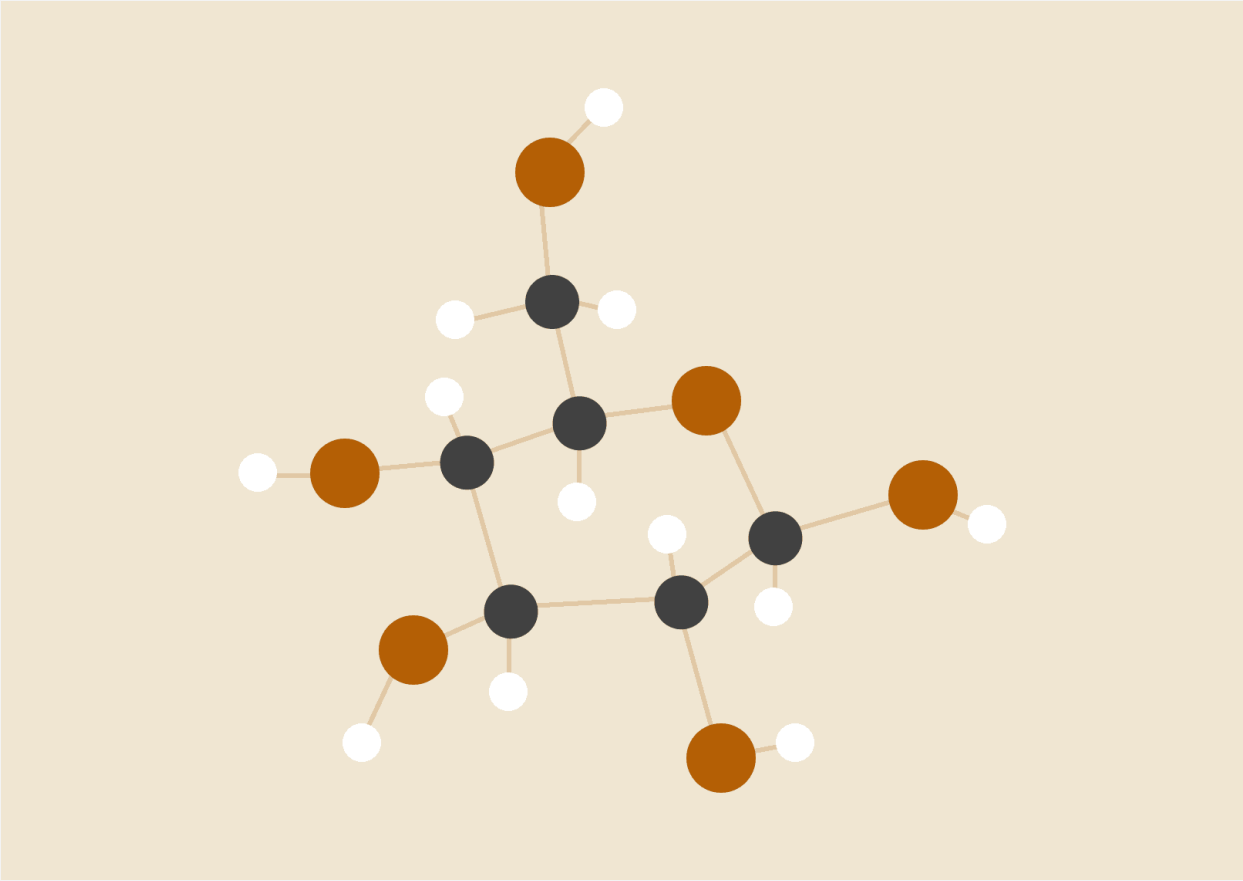
SPYNE Assignment for Computer Vision Researcher



By **Kirtan Mali**

18-10-2024

Project 1: Image Manipulation and Background Replacement

You have been provided with a set of car images, each accompanied by relevant information. Your task is to recreate the provided sample output by manipulating the images based on the following instructions.

https://drive.google.com/file/d/1FSq7BzhJjhz4UnRgkv4hR7QKZS_7vDcu/view?usp=sharing

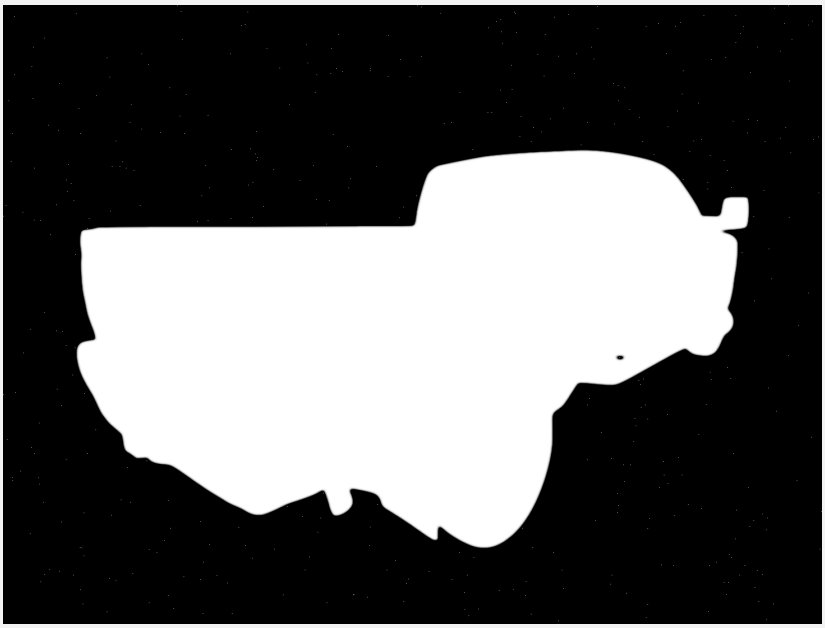
Given the image on the left side along with binary background mask and shadow cutouts, the tasks to perform should generate an image similar to the right image with a custom wall and floor.

The basic workflow for solving the problem,

- Firstly, remove the old background using the car_mask.
- Find the position of the shadow mask with respect to the car.
- Blend the shadow with the car properly.
- Create a custom background using the wall and floor images given.
- Shift the car and shadow to match the perspective of the background.
- Color correct the image to get a perfect match of foreground and background.

TASK 1 - Background Replacement

Given a binary mask for the background, I have to remove the background from the given input images.



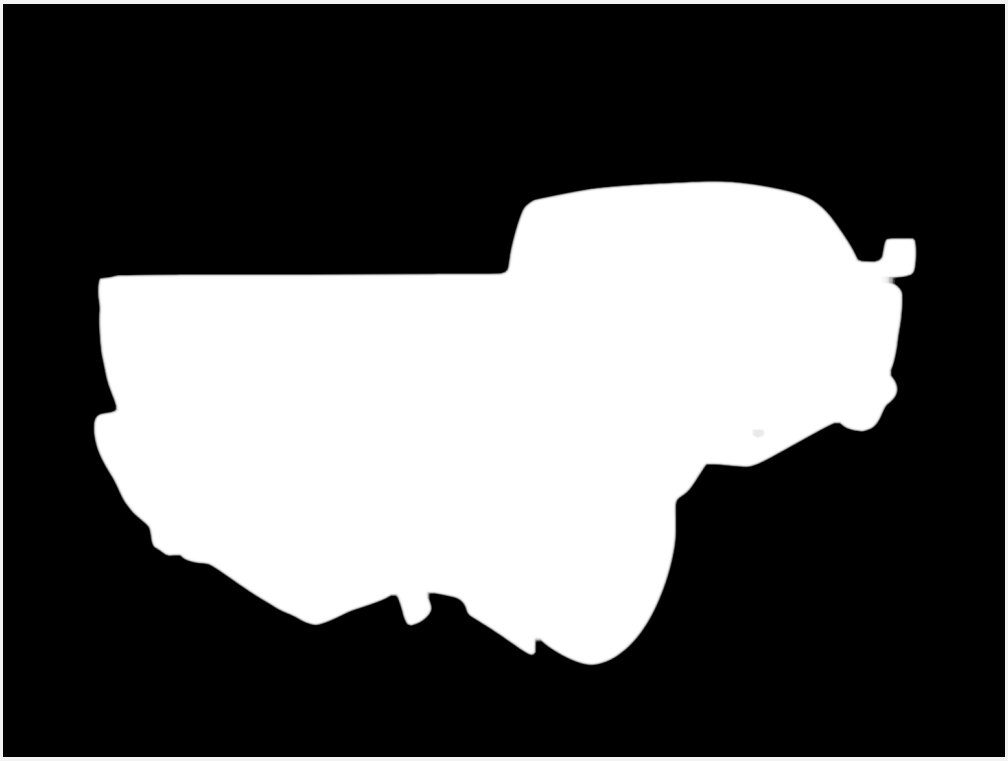
[a] Noisy Car Mask

CHALLENGES FACED

1. The given binary mask had noise of unknown distribution.
2. The binary mask had holes in some places. (Like the dot visible in the white area inside the car)

PROCEDURE

1. To remove pixel noise from the image, using a median blur filter.
2. The noise in the image is essentially just high frequency information that is just not needed in the image and a medianBlur filter works just like any low pass filter.
3. Using morphological operations like using eroding and dilating solves this problem.
4. In Opencv, we first generate a kernel of size `(10,10)` using cv2.getStructuringElement of cv2.MORPH_ELLIPSE shape. And then applying this kernel with the cv2.MORPH_OPEN flag on the inverted binary mask and all the holes should be gone.



[b] Processed Car Mask

TASK 2 - Shadow Placement

Given the shadow mask cutouts, the task is to find the location where to place the shadows and properly blend them with the image.

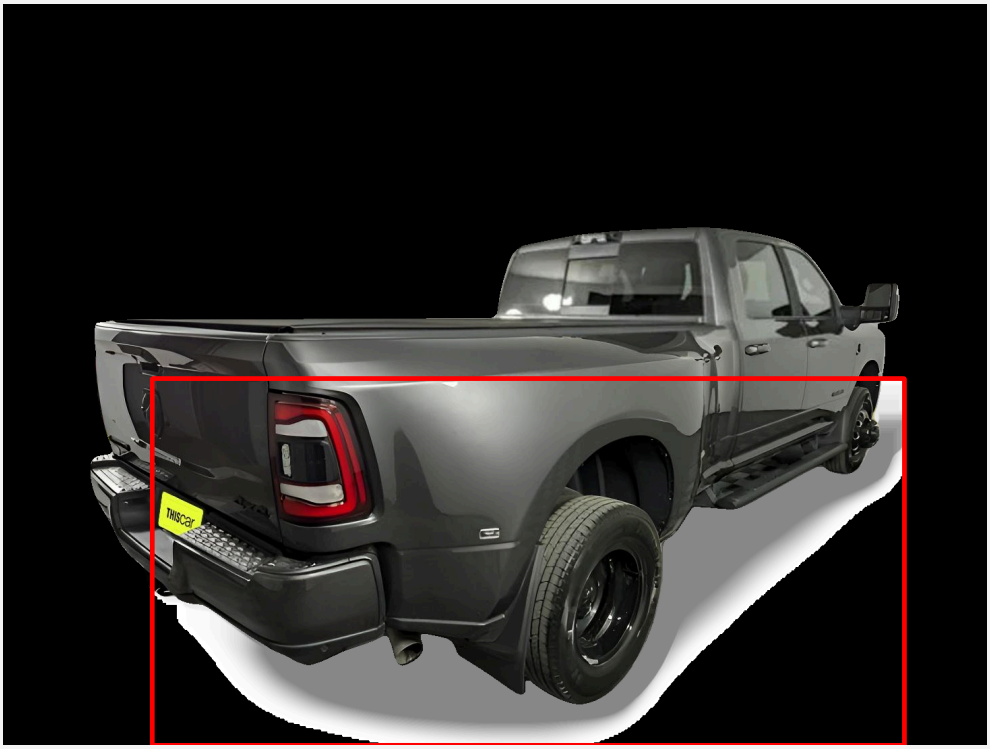
[c] Shadow cutout given

CHALLENGES FACED

1. The main problem here was that the shadow cutouts were of dimensions smaller than the original input image.
2. The shadow cutouts do correspond to the size of the car in image but its position is unknown with respect to the car.

PROCEDURE

1. So, I make use of the background binary mask (with bg as 1 and car as 0) and the thresholded shadow cutout.
2. Perform opencv template matching on both and find the relative position of shadow with respect to the car in the original image.
3. As shown in the image below, the white part corresponds to the shadow matched.
4. And the red rectangle shows the position of the shadow template matched.



[d] Relative positioning of shadow w.r.t car

TASK 3 - Perspective Matching

Here, we want to generate a custom wall and floor background image that matches the perspective of the car and shadow generated.

CHALLENGES FACED

1. The custom background generated from the reference wall and floor images, may not match well with the position of the car in the input image. Also, the difference in lighting also makes a very unnatural composition.

PROCEDURE

1. We crop certain portions of the wall and floor from their respective reference images keeping in mind the aspect ratio of the input image.
2. Note we do not use resizing here so as to not distort the image.
3. Perform color correction so that lighting conditions match background.
4. Lastly we shift the car and shadow generated lower with respect to the custom background, so that merger of car and background does not generate an unnatural composition.

[e] Final result image

Project 2: Car Image Classification

You have been provided with a dataset containing car images. Your task is to train a classification model that can accurately predict the angle of the car in the image.

TASK 1 - Data Analysis and Preprocessing

There are 8 classes in total namely, ["0", "40", "90", "130", "180", "230", "270", "320"] angles. With each class have counts,

0	40	90	130	180	230	270	320
4357	655	3452	4320	4433	3111	3225	3489

We can see the classes are not evenly distributed, especially class “40”, with the lowest frequency among all classes.

So, I perform transformation and augmentations as pre-processing steps.

- Firstly **Resize** the input image to (256, 256) shape keeping the rgb channels.
- Then do a **RandomCrop** operation to (224, 224) shape.
- Applying **ColorJitter** to increase variation in the dataset with the parameters, brightness=0.2, contrast=0.2, saturation=0.1, hue=0.1.
- Then **Scaling** the input to range [0, 1.0], and converting it to tensor.
- Finally doing a **Normalization** with (0.485, 0.456, 0.406), (0.229, 0.224, 0.225) as mean and s.t.d. Using this mean and s.t.d as the models are pre trained on ImageNet dataset.

TASK 2 - Model Training

Training 3 classification models and comparing them.

Namely: VGG16, ResNet50, MobileNet_v3_Large

All models are trained using the same training scheme.

- CrossEntropy as the Loss function for the classification task.
- Adam as the optimizer with learning rate 0.001 and default parameters as provided by pyTorch.
- Learning rate decay scheduler StepLR with step_size 7 and gamma value of 0.1.

Model	Training Loss	Val Loss	Training Acc.	Val Acc.
VGG16	0.0301	0.2174	0.9907	0.9564
ResNet50	0.2579	0.3012	0.8723	0.8544
MobileNetV3	0.1883	0.2604	0.9335	0.9142

Surprisingly VGG16 performs the best, and it seems like it had overfitting on the data. Considering its large number of parameters and small size. It is very slow. So, I mostly use MobileNetV3 for the API.

TASK 3 - API Development

Using Fastapi as the api development framework. It accepts raw images as input and returns a json dictionary with two keys for predicted class angles and corresponding confidence score.

The api endpoint `/get_car_angle` returns the predicted and confidence.

To run the api locally,

```
cd <path-to-the-repo>/project_2

python3 api.py
```

Then, for the client side python script to send image, file can be found in **project_2/client.py**

```
import requests

URL = "http://0.0.0.0:8000/get_car_angle"

IMAGE_NAME = "<path-to-image>"

with open(IMAGE_NAME, 'rb') as f:

    files = {"file": (IMAGE_NAME, f)}

    response = requests.post(URL, files=files)

    print(response.json())
```