Universidade Federal de Campina Grande
João Vitor de Melo Cavalcante e Souza

Para a segunda etapa do processo seletivo, escolhi “Object Tracking”, que deve utilizar primariamente a biblioteca Open CV para: detectar objetos por cor (1), desenhar triângulos nos objetos a serem detectados (2), ter uma interface gráfica para a criação da máscara (3) e exibição de parte da trajetória do objeto (4).

Dividi o processo em partes visto que seria um processo longo para cumprir em apenas “uma vez”. Para os pontos (1) e (2), estudei de perto o video “How to Detect Colors in OpenCV [Python]”, fornecido na própria ata do desafio. Acompanhando o vídeo, fiz dois programas (proto-colorDetector.py e sua variação proto-colorDetectorGif.py) que, mediante uma imagem de um círculo colorido, “detectam” sua cor por meio de máscara. Máscaras de forma simplificada, que funcionam como filtros, são geradas em cima de uma imagem e, de acordo com certos parâmetros (HSV), devolvem a mesma imagem (que pode ser o frame de um vídeo ou gif, como no caso de proto-colorDetectorGif.py) já filtrada. O exemplo que realizei e que trabalhei ao redor foi com o gif “bola3.gif”, que diz respeito a um círculo amarelo que contrai e expande regularmente. É desenhado um retângulo vermelho ao redor da forma detectada e sua máscara é demonstrada em outra janela.

No ponto 3, estudei também pelo vídeo fornecido “Tracking com Ajuste Dinâmico de Cores - Python & OpenCV 4 #08” o qual tratava sobre a criação e alocação dos parâmetros para criação da máscara de forma dinâmica. Para tal, teve de se criar uma “TrackBar”, ou seja, uma janela com barras para determinação, pelo próprio usuário, dos valores HSV a serem “deixados” passar pela filtragem e que estariam presentes na máscara. Para este ponto, que registrei em “proto-tracking-video.py”,

utilizei a filmagem de uma estrada com carros a passar e que, mediante a seleção de parâmetros HSV na janela trackbar, um retângulo seria desenhado na maior área da máscara. A etapa 3 serviu de expansão para as duas primeiras que apenas detectam a cor de forma “estática”, sendo necessária alteração no próprio código para diferentes escolhas de cores e saturações.

Finalizando, no programa definitivo “flowTracking.py”, expandi mais uma vez as funcionalidades essenciais do programa. Para detecção da trajetória das áreas da máscara (que podiam ser carros no exemplo de road.mp4), criei um buffer e, durante o funcionamento do código do ponto 3, preenchi ele assim como um vetor com coordenadas centrais dos retângulos desenhados nas maiores áreas, como funcionará o código anterior. Ao meio da execução, o programa checar se o buffer de tamanho 32 já estaria cheio e, fazendo diversas checagens acerca da direção de maior predominância no vetor do “centro” do retângulo, poderia determinar: (1*)Qual das 4 direções cardinais o veículo poderia estar indo, (2*)Desenharia uma linha na trajetória percorrida e (3*)igualmente demonstraria as coordenadas do centro do retângulo.