MODERN AI AND ROBOTICS APPLICATIONS

1. Rigid-Body Motion

DETERMINE INVERSE OF A ROTATION AND TRANSLATION

A rigid-body motion in 3D space is determined by a rotation matrix *R* and a translation vector *T*. We know the transformation from point 1 to point 2 is expressed as:

$$p_2 = Rp_1 + T$$

(1) Write the transformation between point 2 to point 1, namely, if

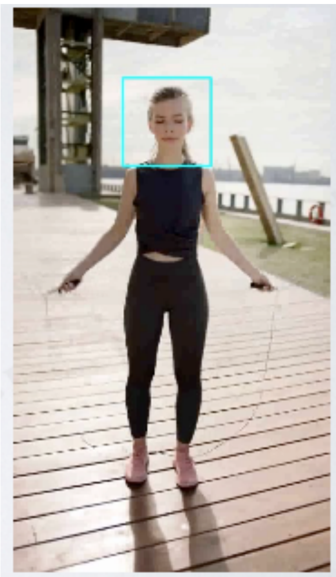
$$p_1 = R'p_2 + T'$$

express R' and T' using *R* and *T*.

(2) Derive the transformation matrix *M* in homogeneous coordinates using *R* and *T*, namely, $\begin{bmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{bmatrix} = M \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix}$.

MODERN AI AND ROBOTICS APPLICATIONS

2. Object Tracking



REVIEW DIFFERENT OPENCV TRACKERS

Please use jump_rope.mp4 file provided in Lecture 5, and extend the object tracking solution from our discussion on KCF during the class to other types of trackers, including TLD, MedianFlow, and GOTURN.

After the implementation of these different tracking solutions, please write a review to report your observations about the quality of the trackers applied to the video file. In particular, please pay attention to the number of lost track frames, the amount of total time to complete, and the quality of the returned bounding box.

Intelligent Racing

3. Image Convolution

IMAGE BLURRING USING WEIGHTED KERNEL

Recall Canny Edge Detector uses a 5×5 weighted blur kernel:

$$\frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}.$$

Please use filter2D() function in OpenCV to blur every frame in highway_video.mp4 video, and then save the results in an output avi file.

Intelligent Racing

4. Difference of Gaussians

EDGE DETECTION USING DIFFERENCE OF GAUSSIANS

Image edge features can be calculated by other methods. In this problem, we will implement another well-known framework called Difference of Gaussians (DoG)

Please use the Lincoln image as an example, apply two different sizes of Gaussian Blur to generate two returning blurred images, and then subtract the two images to form the DoG image. Finally, plot the DoG image as a binary image, namely, select a threshold where those DoG results higher than the threshold will be 255 grayscale value, while those DoG results lower than the threshold will be 0 grayscale value.

Tip: A one-line Gaussian blur function in OpenCV is cv2.GaussianBlur().

5. Floodfill Algorithm

DISPLAY FLOODFILL RESULT IN A MASK

Based on the floodfill_segmentation.py code in Lecture 4, please modify the function call cv2.floodFill() and add a new mask input argument. Further, please refer to the OpenCV documentation to set the proper flag values to have the algorithm to:

1. Search from neighboring 8 connecting pixels instead of default 4;
2. Only return the flood fill results in the mask image and do not modify the input image.