Name: Yuhan Shao
Student Number: 1002281327
UTORid: shaoyuha

CSC320 Assignment 1 Report

■ **Part2：Efficient Implementation** **p.2-3**

■ **Part3: Experimental Evaluation & Report** **p.4-18**

(1) The procedure used to capture the images p.4

(2) The range of image condition p.4

(3) The assessment of the technique's limits p.5

(4) A set of figures showing experimental result p.5-18

- **Setting1 & Conclusion: best p.5-6**
- **Setting2 & Conclusion: good p.7-9**
- **Setting3 & Conclusion: so-so p.10-11**
- **Setting4 & Conclusion: not bad p.12-13**
- **Setting5 & Conclusion: bad p.14-15**
- **Setting6 & Conclusion: bad p.16-18**
- **Overall Conclusion p.18**

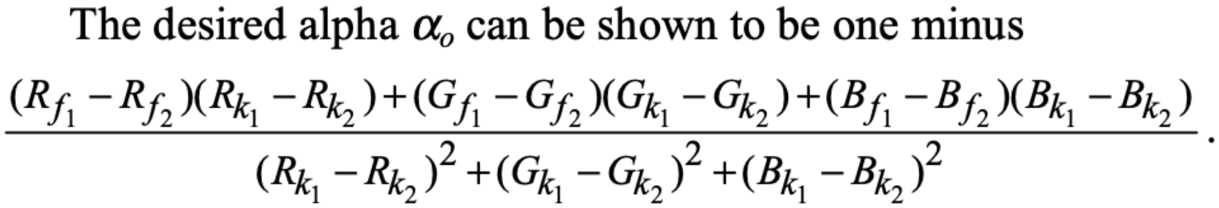
■ **Part4: Written Question** **p.19-20**

■ **Bonus: Living Matting** **p.20**

- **Part 2: Bonus Component: Efficient Implementation:**

In order to break reference solution's speed, I optimized the solution according to the reading material "Blue Screen Matting".

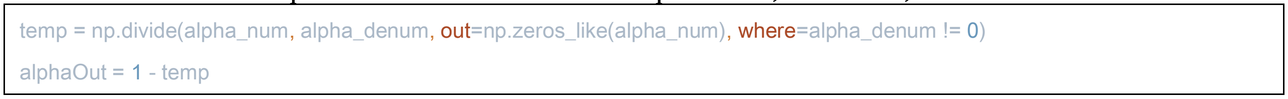
I calculated the alpha value with the equation on the page 6 as follow:



The desired alpha α_o can be shown to be one minus

$$\frac{(R_{f_1}-R_{f_2})(R_{k_1}-R_{k_2})+(G_{f_1}-G_{f_2})(G_{k_1}-G_{k_2})+(B_{f_1}-B_{f_2})(B_{k_1}-B_{k_2})}{(R_{k_1}-R_{k_2})^2+(G_{k_1}-G_{k_2})^2+(B_{k_1}-B_{k_2})^2}.$$

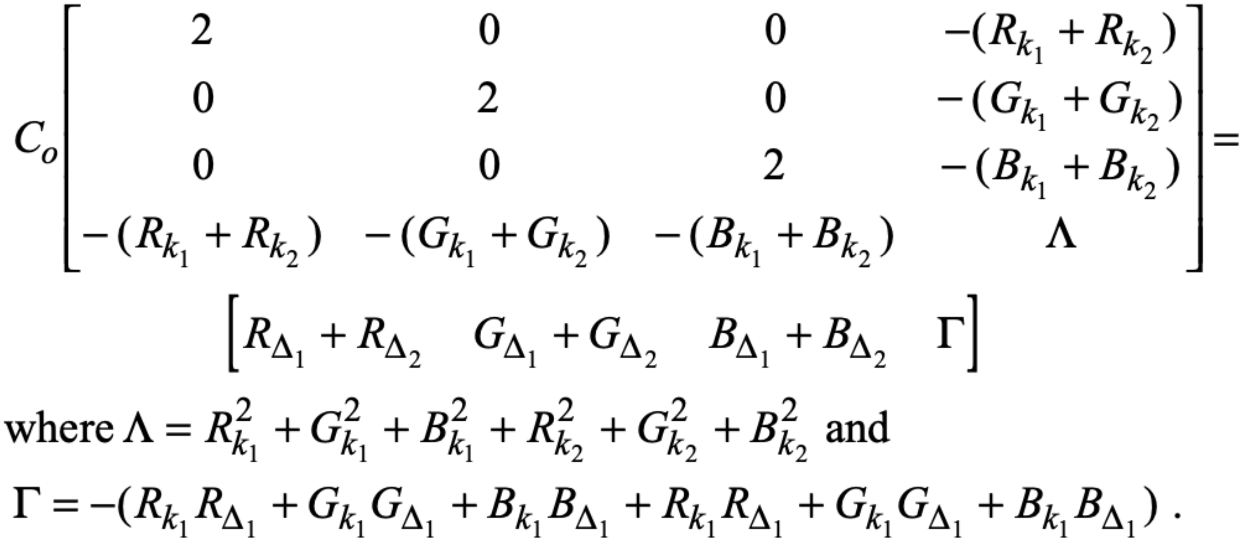
Note that division requires the denominator not equals to 0, therefore, I did a trick here with:



```
temp = np.divide(alpha_num, alpha_denum, out=np.zeros_like(alpha_num), where=alpha_denum != 0)
alphaOut = 1 - temp
```

Explanation: for each pixel, when the denominator equals to 0, it implies: R$_{k1}$ = R$_{k2}$, G$_{k1}$ = G$_{k2}$, B$_{k1}$ = B$_{k2}$, i.e.: the two pixels in the same position on the two back images have the same intensity. Since the pixel in this position should be considered as 'background pixel', we want it as "transparent" as possible. Hence, we give this pixel the alpha value 0.

After calculating the alpha value for each pixel, I plug the alpha value into the following equation on the page 5 as follow:



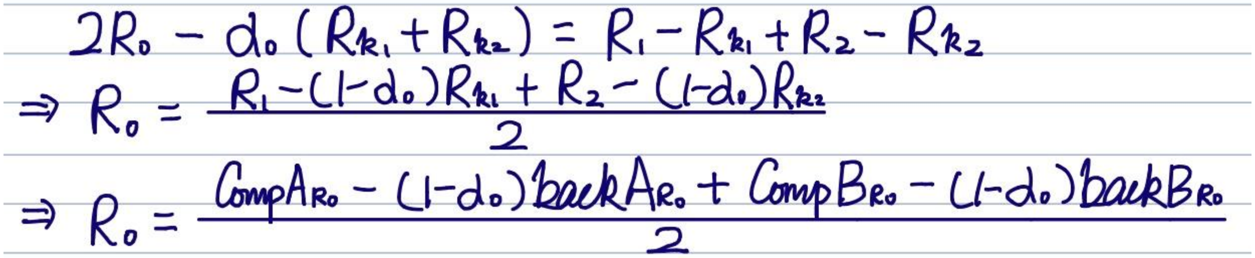
$$C_o\begin{bmatrix} 2 & 0 & 0 & -(R_{k_1}+R_{k_2}) \\ 0 & 2 & 0 & -(G_{k_1}+G_{k_2}) \\ 0 & 0 & 2 & -(B_{k_1}+B_{k_2}) \\ -(R_{k_1}+R_{k_2}) & -(G_{k_1}+G_{k_2}) & -(B_{k_1}+B_{k_2}) & \Lambda \end{bmatrix} =$$

$$\begin{bmatrix} R_{\Delta_1}+R_{\Delta_2} & G_{\Delta_1}+G_{\Delta_2} & B_{\Delta_1}+B_{\Delta_2} & \Gamma \end{bmatrix}$$

where $\Lambda = R_{k_1}^2+G_{k_1}^2+B_{k_1}^2+R_{k_2}^2+G_{k_2}^2+B_{k_2}^2$ and

$$\Gamma = -(R_{k_1}R_{\Delta_1}+G_{k_1}G_{\Delta_1}+B_{k_1}B_{\Delta_1}+R_{k_1}R_{\Delta_1}+G_{k_1}G_{\Delta_1}+B_{k_1}B_{\Delta_1}).$$

Then solve the R, G, B for each pixel as follow: (take R channel for example)

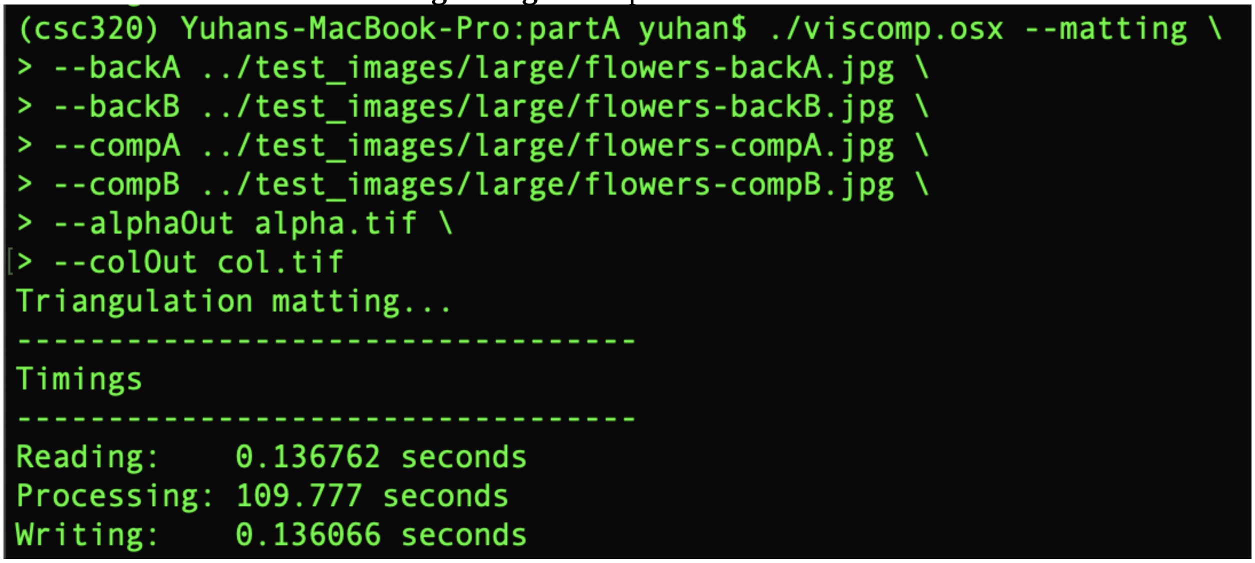


$$2R_o - \alpha_o(R_{k_1}+R_{k_2}) = R_1 - R_{k_1} + R_2 - R_{k_2}$$

$$\Rightarrow R_o = \frac{R_1-(1-\alpha_o)R_{k_1}+R_2-(1-\alpha_o)R_{k_2}}{2}$$

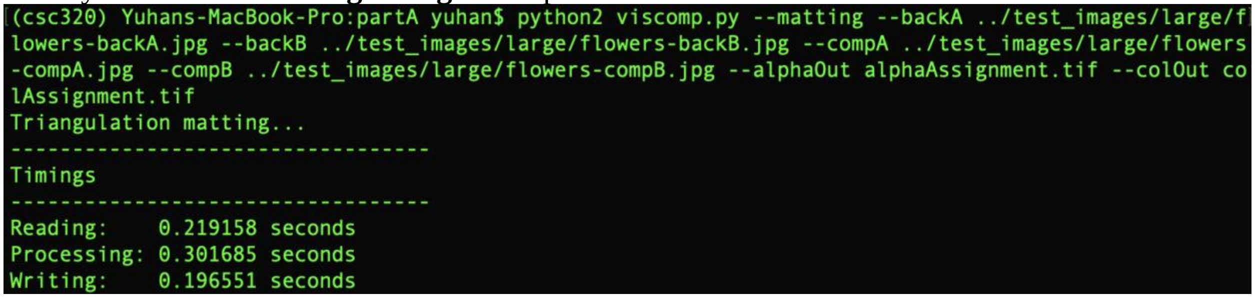
$$\Rightarrow R_o = \frac{CompA_{R_o}-(1-\alpha_o)backA_{R_o}+CompB_{R_o}-(1-\alpha_o)backB_{R_o}}{2}$$

- Reference solution for the **large image** example:



```
(csc320) Yuhans-MacBook-Pro:partA yuhan$ ./viscomp.osx --matting \
> --backA ../test_images/large/flowers-backA.jpg \
> --backB ../test_images/large/flowers-backB.jpg \
> --compA ../test_images/large/flowers-compA.jpg \
> --compB ../test_images/large/flowers-compB.jpg \
> --alphaOut alpha.tif \
[> --colOut col.tif
Triangulation matting...
----------------------------------
Timings
----------------------------------
Reading:    0.136762 seconds
Processing: 109.777 seconds
Writing:    0.136066 seconds
```

- My solution for the **large image** example:



```
[(csc320) Yuhans-MacBook-Pro:partA yuhan$ python2 viscomp.py --matting --backA ../test_images/large/f
lowers-backA.jpg --backB ../test_images/large/flowers-backB.jpg --compA ../test_images/large/flowers
-compA.jpg --compB ../test_images/large/flowers-compB.jpg --alphaOut alphaAssignment.tif --colOut co
lAssignment.tif
Triangulation matting...
----------------------------------
Timings
----------------------------------
Reading:    0.219158 seconds
Processing: 0.301685 seconds
Writing:    0.196551 seconds
```

Explanation for the optimization:
The original version took a long processing time, this probably because it uses the for loop iterates on each pixel in the image (there are many be thousands of pixels in an image) and calculates the alphaOut and colOut individually. Therefore, the original version of the algorithm is too slow. However, in the optimized one, numpy matrix calculation allows me to calculate the whole image for each channel at once, this change accelerates the processing time in the new Matting algorithm.

■ **Part 3: Experimental Evaluation & Report:**

(1) The procedure(s) I used to capture my images:

To capture the different set of images, I use the following manual camera App:
This App allows me to adjust and fix the focus, shutter speed and ISO manually.

Yamera (Manual Camera)

Manual Focus, Exposure and WB.

OPEN

4.6 ★★★★☆

5 Ratings

4+

Age

For each setting, I processed use the following step:

- First, set up the focus, shutter speed and ISO values in the App manually and fix the value for each setting.
- Secondly, take the Setting1 with regular black and white background for example:

➔ Capture the white background (computer mainframe box in BA2240 with a white A4 paper attached to it), as **background1**.

➔ Then, put the target object in front of the captured white background1, and took a picture, this is **comp1**.

➔ Then, **keep the camera and object stay in the same place (also with the same environment light),** and removed the attached white paper, and took a picture, this is **comp2**. (comp2 with the pure black background). **[*]**

➔ Finally, moved away the target front object, and took a picture, this is **backgound2**.

- Also note that all the images should have the same size, including the newbackground in the createComposite procedure.

(2) The range of imaging conditions I tried:

- **Setting1 (best):** regular black and white backgrounds with simple paper cup (object without transparent and mental reflective).
- **Setting2 (good):** yellow and blue backgrounds with transparent object (transparent water bottle).
- **Setting3 (so-so):** regular black and white background with mental reflective object (can).
- **Setting4 (not bad):** High ISO and with black and white background where black background has a little white 'bottom boundary'.
- **Setting5 (bad):** similar background with slightly different changes.
- **Setting6 (bad):** complex background (lots of letter noise) with mental reflective object. (lipstick)

(3) The assessment of the technique's limits:

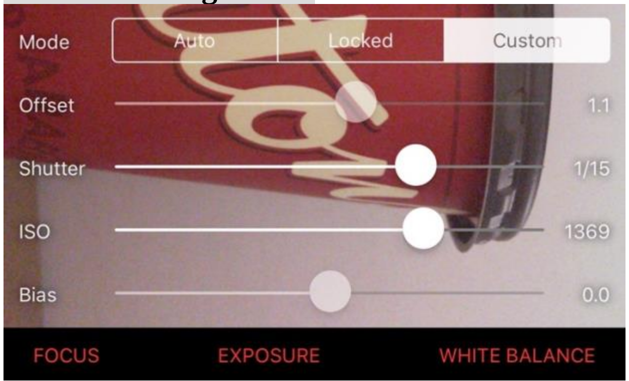
As I mentioned in the **[*]** above, it is important to "**keep the camera and object stay in the same place**", it is easy to keep the object in the same place for each capture, but since I did not have a tripod to fix the position of the camera, it is hard to keep the camera still. Therefore, in some alphaOut.tif and colOut.tif, there is a "blur" around the front object.

(4) A set of figures showing experimental results

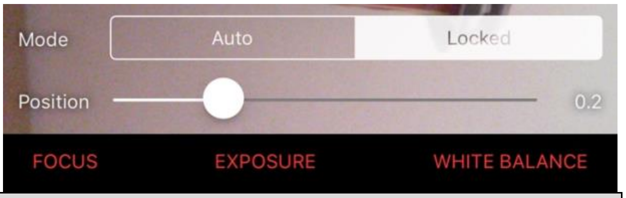
Input for trianglationMatting: background1, background2, comp1, comp2
Output from trianglationMatting: alphaOut, colOut
Input for createComposite: alphaOut, colOut, newbackground
Output from createComposite: compOut

- **Setting 1:**

Camera Settings Data:

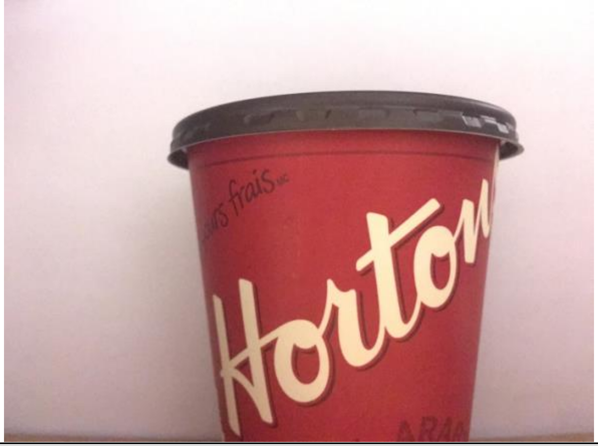
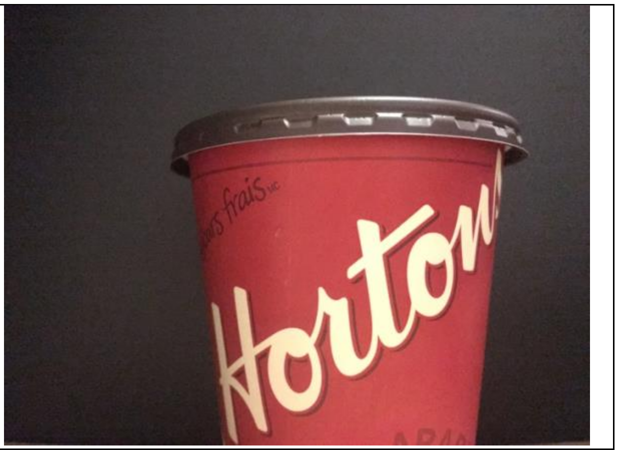
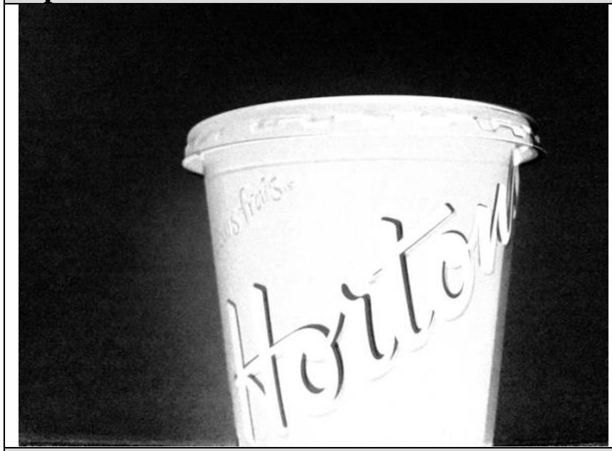
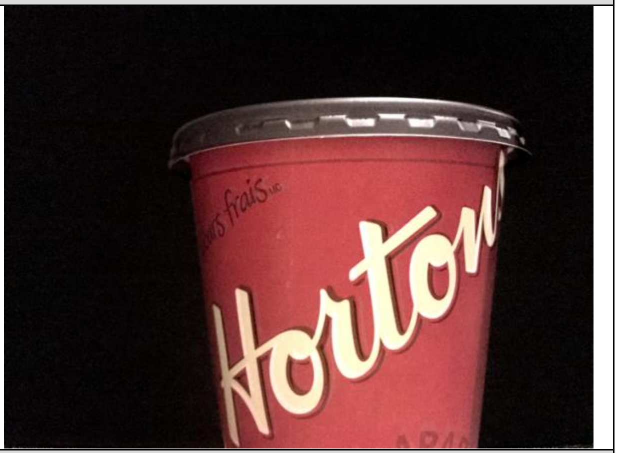
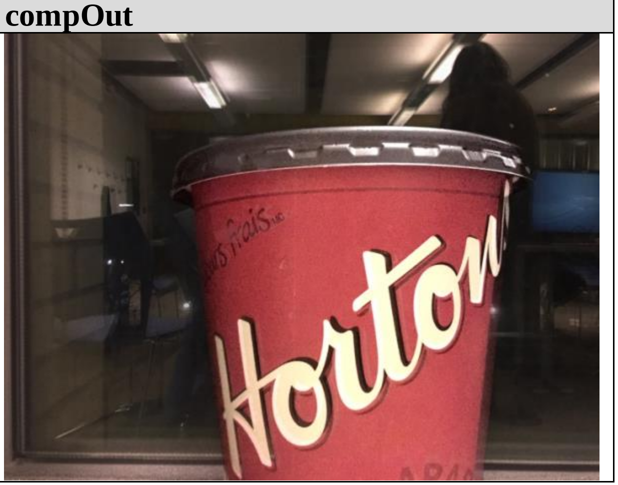


Mode: Auto | Locked | Custom
Offset 1.1
Shutter 1/15
ISO 1369
Bias 0.0



Mode: Auto | Locked
Position 0.2

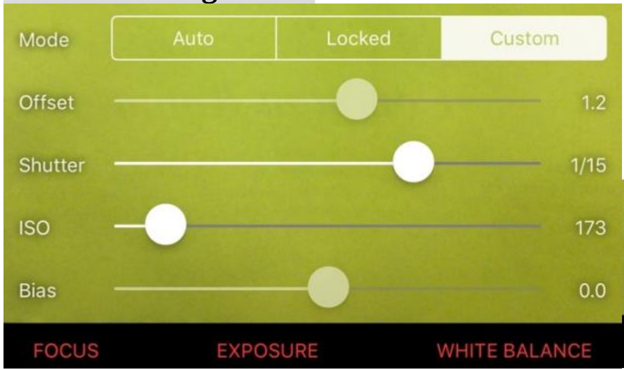
background1	**background2**
comp1	**comp2**

	
alphaOut	**colOut**
	
newbackground	**compOut**
	

Analysis & Conclusion:
This setting with the regular black and white backgrounds works well on both triangulaitonMatting and createComposite algorithm. Except as mentioned in "**The assessment of the technique's limits**" part, there is a slight "blur" boundary around the front objects. This setting is considered to be a perfect experimental result.

- **Setting 2:**

Camera Settings Data:



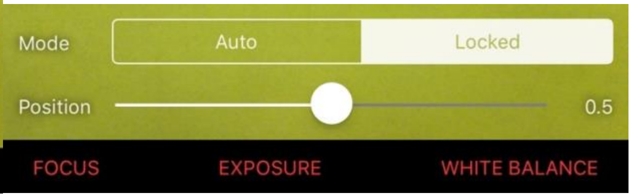
Mode: Auto | Locked | [✓ Custom]

Offset 1.2

Shutter 1/15

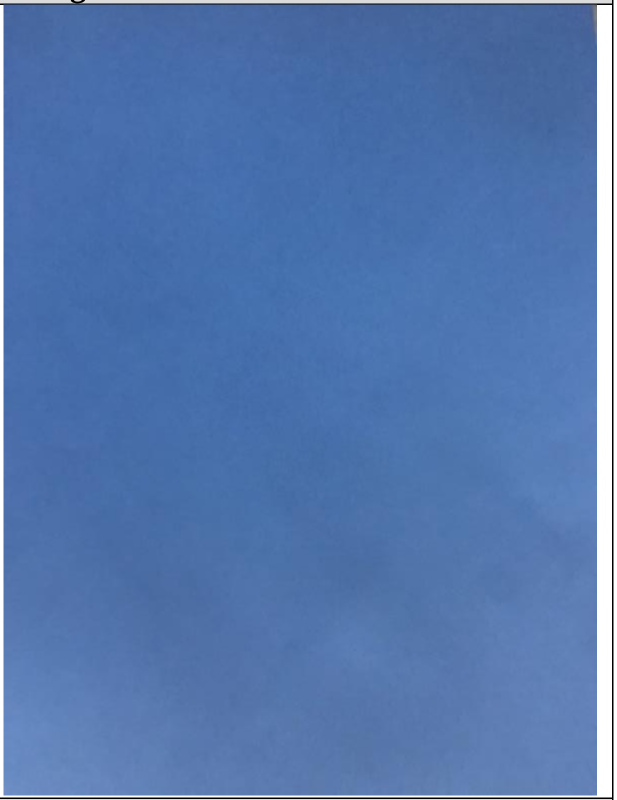
ISO 173

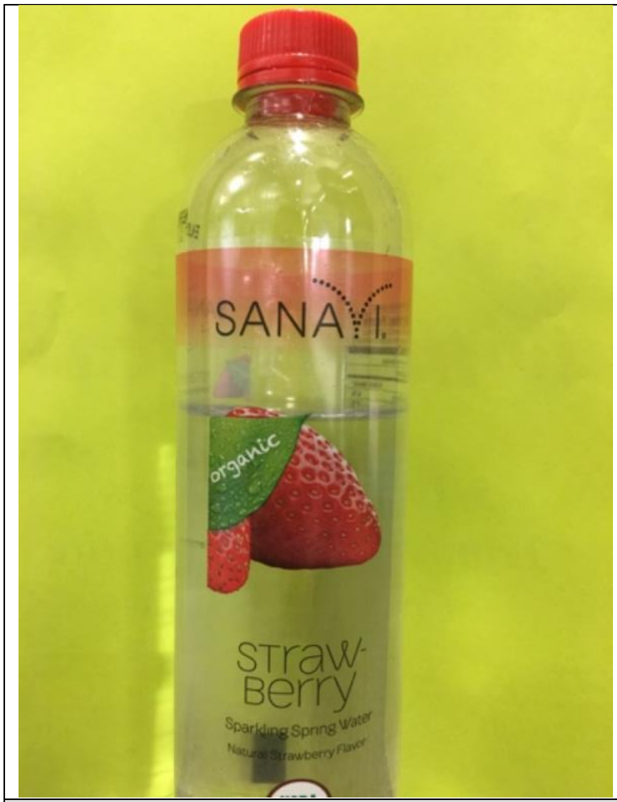
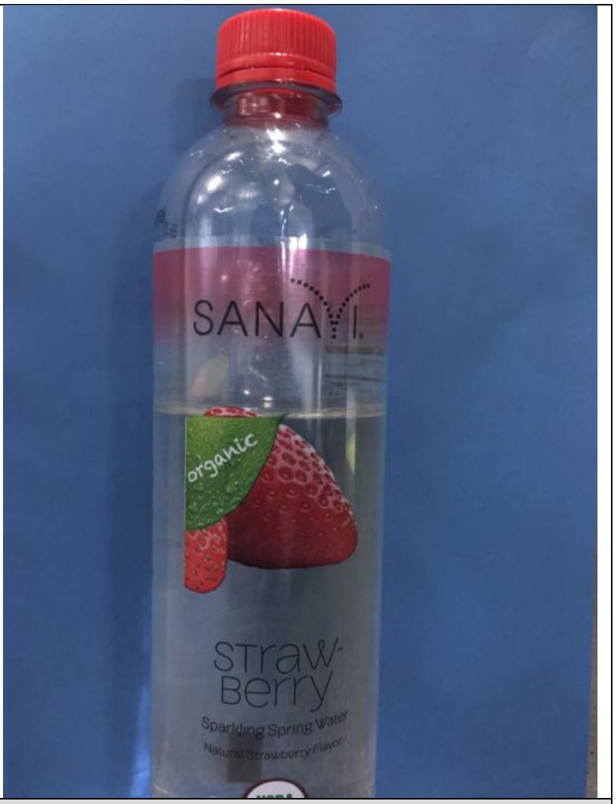
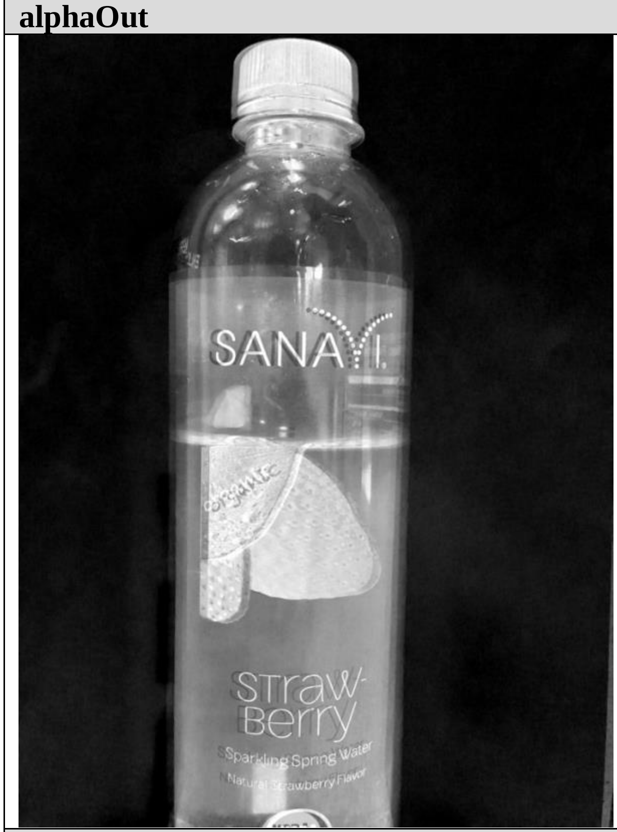
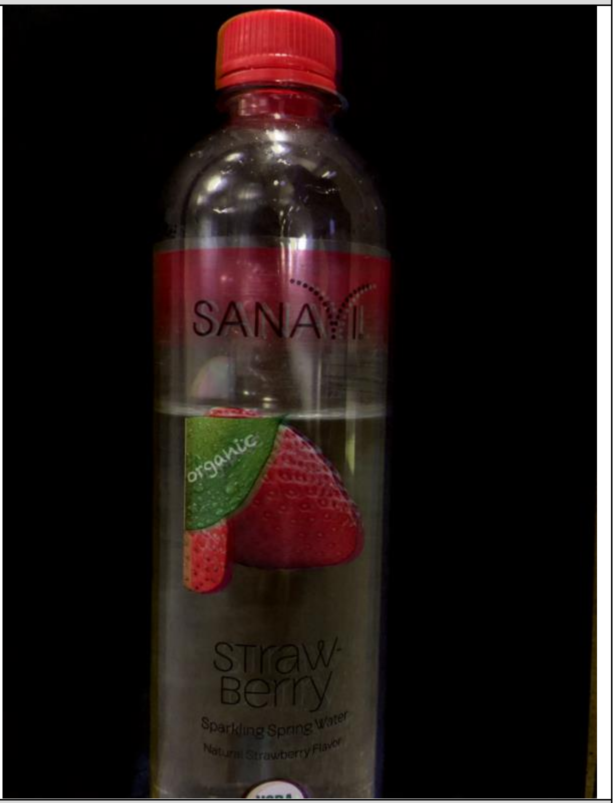
Bias 0.0

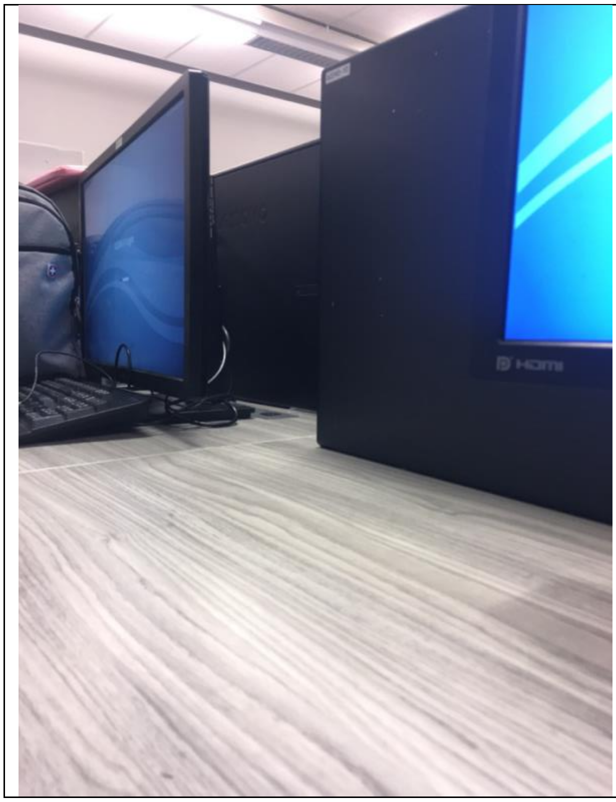


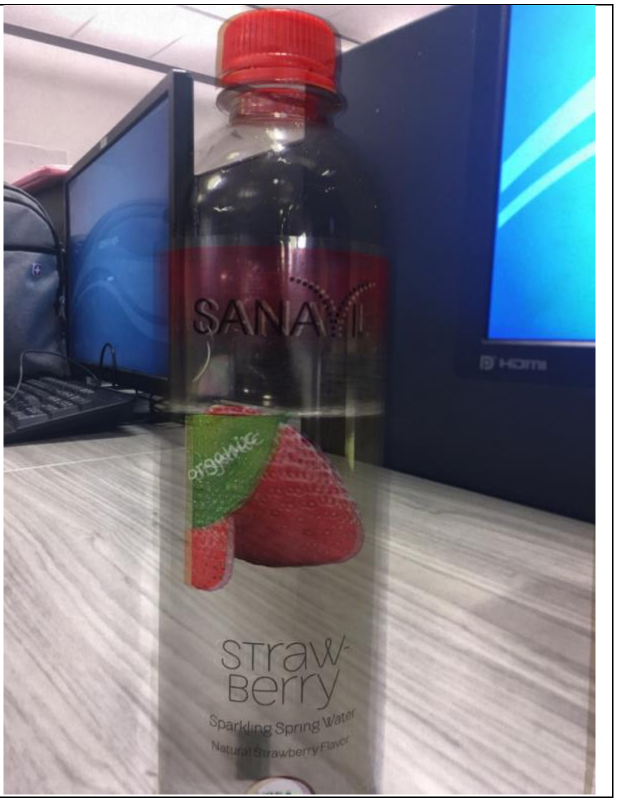
Mode: Auto | [✓ Locked]

Position 0.5

background1	**background2**
	
comp1	**comp2**

<table>
<tr>
<td>
</td>
<td>
</td>
</tr>
<tr>
<td><b>alphaOut</b></td>
<td><b>colOut</b></td>
</tr>
<tr>
<td>
</td>
<td>
</td>
</tr>
<tr>
<td><b>newbackground</b></td>
<td><b>compOut</b></td>
</tr>
</table>

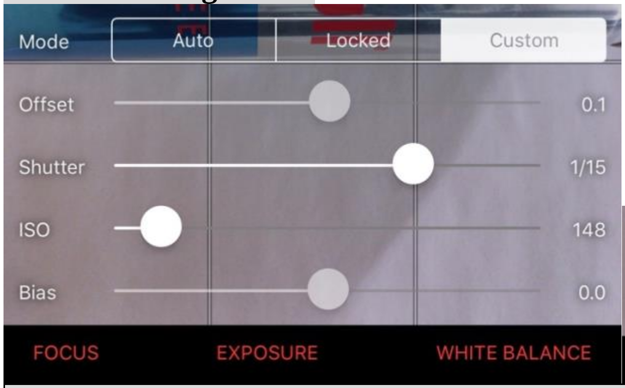




Analysis & Conclusion:
This setting has the same problem with the “blur” boundary around the object. However, the compOut image is not as that prefect as the composition in the first setting. It is a little darker than what we expected. But overall, it is a good experimental result but not as perfect as the result from the setting1.

- **Setting 3:**

Camera Settings Data:



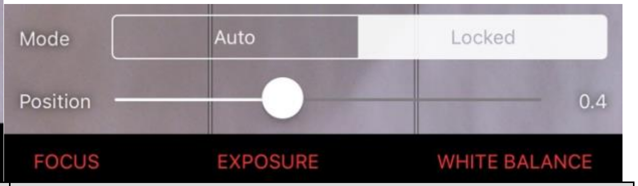
Mode: Auto | Locked | Custom

Offset 0.1

Shutter 1/15

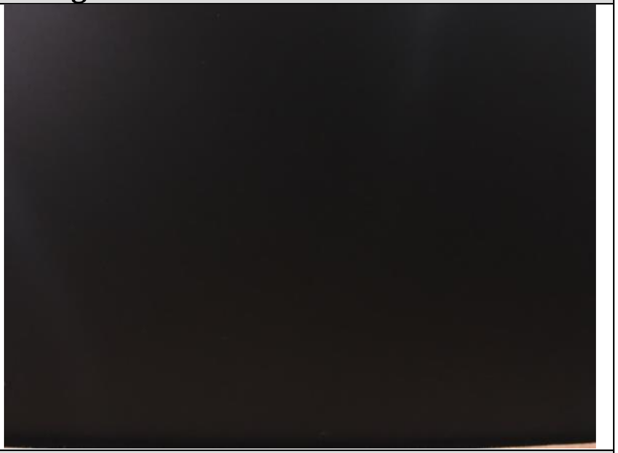
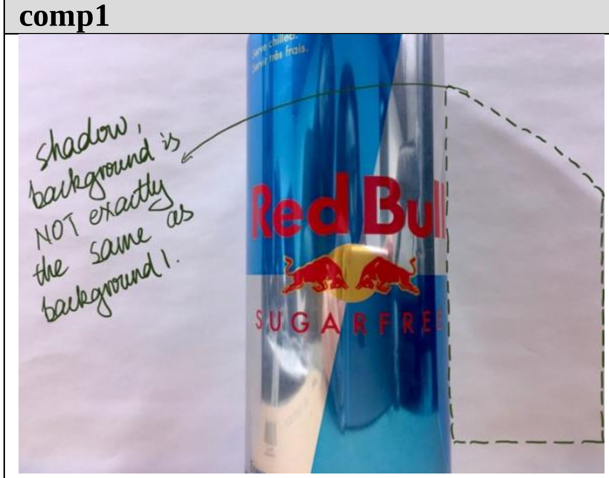
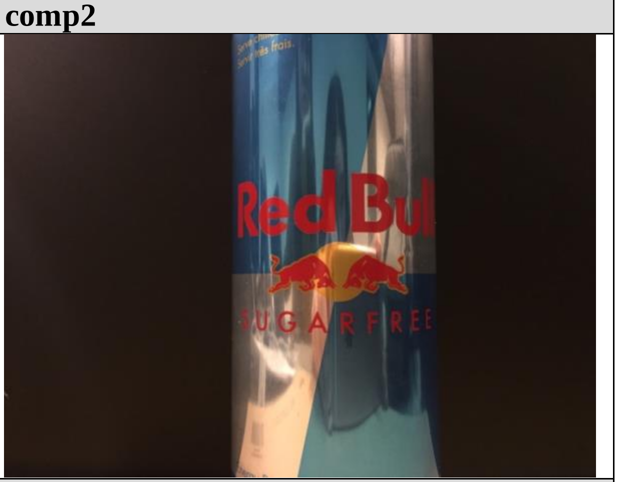
ISO 148

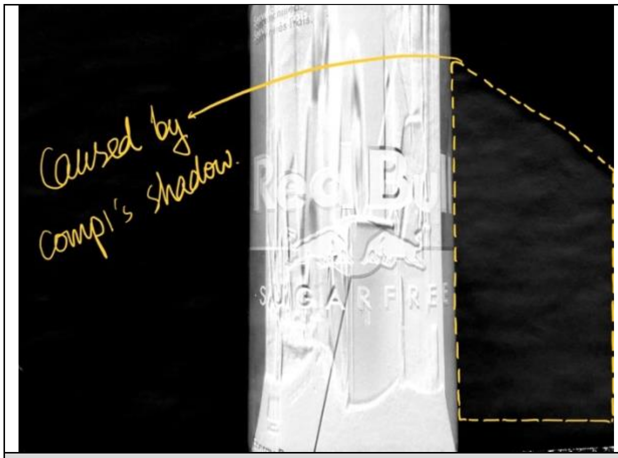
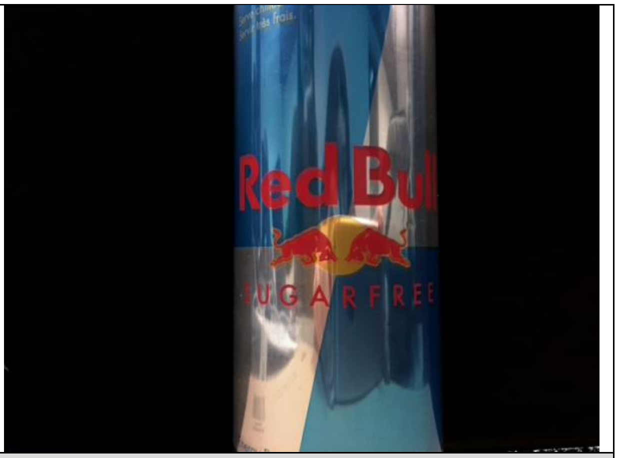
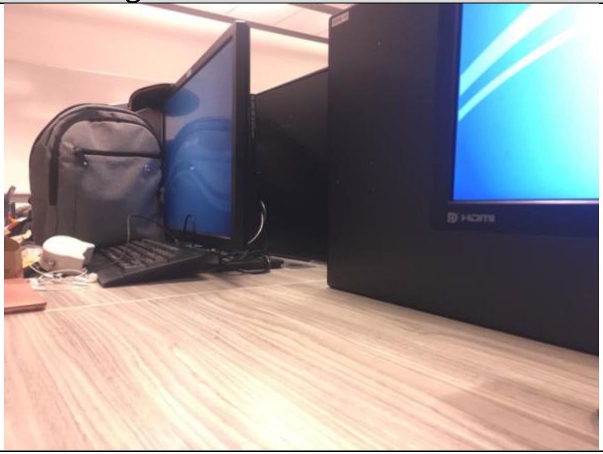
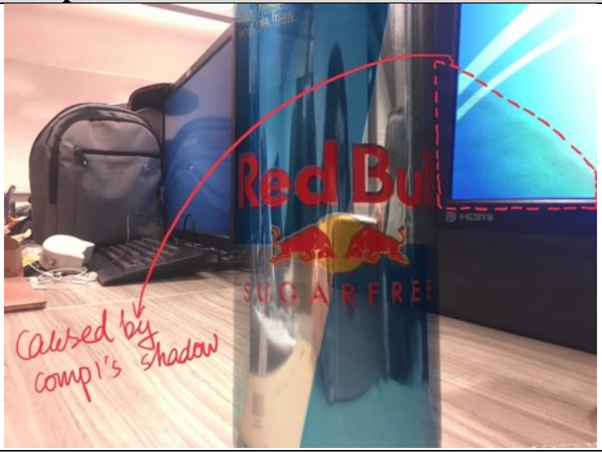
Bias 0.0



Mode: Auto | Locked

Position 0.4

background1	**background2**
	
comp1	**comp2**
 shadow, background is NOT exactly the same as background1.	
alphaOut	**colOut**

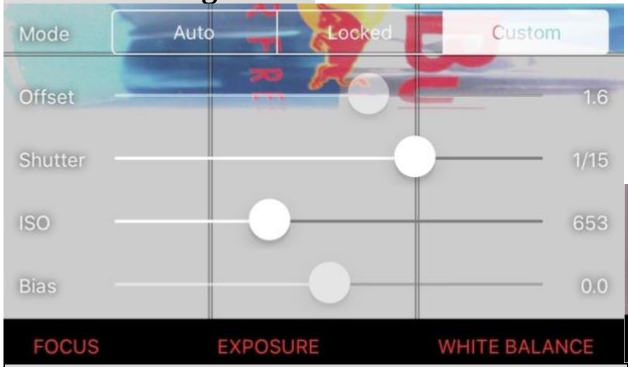
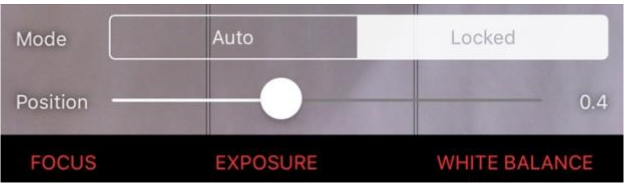
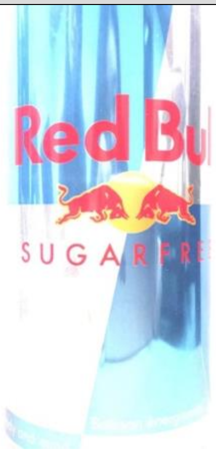
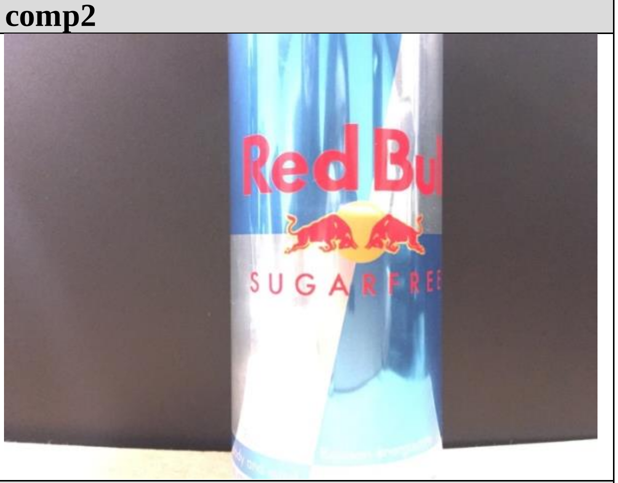
	
newbackground	**compOut**
	

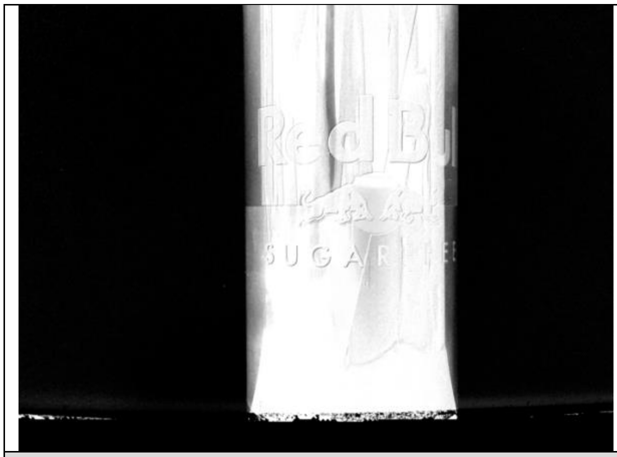
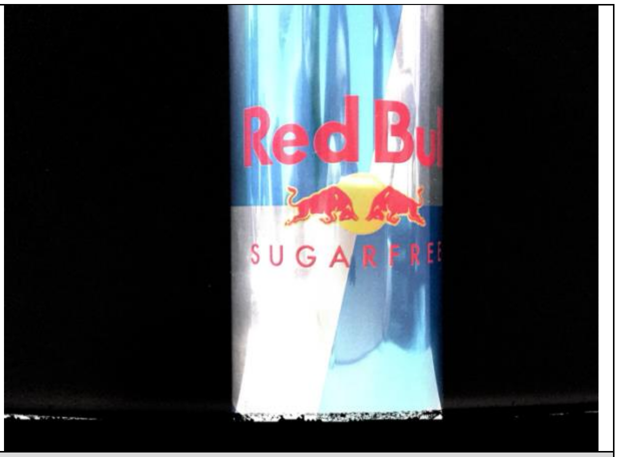
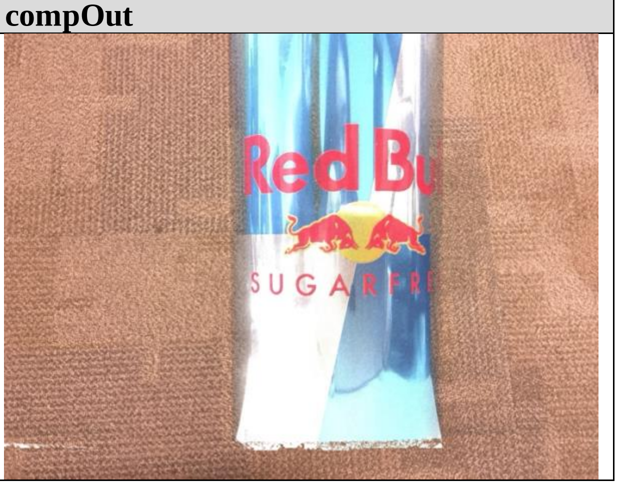
Analysis & Conclusion:

In the comp1, we can see that there is a “shadow” at the right of the object. The pixels in the “shadow” area are different from the pixel intensities from that area in background1. And the pixel intensities in that area are happened to be similar to the background2. Therefore, in alphaOut.tif, we can see that the pixels in the “shadow” are slightly brighter than the other background pixels. This “shadow” effect also reflects on the compOut, we can see that there is a rectangle “shadow” on the bottom left corner of the screen. Overall, the experimental result is not that bad. (this case is similar to the Part 4: Written Question)

- **Setting 4:**

Camera Settings Data:

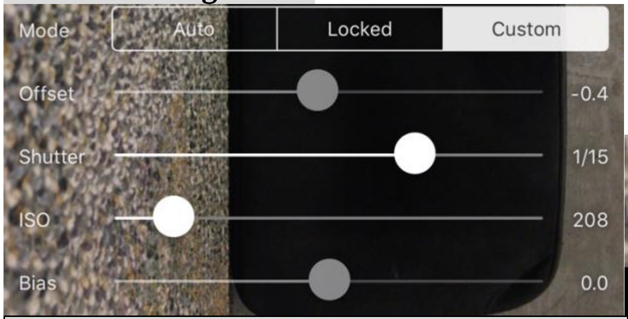
 Mode Auto Locked Custom Offset 1.6 Shutter 1/15 ISO 653 Bias 0.0 FOCUS EXPOSURE WHITE BALANCE	 Mode Auto Locked Position 0.4 FOCUS EXPOSURE WHITE BALANCE
background1	**background2**
comp1	**comp2**
 	
alphaOut	**colOut**

	
newbackground	**compOut**
	

Analysis & Conclusion:
In this high ISO setting, we can see that the bottom of the “Red Bull” can is overexposure, and the bottom boundary of both two backgrounds are happened to be white, therefore, the bottom part of the “Red Bull” can is considered to be the background rather than the front object. Hence, in the compOut image, we can see that the bottom part of the “Red Bull” can disappear. This experimental result is bad on the bottom part of the overexposure “Red Bull” can.

- **Setting 5:**

Camera Settings Data:



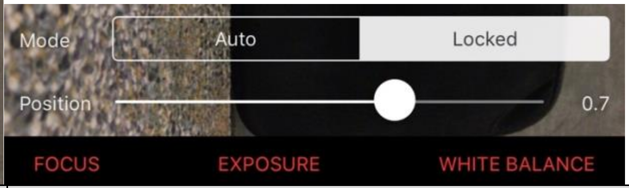
Mode Auto Locked Custom

Offset -0.4

Shutter 1/15

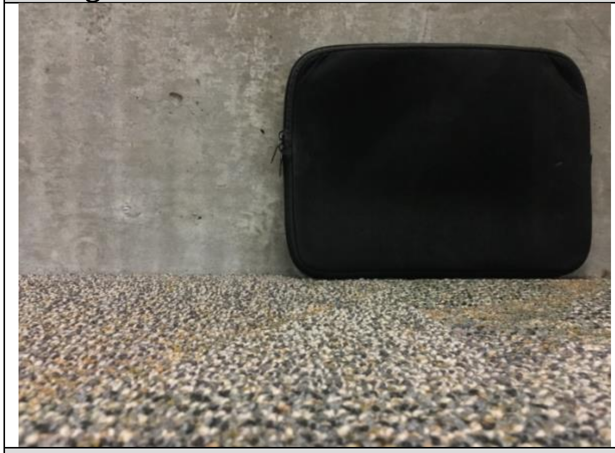
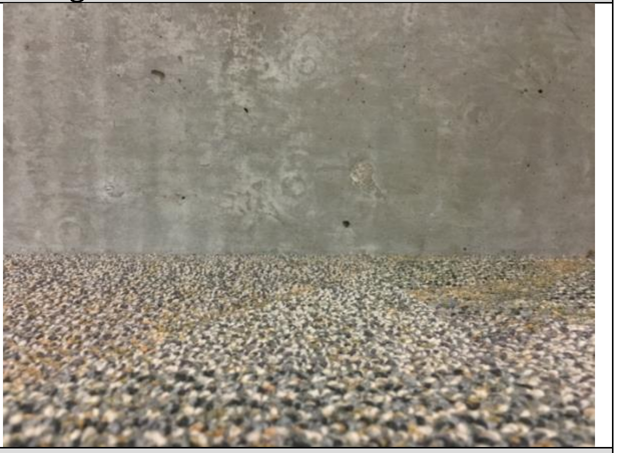
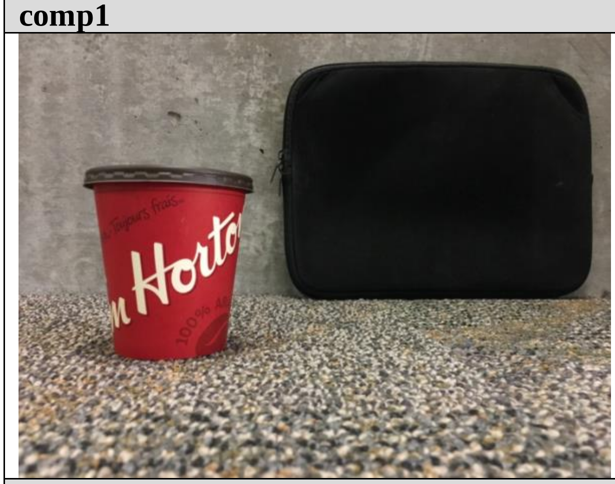
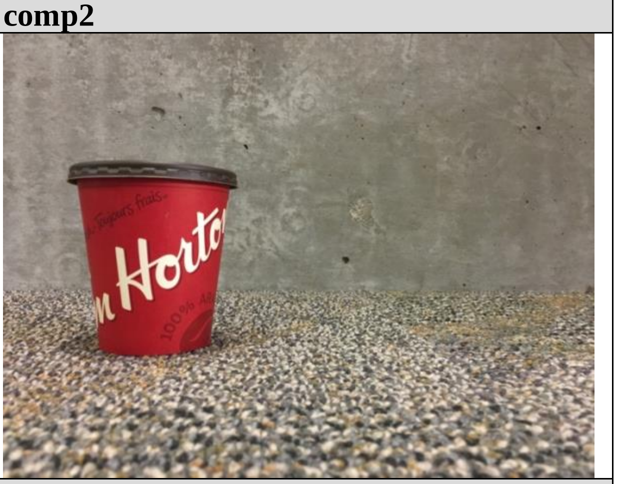
ISO 208

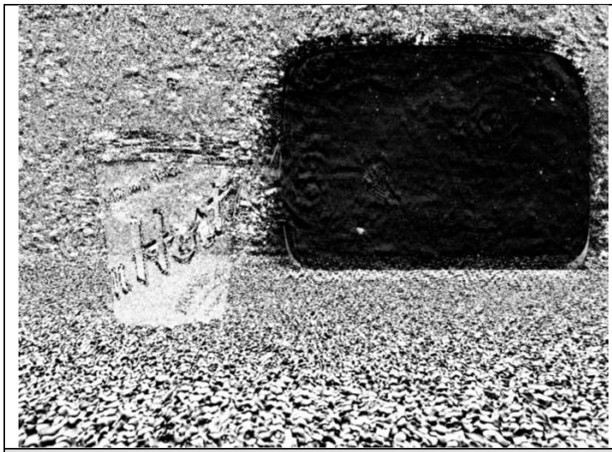
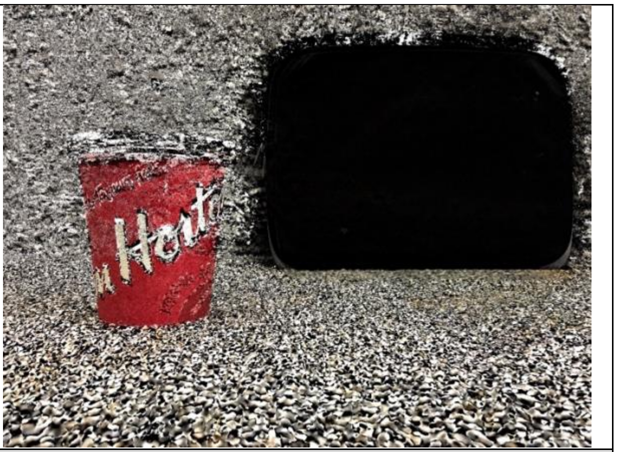
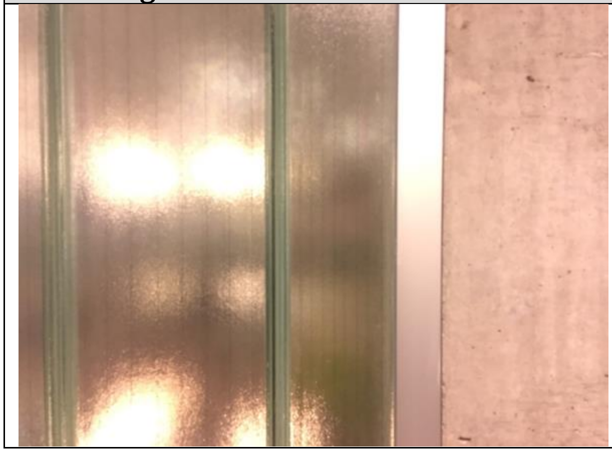
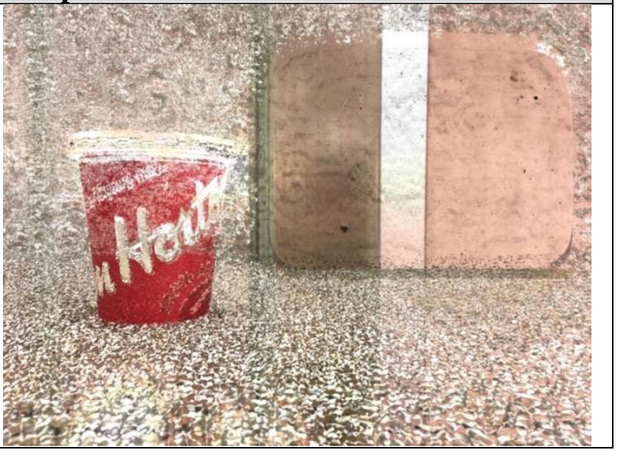
Bias 0.0



Mode Auto Locked

Position 0.7

background1	background2
	
comp1	**comp2**
	
alphaOut	**colOut**

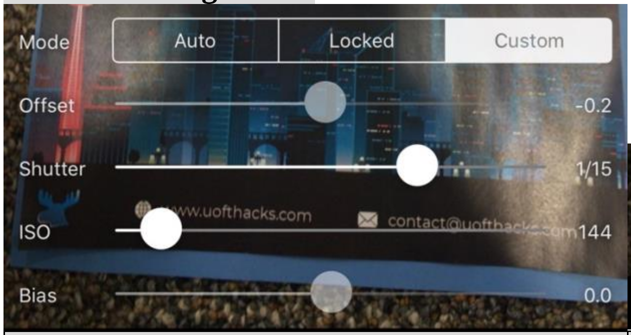
	
newbackground	**compOut**
	

Analysis & Conclusion:

The experimental result of this setting is **really bad**. Since the **two backgrounds are too similar**, everything excepts the laptop bag (the only different between two background) are identified as the front object. This because Triangle Matting algorithm considers the pixels (in the same position in two comp image) with the similar intensity to be the front object. Therefore, only the black laptop bag is considered as the background.

- **Setting 6:**

Camera Settings Data:



Mode: Auto | Locked | Custom

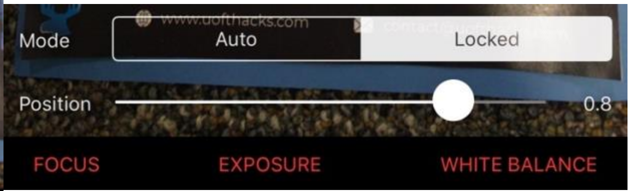
Offset: -0.2

Shutter: 1/15

ISO: 144

Bias: 0.0

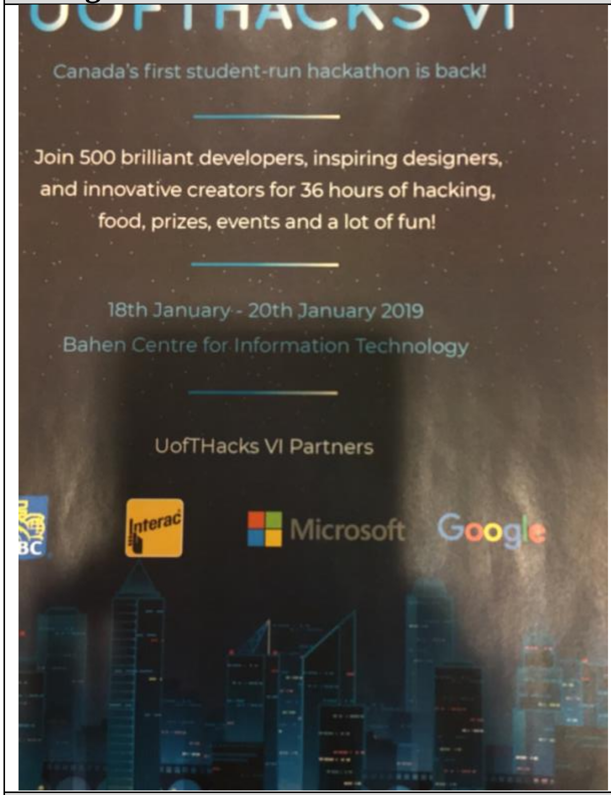
background1



Mode: Auto | Locked

Position: 0.8

background2



UOFTHACKS VI

Canada's first student-run hackathon is back!

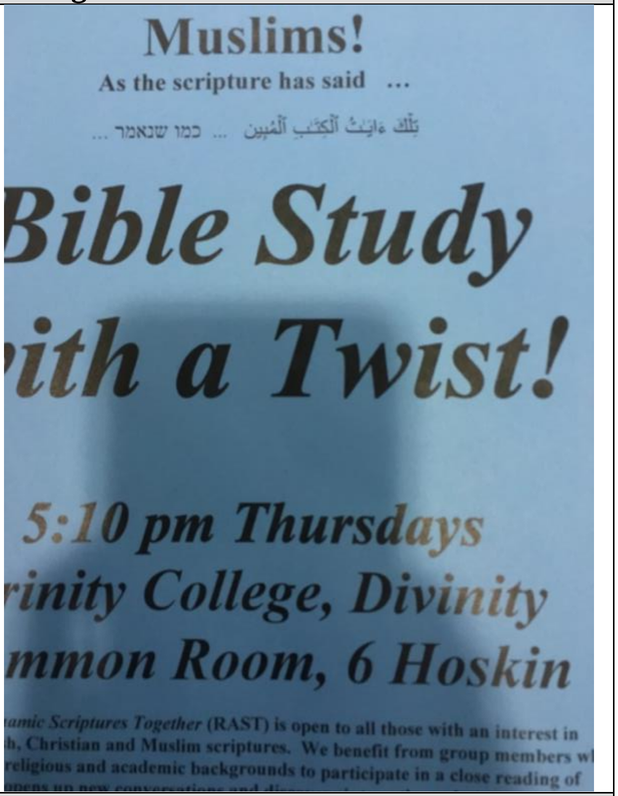
Join 500 brilliant developers, inspiring designers, and innovative creators for 36 hours of hacking, food, prizes, events and a lot of fun!

18th January - 20th January 2019

Bahen Centre for Information Technology

UofTHacks VI Partners

comp1



Muslims!

As the scripture has said ...

... כמו שנאמר ... تِلْكَ ءَايَٰتُ ٱلْكِتَٰبِ ٱلْمُبِينِ

Bible Study

ith a Twist!

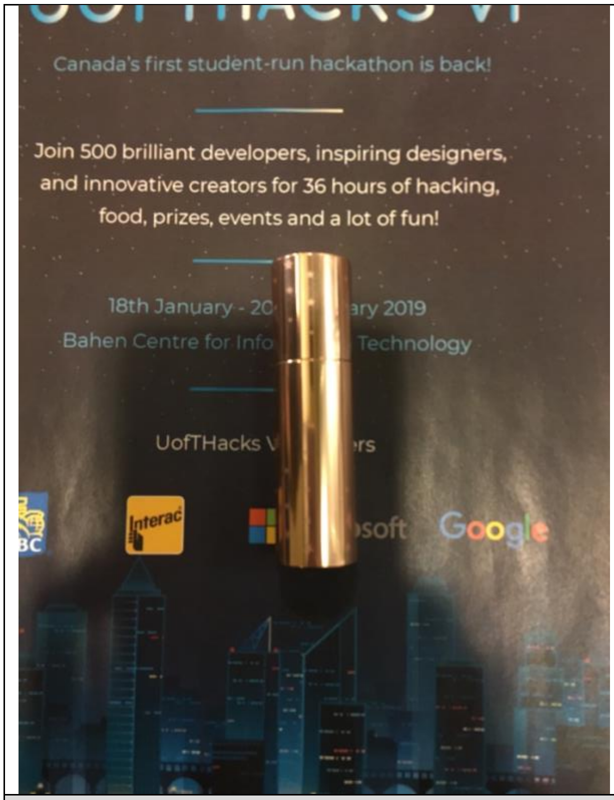
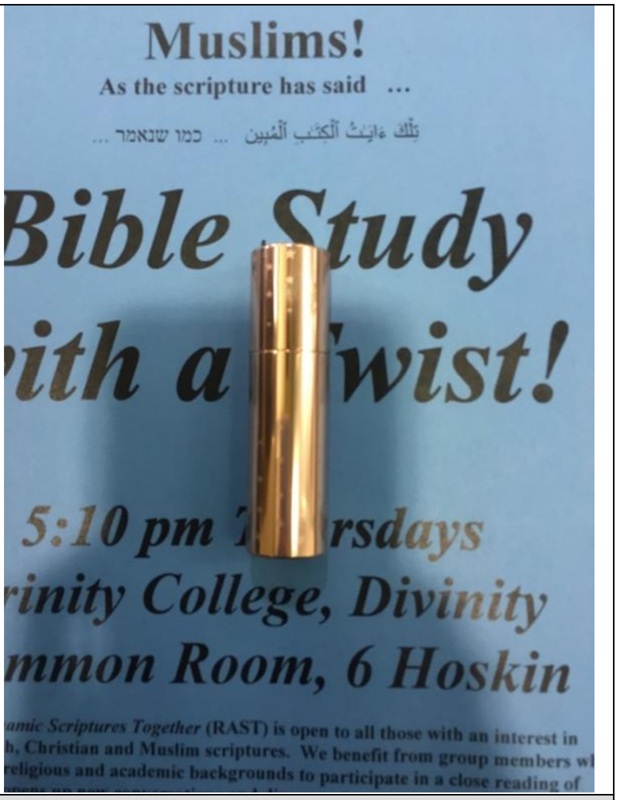
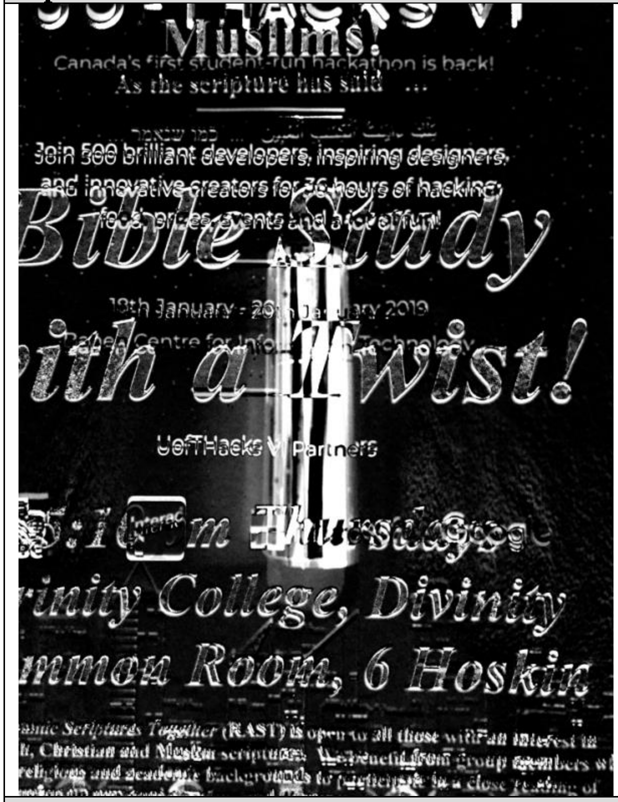
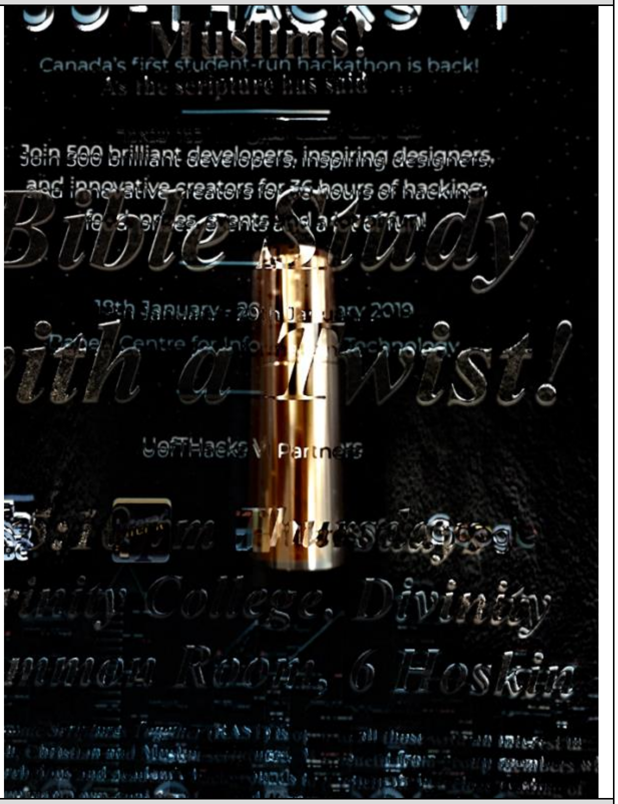
5:10 pm Thursdays

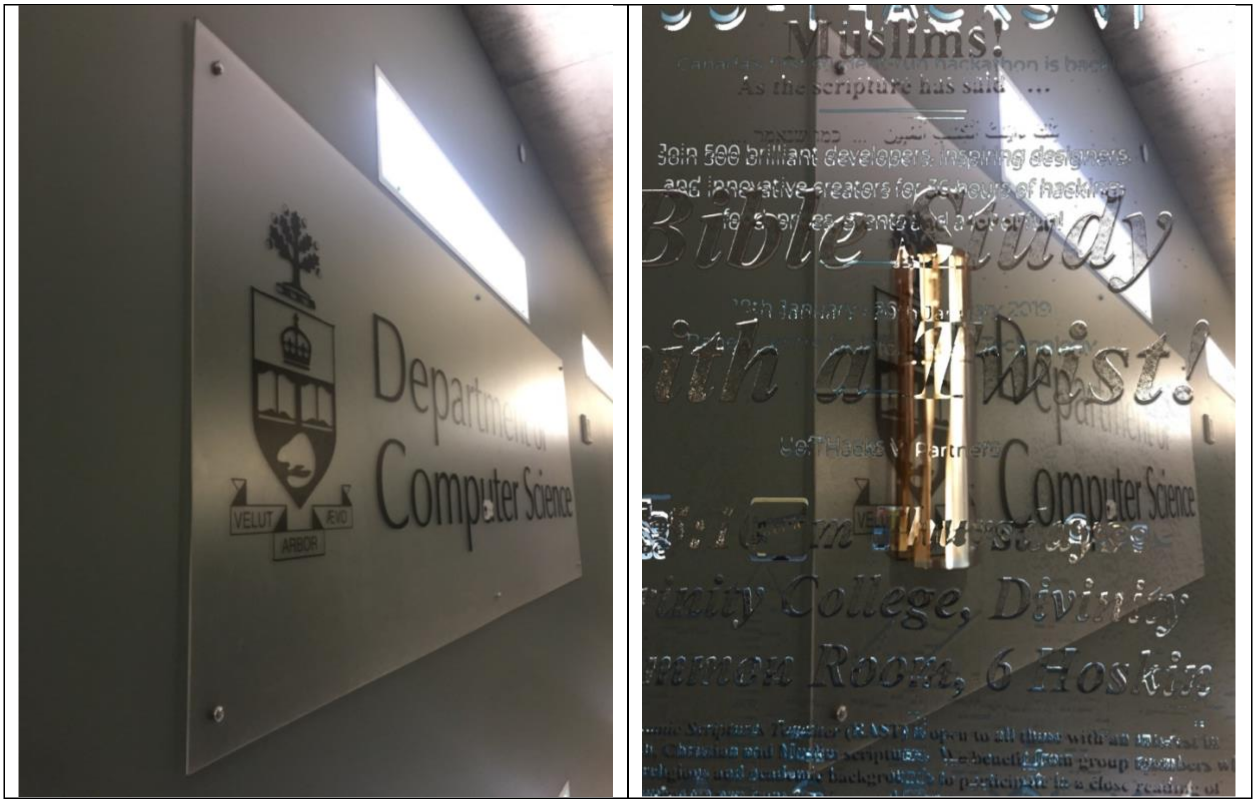
rinity College, Divinity

mmon Room, 6 Hoskin

amic Scriptures Together (RAST) is open to all those with an interest in h, Christian and Muslim scriptures. We benefit from group members wi religious and academic backgrounds to participate in a close reading of opens up new conversations and di

comp2

 Canada's first student-run hackathon is back! Join 500 brilliant developers, inspiring designers, and innovative creators for 36 hours of hacking, food, prizes, events and a lot of fun! 18th January - 20 ary 2019 Bahen Centre for Info Technology UofTHacks V rs Interac soft Google	 Muslims! As the scripture has said … Bible Study ith a wist! 5:10 pm rsdays rinity College, Divinity mmon Room, 6 Hoskin
alphaOut	**colOut**
	
newbackground	**compOut**


Department of
Computer Science
VELUT
AEVO
ARBOR
Muslims!
As the scripture has said ...
Join 500 brilliant developers, inspiring designers
Bible Study
Computer Science

Analysis & Conclusion:
The experimental result of this setting is also bad. This is because the two background are too complex, i.e.: has too many noises (letters). And it is hard for the algorithm to differentiate the front object from the background.

- **Overall Conclusion:**

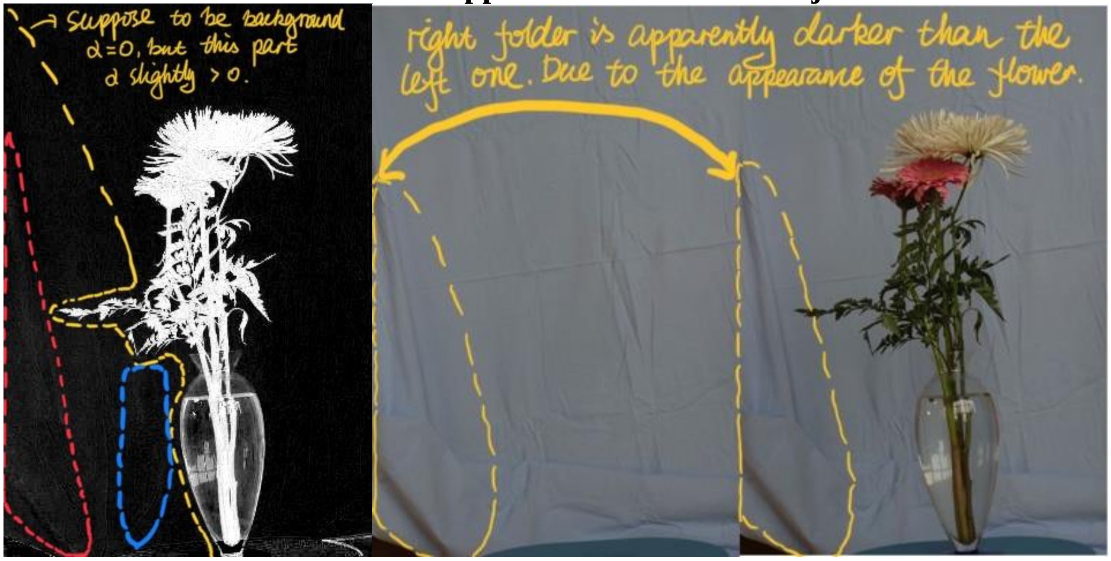
In order to make Triangulation Matting algorithm works well, we need:

1. The two backgrounds need to satisfy:
   Make all pixels of the background as different as possible, otherwise may have the situations happened in the Setting5 and Setting6. Therefore, it is better to use simple, pure different color backgrounds. On the other hand, we need to make all pixels of the background different from the front object.
2. The front object is better not overexposure, otherwise may counter the situation happened in the Setting4. Also, we should pay attention to the lighting in the environment, in order to avoid the shadows when we put the object in front of the background, otherwise may have the situation happened in the Setting3.
3. For each capture, it is better to keep the same lighting environment and camera settings. When we take a picture of compA and compB, it is better to keep the front object in the same position and use the tripod to fix the position of the camera. Otherwise, may have the situations happened in the Setting1 and Setting2, which has a “blur boundary” around the front object.

■ Part 4: Written Question:

To answer the question why the alpha matte on the right side of the vase has zero or near-zero intensities whereas on its left side the intensities are low but not zero, there are the following two aspects to consider:

1. Shadows that come from the appearance of the front object.


suppose to be background
α=0, but this part
α slightly > 0.
right folder is apparently darker than the left one. Due to the appearance of the flower.

When we put the object in front of the background, the object may block out or reflect some lights. Therefore, compare the two images, we can see that the background on the right is slightly darker than the left one. According to the alpha equation:

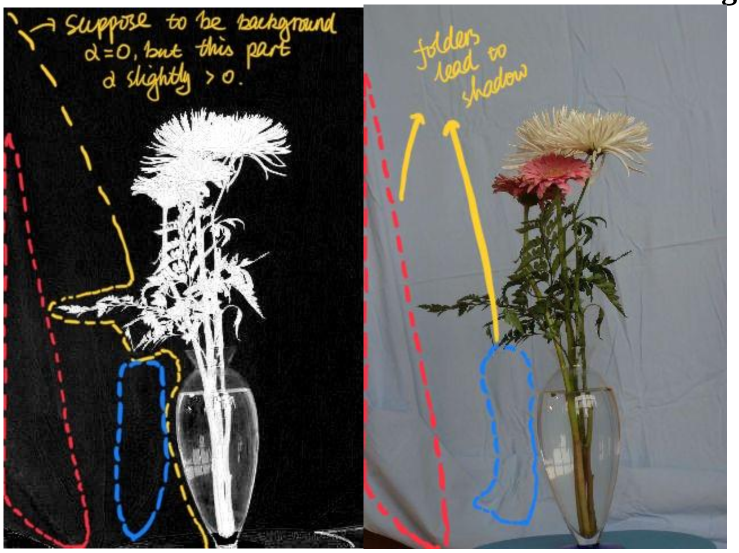
$$\alpha = 1 - \frac{(R_{f1} - R_{f2})(R_{k1} - R_{k2}) + (G_{f1} - G_{f2})(G_{k1} - G_{k2}) + (B_{f1} - B_{f2})(B_{k1} - B_{k2})}{(R_{k1} - R_{k2})^2 + (G_{k1} - G_{k2})^2 + (B_{k1} - B_{k2})^2}$$

In the above situation (i.e.: compA and backA are slightly different), we have:

$$R_{f1} - R_{f2} \neq R_{k1} - R_{k2},\ G_{f1} - G_{f2} \neq G_{k1} - G_{k2},\ B_{f1} - B_{f2} \neq B_{k1} - B_{k2}$$

i.e.: $\alpha \neq 0$, which means that "background" pixel would not be transparent in the alphaOut image, therefore, the left side intensities are low but not zero.

2. Shadows that comes from the folders on the background.



The folders on the compA/backA (white background) have the dark shadows, and the backB (black background) is happened to be back, therefore, the intensities of the "shadow" pixels on the compA are similar to the correspond pixels in backB. According to the alpha equation:

$$\alpha = 1 - \frac{(R_{f1} - R_{f2})(R_{k1} - R_{k2}) + (G_{f1} - G_{f2})(G_{k1} - G_{k2}) + (B_{f1} - B_{f2})(B_{k1} - B_{k2})}{(R_{k1} - R_{k2})^2 + (G_{k1} - G_{k2})^2 + (B_{k1} - B_{k2})^2}$$

In the above situation, we consider the alpha value of the "shadow" pixel:

$(R_{f1} - R_{f2} = 0 \parallel R_{k1} - R_{k2} = 0)$ &&
$(G_{f1} - G_{f2} = 0 \parallel G_{k1} - G_{k2} = 0)$ &&
$(B_{f1} - B_{f2} = 0 \parallel B_{k1} - B_{k2} = 0)$

Therefore, $\alpha \neq 0$, which means that "background" pixel would not be transparent in the alphaOut image, therefore, the left side intensities are low but not zero.

■ Part 5: Bonus Component: Living Matting:

Technique derived from the following on-line resource:

McGuire, Morgan, and Wojciech Matusik. "Real-Time Triangulation Matting Using Passive Polarization". *Vcg.Seas.Harvard.Edu*, 2019, https://bit.ly/2DAIVI6

In this paper, it introduces a new technique which operates in real-time on video and this technique does not impact the illumination color of the scene. This new method used a gray, polarized screen, and film the scene with a single camera containing two differently-polarized sensor, which shown as the figure below:

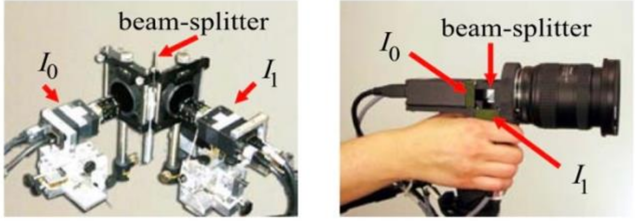


Figure 1: *Left*) Prototype camera that splits on polarization. *Right*) Hand-held form factor with the beam-splitter behind of the lens.

We can use this camera simultaneously captures an image of the actor against an apparently black and gray backgrounds. Then, with these two images, we can triangulate the foreground color and matte. In this paper, it also mentions another method which based on a polarized chroma-key algorithm, but the algorithm mentions in this paper gives better sub-pixel results and the screen works with existing lighting infrastructure.