Assignment 1 — Foreground Segmentation and Poisson Blending

Omri Tzur 318512308

GrabCut algorithm

Implementation of the grabcut algorithm according to what we have learned in class + added article:

1.Initialize two Gaussian Mixture Models (GMMs): one for the background and one for the foreground .

2.Create the N-link capacities.

3.Iterate until energy convergence is achieved or up to the default of 5 iterations:

3.1 In each iteration, update the GMMs based on the most recent mask.

4.Create a graph where each vertex represent a pixel in the image + source/target vertexes.

4.1 Add edges from the source vertex to each vertex(pixel),each target and between any pair of vertexes vertex to pixels. that represent neighboring

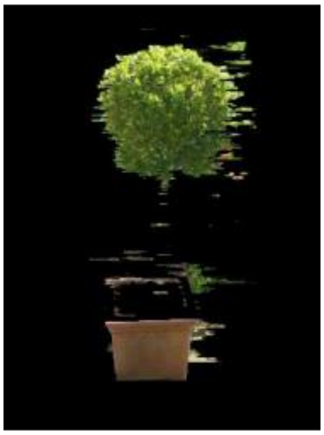
5.Finding minimal cut, updating the mask and check for energy convergence.

For the majority of images, this method produced satisfactory outcomes in terms of both accuracy and quality (using accuracy and the Jaccard index).

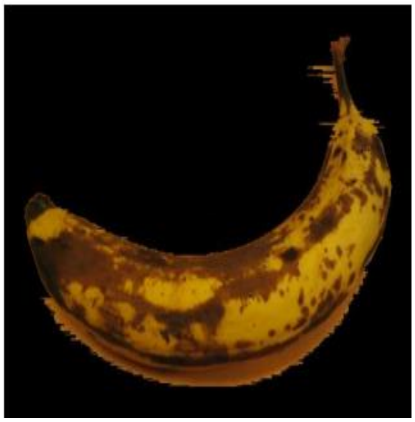
However, there were a few images where the results were not that successful.

let's take a look at the failure cases:

On the bush image I found it very difficult to extract the bush from the image. To my opinion the main reason is the similarity between the background and the leaves. I've tried few changes as increasing/decreasing the numbers of iterations or GMMs but it didn’t solve the problem and caused to removal of main parts of the bush like the stem.

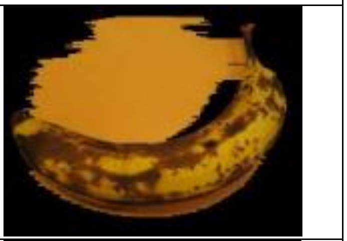
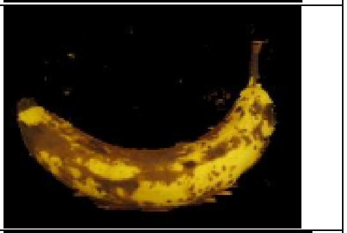
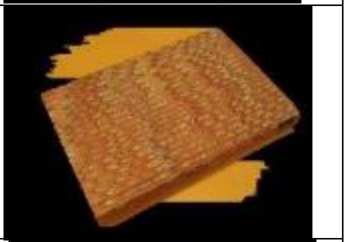
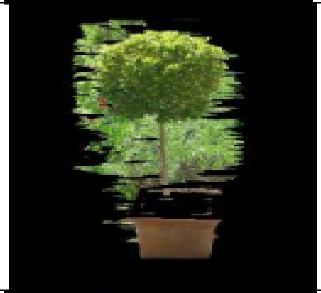
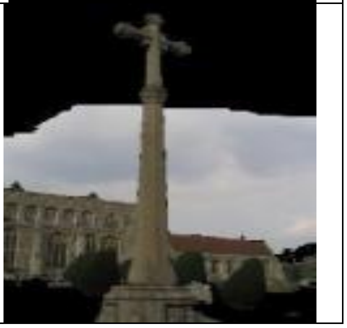


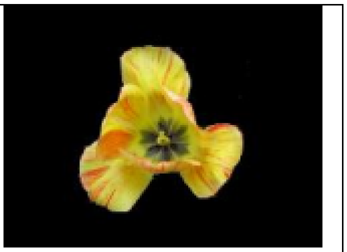
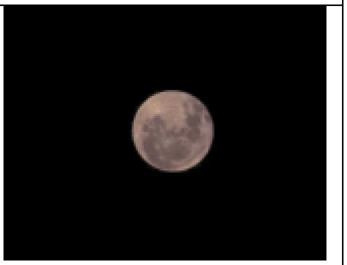
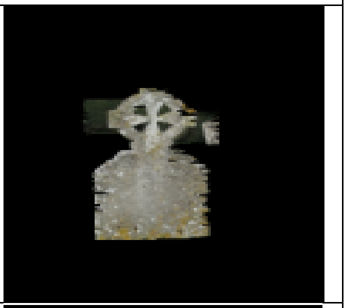
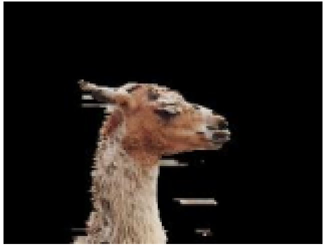
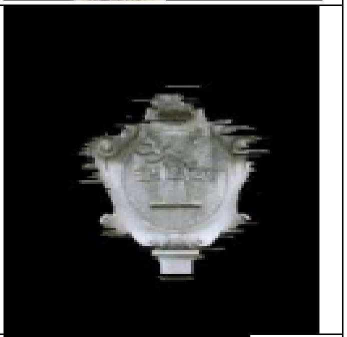
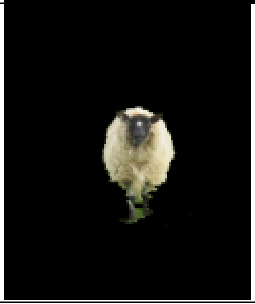
The banana1 image was the first image to try and it wasn’t good. When running the banana2 image and getting quiet good results I noticed the difference between their backgrounds. The banana1 background colors are much more similar to the banana itself and make it more difficult for the grabcut algorithm to separate between the banana and the background.It took me a while but eventually I found out that increasing the amount of iterations and using only 2 GMMs is the key to the best result.



All images results

The following results calculated by using the default settings (5 iteration of the GrabCut algorithm and 5 GMMs) with convergence threshold of 700 (I tried a lot of options and got to conclusion that 700 is the best case).

image	jaccard	accuarcy	time	Result
Banana 1	0.4673	0.6916	1:28	
Banana2	0.9413	0.9767	1:01	
book	0.7878	0.9004	1:47	
Bush	0.4887	0.8219	0:30	
cross	0.4345	0.6220	01:16	

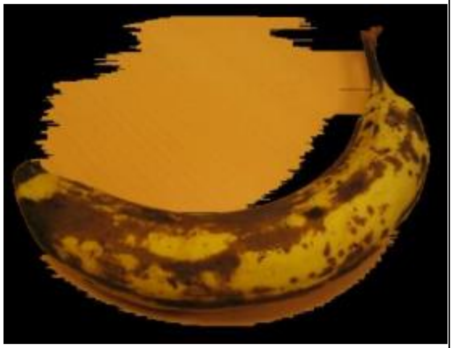
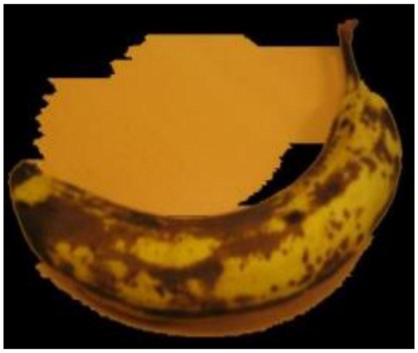
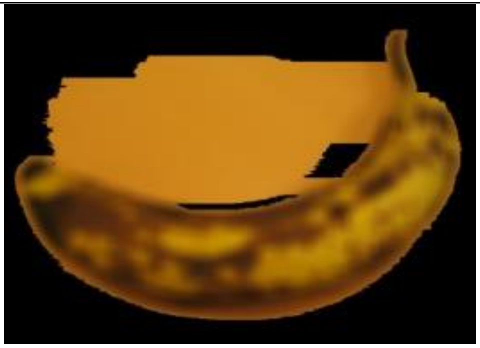
flower	0.9721	0.9950	00:34	
fullmoon	0.9060	0.9938	0:08	
grave	0.7983	0.9761	00:26	
llama	0.9066	0.9891	0:25	
memorial	0.9018	0.9661	0:35	
sheep	0.9194	0.9961	0:25	

Other methods

The most complicated images were the banana1 and cross so I chose them to look at the differences between them and between the requested methods.

Banana blur results:

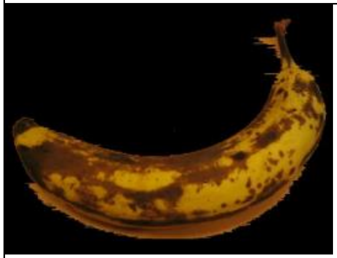
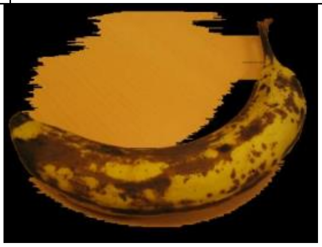
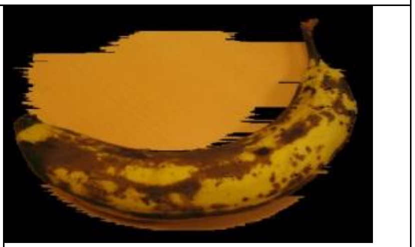
No blur	Small blur(5,5)	High blur(20.20)


Jaccard=0.4591
Accuracy=0.7004
	
	Jaccard=0.7109
	Accuracy=0.9511
		
		Jaccard=0.4851
		Accuracy=0.7283

I was expecting that the blur method will reduce the noise levels and increase the clarity od the borderlines but as we can see on the banana case we can see a very small impact of the blur , I believe its because of the similarity of the background and the banana colors(as I mentioned earlier).

Banana with different GMMs results:

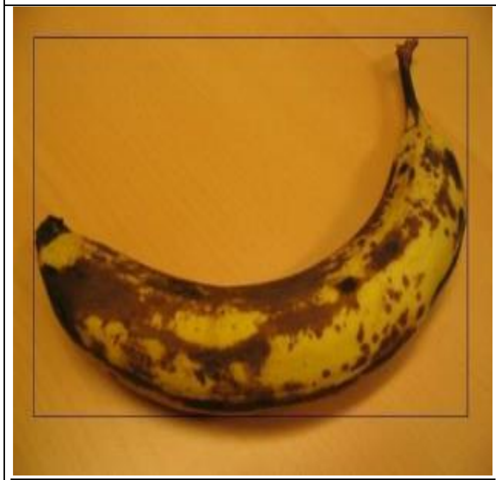
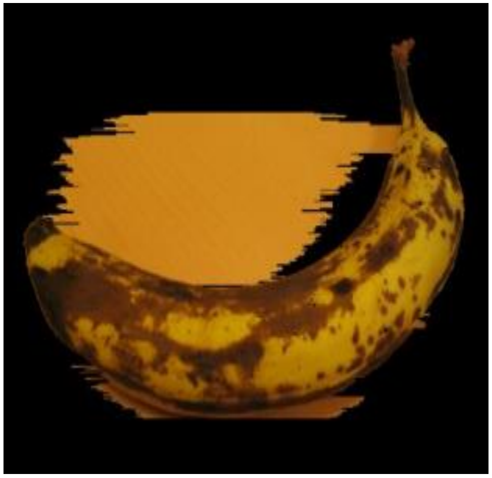
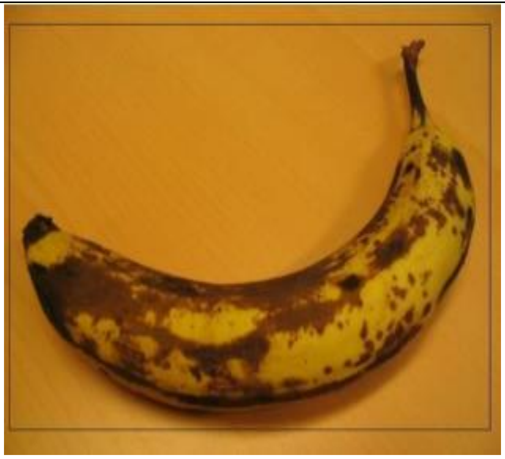
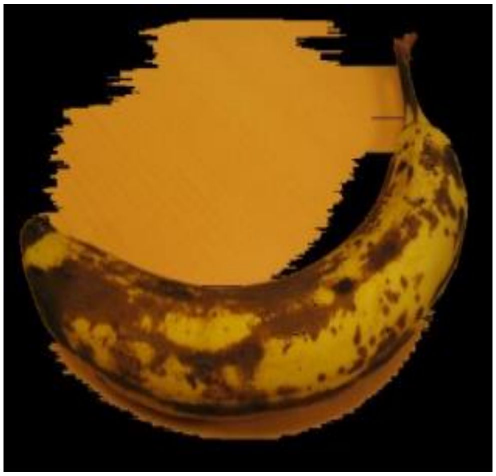
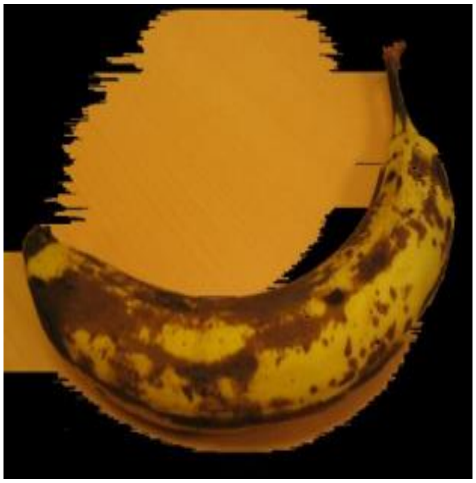
2 GMMs	5 GMMs	10 GMMs


Accuracy=0.9707
Jaccard=0.8566
	
	Accuracy=0.6948
	Jaccard=0.4558
		
		Accuracy=0.7043
		Jaccard=0.4655

We can see that using only 2 GMMs yielded better results than the 5 GMMs which were the default. On the other hand the 10 GMMs didn't improve the results and even got worse. From the banana example we can understand the impact of the GMMs because Increasing the number of GMMs can create more clear details and is much more useful for complex textures or gentle color variations that need distinction , it can also overfit noise as foreground.

Rectangle Initializaion Results

mall	**Default**	**Big**

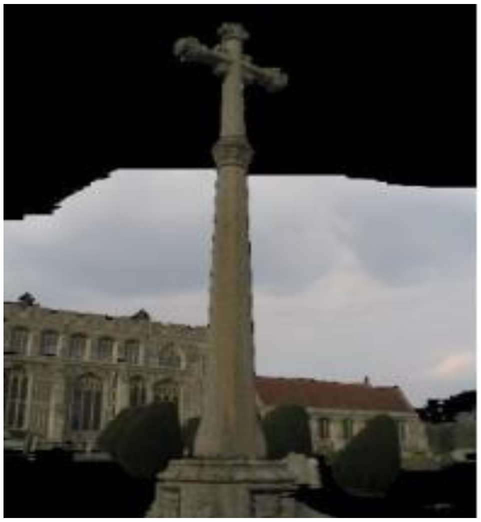
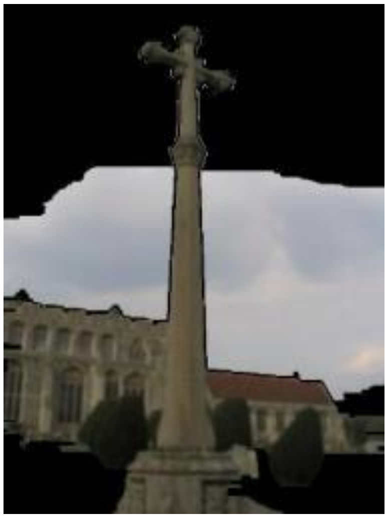
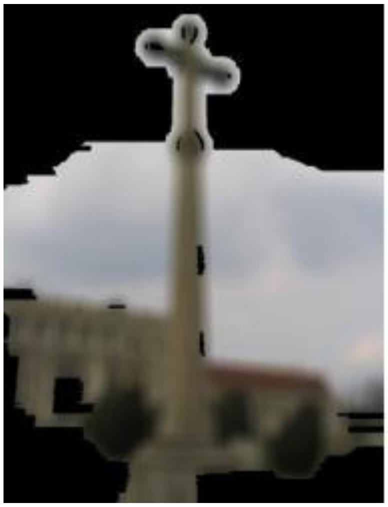


Accuracy=0.9745
Jaccard=0.9354
	
	
	Accuracy=0.7003
	Jaccard=0.4629
		
		
		Accuracy=0.6471
		Jaccard=0.4288

We can see that smaller rectangle create better results because the first estimation of the borderlines is much more accurate and minimize the noise during clustering.

On the other hand , the predefinition of the BG pixels make it more difficult to classify correctly the BG pixels from inside the rectangle because the lack in their neighboring pixels. We'll this impact at the cross image.

Cross blur results

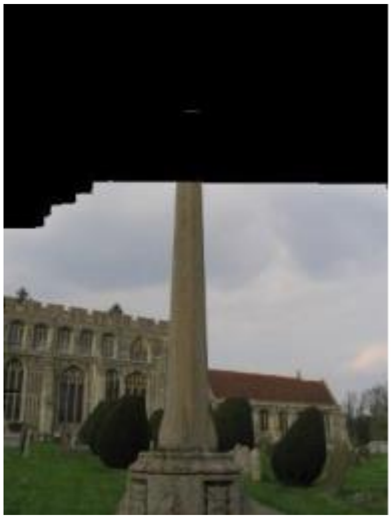
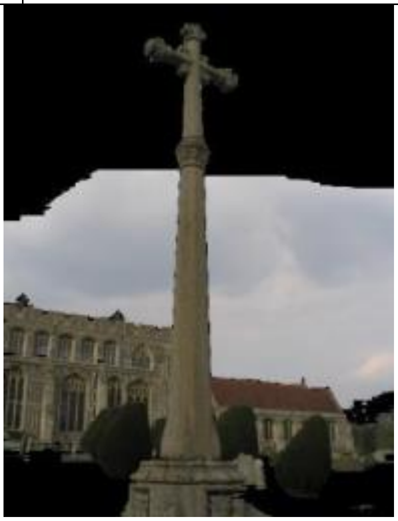
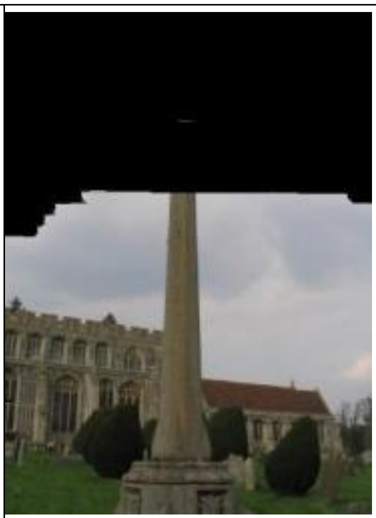
No blur	Small blur(5,5)	High blur(20.20)


Jaccard=0.4472
Accuracy=0.6202
	
	Jaccard=0.4230
	Accuracy=0.6098
		
		Jaccard=0.3465
		Accuracy=0.5339

We can notice that the small blur barely made a difference , but as we saw at the banana image , the heavy blur remains ineffective ,make the results much worse and rendering the banana and cross objects quite unclear.

Cross GMMs results:

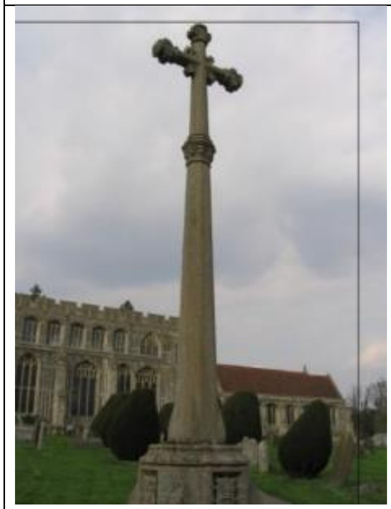
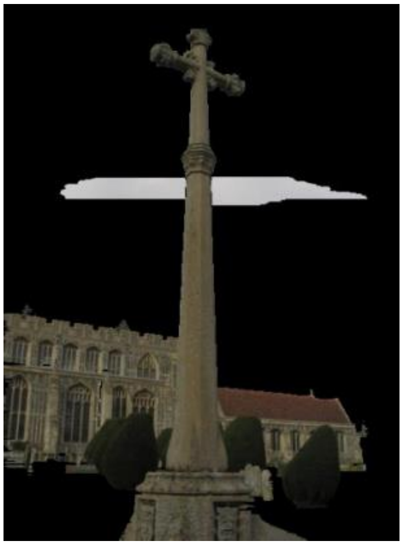
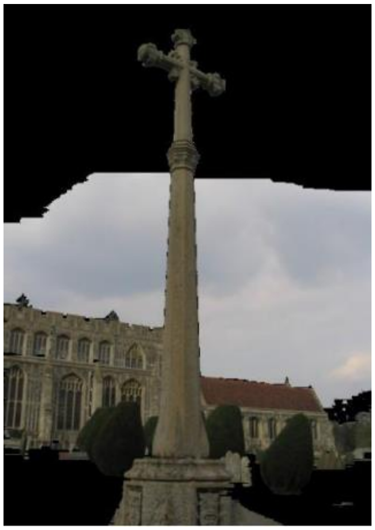
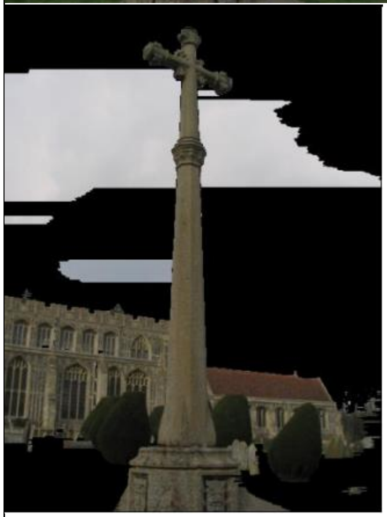
2 GMMs	5 GMMs	10 GMMs


Accuracy=0.6910
Jaccard=0.5117
	
	Jaccard=0.4339
	Accuracy=0.6211
		
		Accuracy=0.7001
		Jaccard=0.5433

as opposed to the banana image we see at the cross image that both decreasing and increasing the number of GMMs is only damaging the result. The results led me to think that each image should have it specific ideal GMMs number according to her texture and complication for achieving the best result and avoiding underfitting/overfitting.

Rectangle Initializaion Results

small	Default	big



Accuracy=0.9014
Jaccard=0.7228
	
	
	Accuracy=0.6198
	Jaccard=0.4446
		
		
		Accuracy=0.7373
		Jaccard=0.5411

As we saw on the banana image , the smaller rectangle gave us the best result.

The surprising part is that the bigger rectangle improved the accuracy of the default rectangle although the difference between them is very minor.(honestly I don’t understand why)

Poisson Blending algorithm

1.First, I check the dimensions of the input images to ensure they are compatible with the target image. If necessary, pad the input images with zeros to center them on the background image.

2.define the Laplacian operator to initiate the blending process.

3.Construct the matrix A for the equation. For each pixel labeled 0 in the mask image, set the corresponding row in matrix A to 1, ensuring the result will use data from the target image for those pixels (considered background).

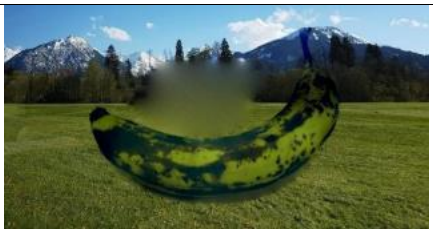
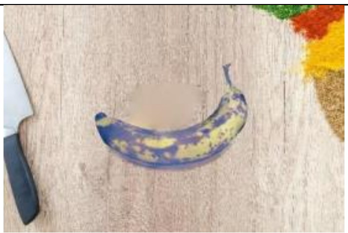
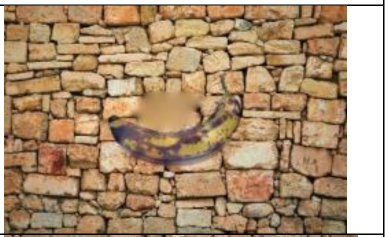
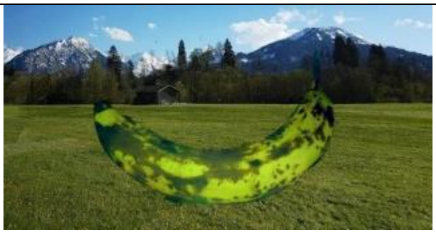
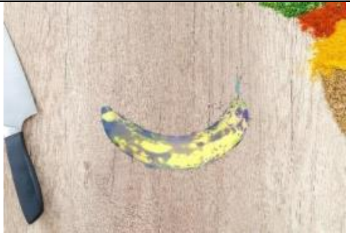
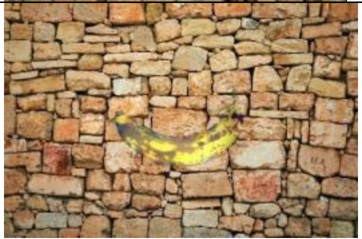
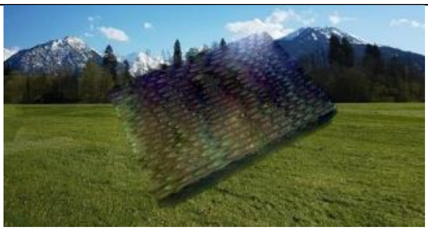
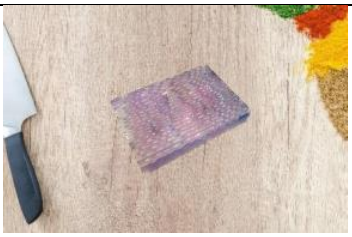
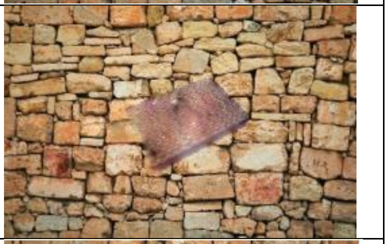
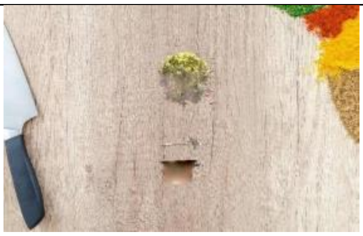
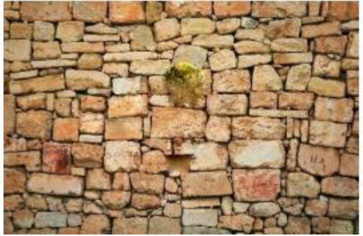
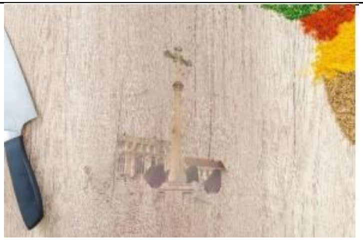
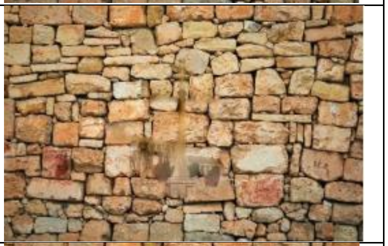
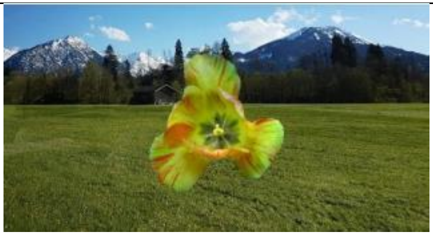
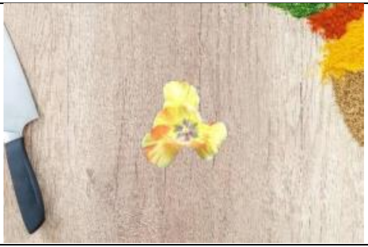
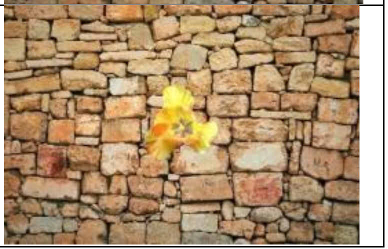
4.For pixels labeled as foreground in the mask image and located at the borders, remove their entry from matrix A according to the 'difficult approach' explained in the clarification.

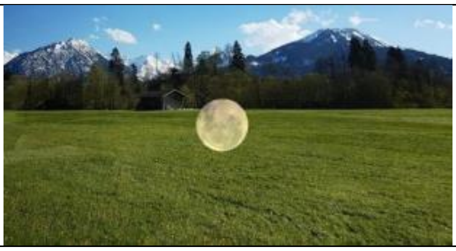
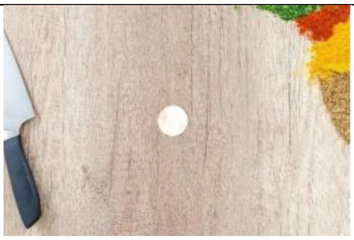
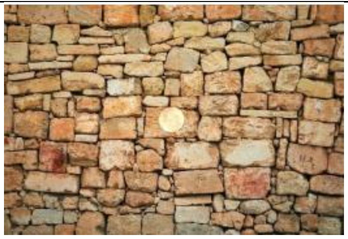
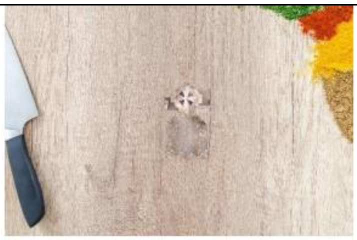
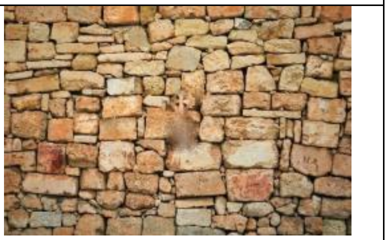
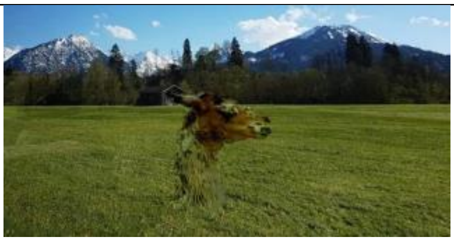
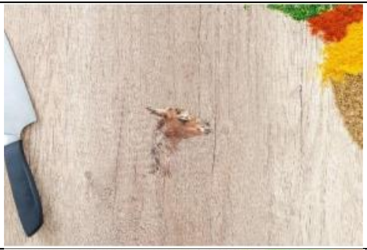
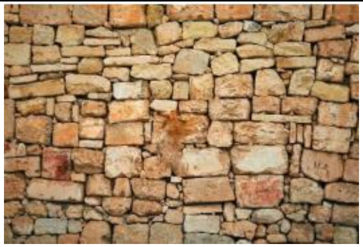
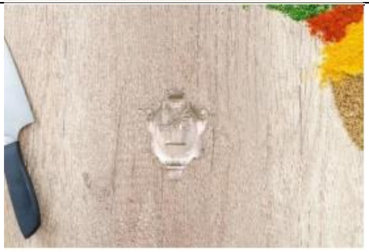
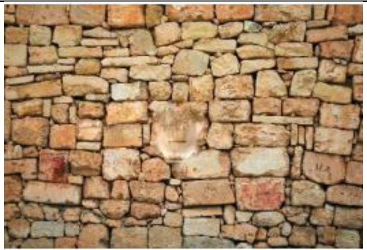
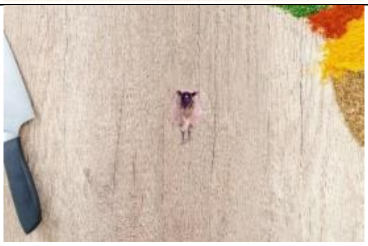
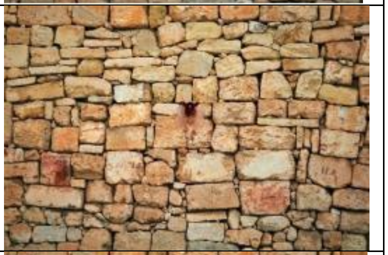
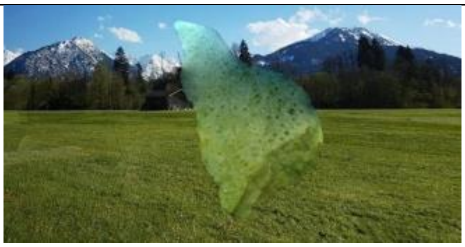
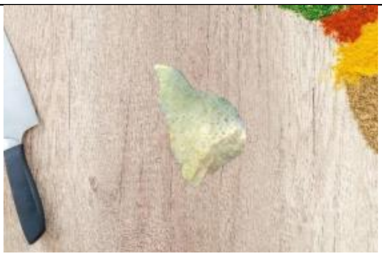
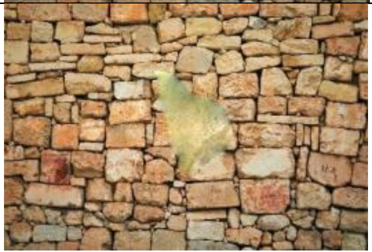
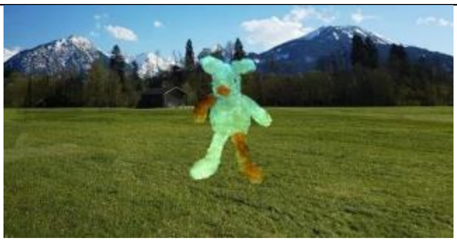
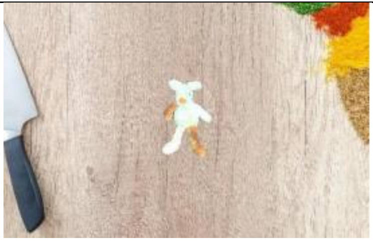
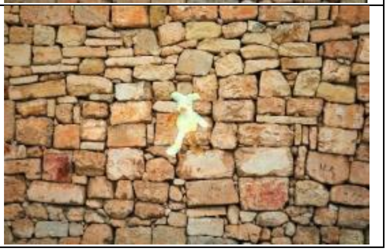
5.To solve the Ax = B equation, create the vector B separately for each color channel (RGB), then merge the results.

The solution to this equation provides the final blended image.

In certain images, the results were unsatisfactory due to masks that were not accurately aligned with the object, leading to blurred edges and smudging. In these cases, parts of the background were incorrectly identified as foreground, causing visible artifacts around the object. Although the blending was not perfect, it still performed reasonably well. Another challenge occurred when the colors of the object and the background were too similar, making it difficult to clearly distinguish the object after blending, as it seemed to fade into the background.

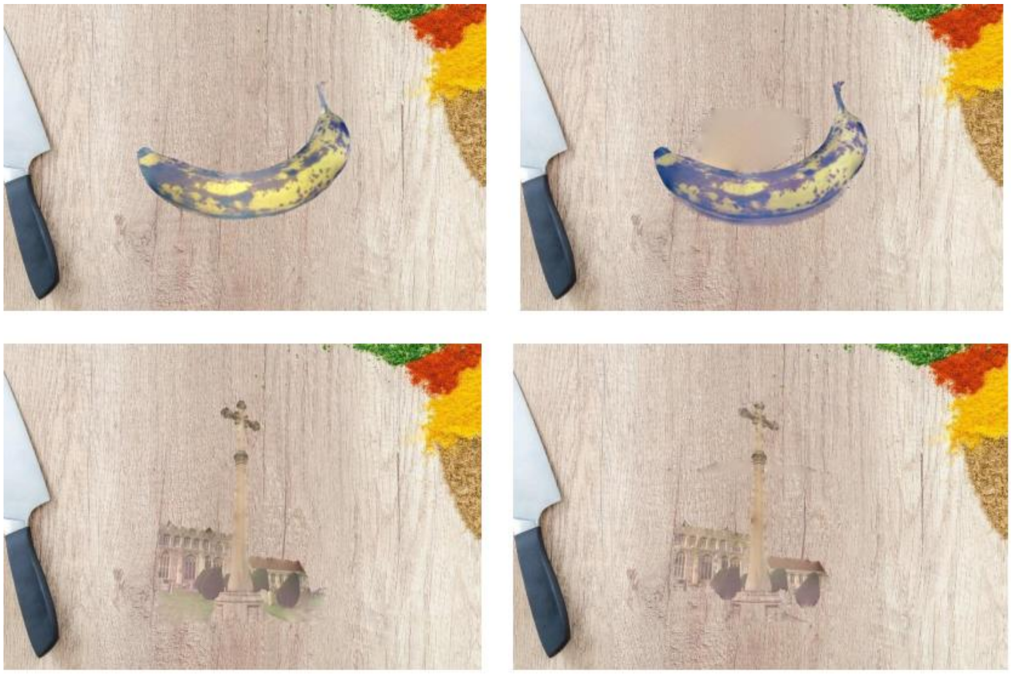
Results

name	Grass mountain	Table	Wall
Banana1			
Banana2			
book			
bush	**Failure**		
cross	**Failure**		
flower			

fullmoon			
grave	**Failure**		
llama			
memorial	**Failure**		
sheep	**Failure**		
stone			
teddy			

Results with Not Tight Mask

Those are the results of the blending using the mask we created for the banana and cross on the table to the result of the blending with the mask we received for it.



The goal is to blend an object from one image into another by minimizing the difference between the source and target images. The mask defines the region of the source image that should be blended into the target image. If the mask is not tight around the object, some parts of the background may be included in the blending process. These parts will then influence the Poisson equation's solution, resulting in blurred or distorted areas around the object, as we can see clearly above the banana and around the cross. This occurs because the gradient values of the background outside the object's boundary will differ from the gradient values of the object, leading to a discontinuity in the solution of the equations used for the blending.