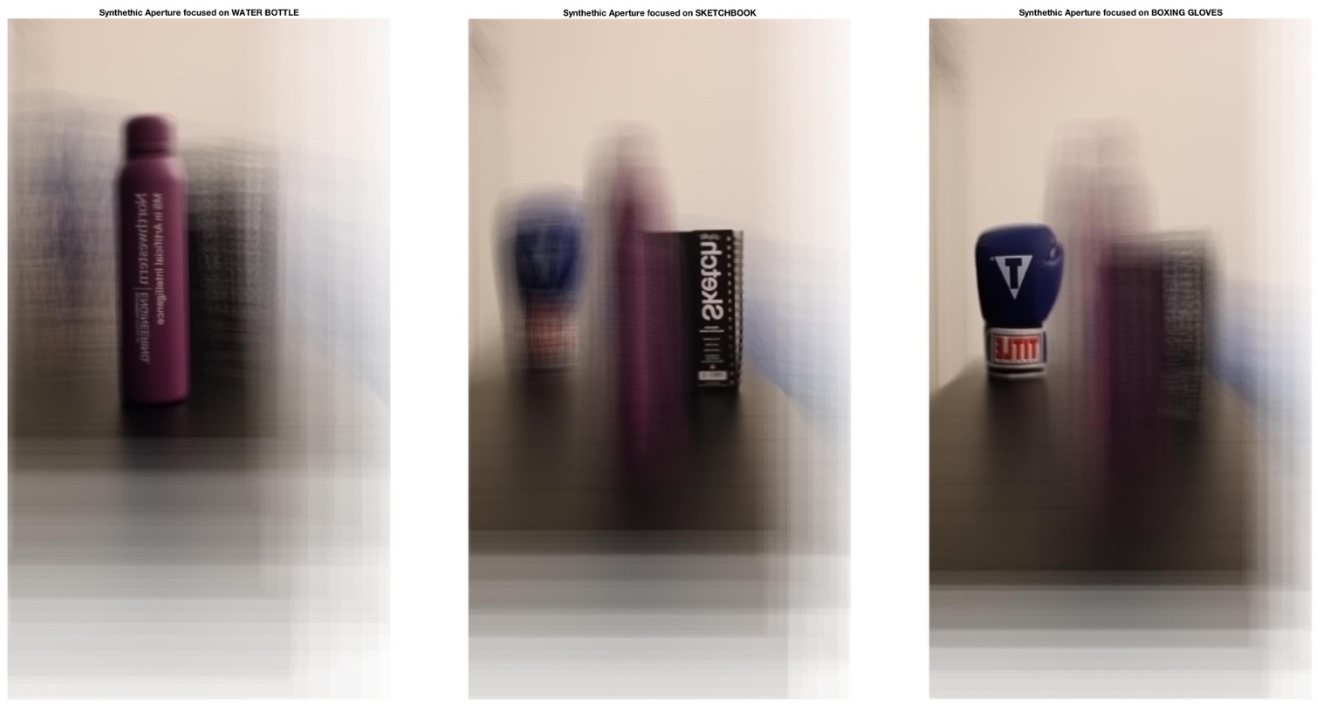
Synthethic Aperture focused on WATER BOTTLE
Synthethic Aperture focused on SKETCHBOOK
Synthethic Aperture focused on BOXING GLOVES

Figure 6: Synthetic Aperture results on vid ‘2.mp4’ while scaling back to approx. 60 frames

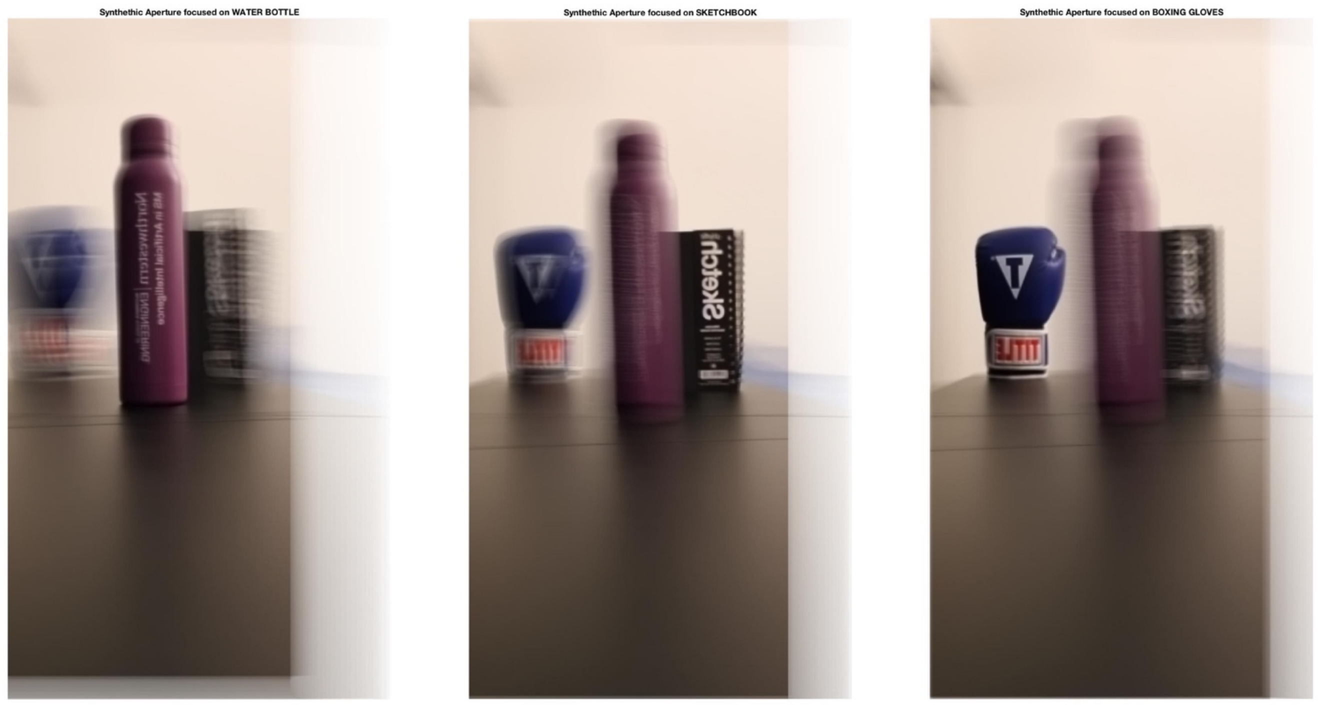
Synthethic Aperture focused on WATER BOTTLE
Synthethic Aperture focused on SKETCHBOOK
Synthethic Aperture focused on BOXING GLOVES

Figure 7: Synthetic Aperture results on vid ‘2.mp4’ while keeping ALL frames in processing (slightly better than from vid ‘1.mp4’)

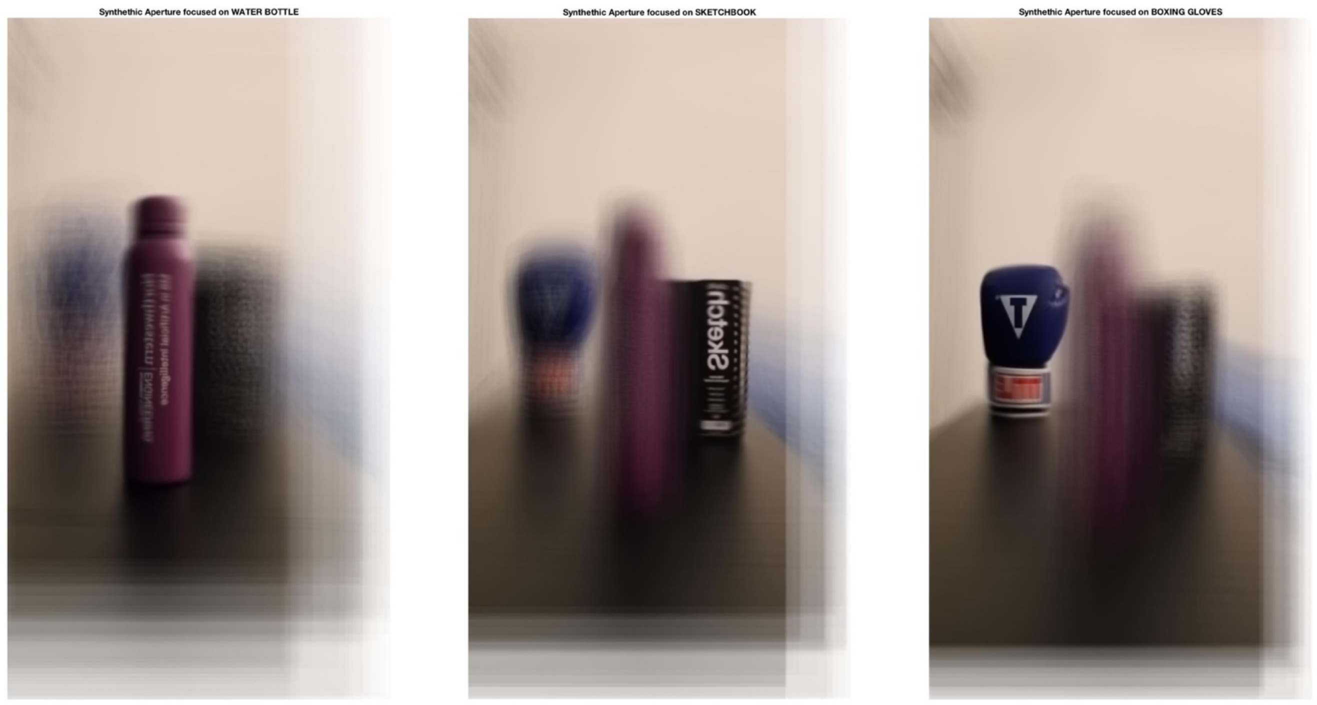
Synthethic Aperture focused on WATER BOTTLE
Synthethic Aperture focused on SKETCHBOOK
Synthethic Aperture focused on BOXING GLOVES

Figure 6: Synthetic Aperture results on vid '3.mp4' while scaling back to approx. 60 frames

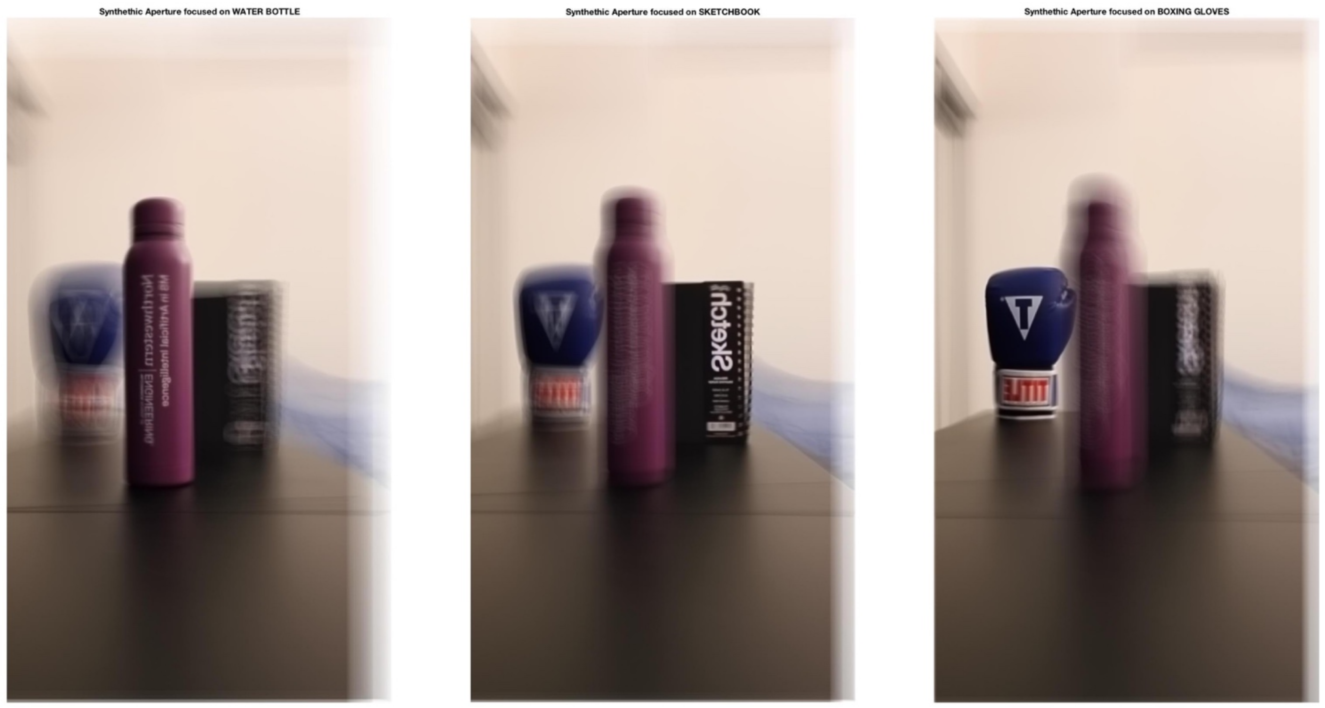
Synthethic Aperture focused on WATER BOTTLE
Synthethic Aperture focused on SKETCHBOOK
Synthethic Aperture focused on BOXING GLOVES

Figure 9: Synthetic Aperture results on vid '3.mp4' while keeping ALL frames in processing (BEST RESULT!)

You know, in every picture, the purple water bottle is less focused than the boxing glove/sketchbook. The template was selected fine, so the slight blurriness must be due to how the text isn't quite large enough and slight shifts have more of a drastic effect on shaking away its subtleness.