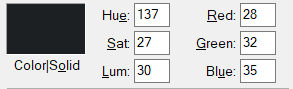
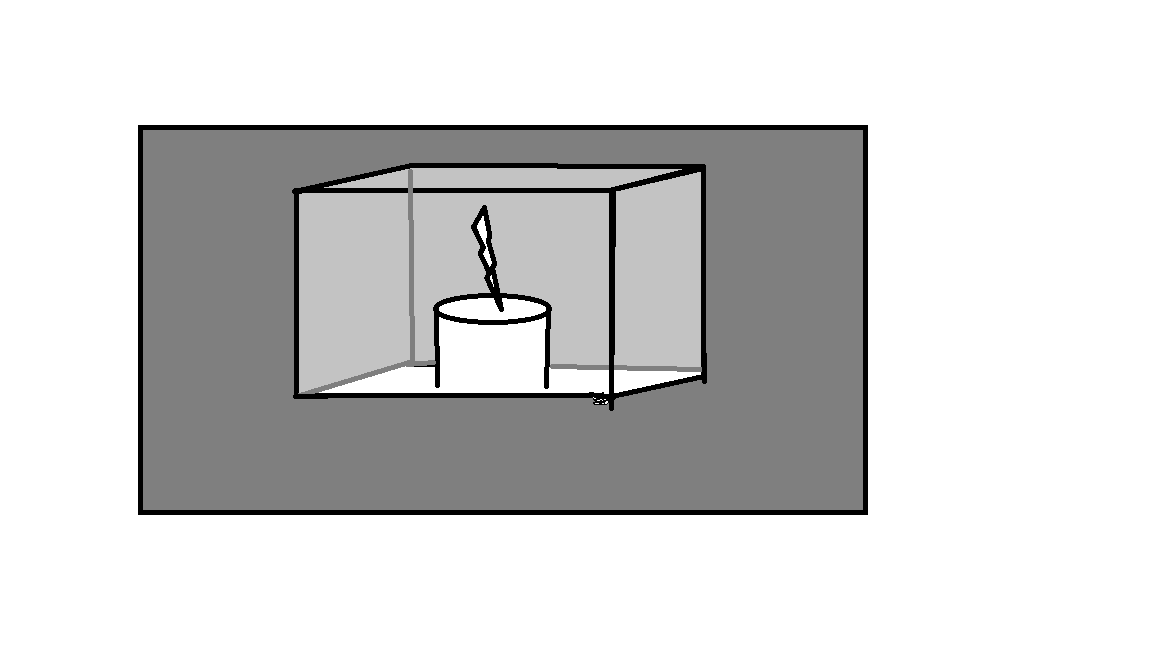
Part 3:

3.1:

 The pixels on the left side of the object have non-zero alpha because the algorithm thinks that there are foreground objects that exist there. The alpha picture above has its brightness upped by 50% to highlight the non-zero alpha to the left of the object. The non-zero alpha is here because in the original picture with the object and the lighter background, there are wrinkles and creases in the background that are picked up by the algorithm are not in the left side of the other darker background. From the result of the experiments we did with the given implementation of the algorithm, when there are significant objects in the background, like shiny logo(experiment A), or heavy creases in cloth (experiment B), they will be picked up by the algorithm despite also being in the pure background picture. This is because despite being the same background, the background in the composite picture with the object is slightly darker than the pure background without the object. This slight difference caused the wrinkles to be determined to be foreground. The lighter background is relatively even to the top and the right of the object, so they did not get picked up as part of the foreground objects. Therefore, the non-zero alpha to the right of the object is the uneven background picked up by the algorithm to be part of the foreground. However, there are also uneven wrinkles in the darker background, to the right of the object. These did not appear in the alpha picture though. This is because the colour of the darker background is very close to pure black (colour sample pictured to the right), which is Red=0, Green=0, Blue=0. Because it is this close, it is not recognizable when put in the alpha picture. Also, this small difference also caused the slight brightness difference between the composite and pure background pictures to be a non-issue.

3.2:

C = Composite of all four layers.



F = Front face of case, foreground

O = Object, foreground

B = Back face of case, background

K = Background behind the case, background

CBK = αB \* CB + (1 - αB) \* CK

COBK = αO \*CO + (1 - αO) \*CBK

C = αF \*CF + (1 – αF) \*COBK

Part 4

One potential solution that might be able to solve the problem of live matting is with a slow-motion capturing camera and two backgrounds that can switch much faster relative to the motion of the foreground object. Although I am not sure how feasible this solution is in a real-life situation, if the two background can be switched fast enough, relatively, the foreground would seem like it is not moving. The slow-motion camera is expensive but available, and the background could be digital or physical, as long as it can be switched much faster in relative to the motion of the foreground object. Another potential solution is inspired by the research paper titled “Integrated Foreground Segmentation and Boundary Matting for Live Videos”. For this paper, the author purposed to use machine learning, to “train and maintain two competing one-class support vector machines at each pixel location, which model local color distributions for both foreground and background, respectively.” This technique can capture moving foreground objects and moving background at the same time and discriminate between the two. Thus, this technique can also be applied to a moving foreground and two static background. In fact, I believe that this technique is probably overkill for our situation. The use of machine learning on the moving object with two different background can probably also solve our live matting problem, with enough training.

References:

Gong, M., Qian, Y., & Cheng, L. (2015). Integrated Foreground Segmentation and Boundary Matting for Live Videos. *IEEE Transactions on Image Processing*, *24*(4), 1356–1370. doi: 10.1109/tip.2015.2401516