Converting Pixels to Real-Life Distances Through Detection of an Object With a Known Size to Aid in Stroke Rehabilitation

By Gavin Fountain, Jun Min Kim, Emily Mao, Francisco Carcamo

Introduction

The “Box and Block” test is a common rehabilitation procedure that individuals who have gone through a stroke go through. The test consists of two wooden boxes split in the middle by a vertical divider. Individuals must move small wooden blocks from one box over the divider to the other. While this task is trivial for those who have not suffered from a stroke, it is very difficult for someone to do with the arm they have experienced a stroke with.

Blocks that the individuals moved over the divider are counted within a set time. How many blocks they move, along with how they moved their arms and hands is useful in determining rehabilitation progress.

An example of a Box and Block test is shown below:

So far, there has been no automation of this process. A physical therapist must sit with an individual and time the patient. This project is part of a larger research project being done by Professor King and Professor Farrens of UC Davis to automate the process.

The section of the project this report focuses on will be the topic of determining the real-life distance of each pixel in the video, which will allow us to determine the speed of an individual's hand.

Motivation: why is this an important problem to solve?

The number of blocks that an individual was able to get over the divider and their hand speed are important statistics that physical therapists use in stroke rehabilitation. If we are able to fully automate the process, it could lower the cost of stroke rehabilitation and make it much more accessible. Rather than a person having to book an appointment with a physical therapist, they could simply record a video of themselves doing the Box and Block test. The results and the statistics of the test could then be sent to a physical therapist or a doctor for examination.

So far, there isn't a method applied in this field to determine the speed in which a person's hand moves. If we could automatically determine how much distance each pixel in a video is for every video,

we could calculate the speed of the hand (the project already has ways to track a hand and get the pixel speed at which it moves).

Why is this a difficult problem to solve?

The main challenge in this project is to have a tight bound around an object of a known size so that we have the length of the object in pixels. Then the pixels could be translated into real-life distance by dividing the number of pixels by the known length/width/height of the object.

In our case, there is always one constant object of a fixed size in all of the videos: The Box and Blocks set, the dimensions of which we know. Therefore, if we can have a tight bound around any of the dimensions of the box, we have the real-life distance of each pixel, which can then be used to determine the speed of the hand in centimeters.

Note: The speed of the hand in pixels is determined by another part of the research project where a machine learning algorithm tracks the hand and its speed in pixels.

At first, this problem seems trivial–why not simply use a box detector or an edge detector? However, these simple approaches proved ineffective, as different videos were captured at varying distances, camera angles, lighting conditions, and video qualities. A more advanced approach was needed for a robust computer vision algorithm that could bound the box under differing video conditions.

We would also like to note that both Professors King and Farrens considered this pixel to real-life translation as a non-trivial problem that needed work on, which is where we got the idea for this project. We hope this segment provides enough context to why this problem was not simple.

Dataset

We will be using a dataset obtained at University of California, Irvine of videos of different patients performing the Box and Block test. Permission to use the dataset was obtained by Professors King and Farrens, but the videos (other than images taken to help demonstrate in this report) are kept private.

Methods

We will try the following methods and compare each model to see which one was most successful in tightly bounding the Box and Blocks test box in the video.

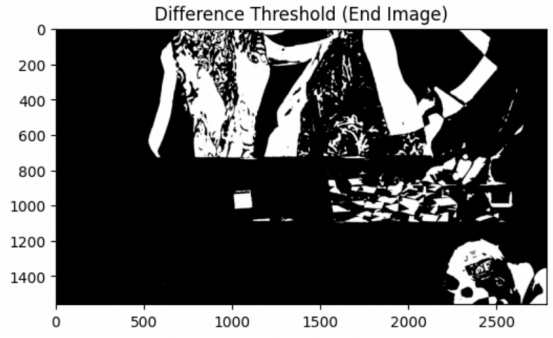
1) Contour-based Object Detection
2) Color-based box detection
3) Box detection using the Segment Anything Model (SAM)

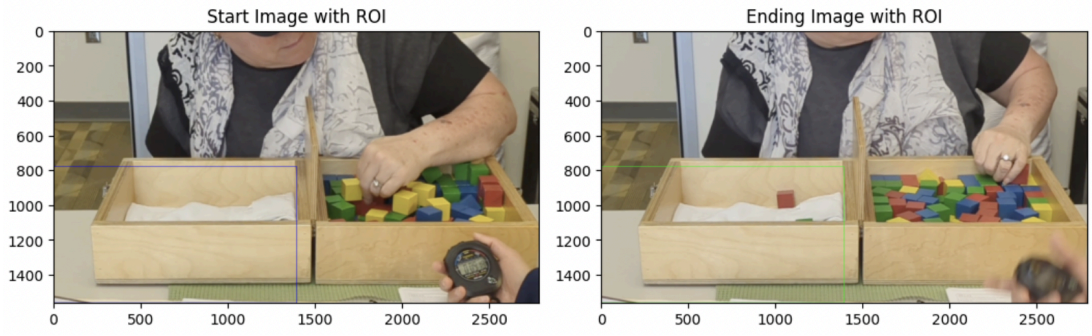
Contour-based Object Detection

Contour-based object detection identifies objects in an image by detecting and analyzing their boundaries or edges, often using techniques like edge detection (e.g., Sobel, Canny) and contour tracing algorithms. This method leverages the shape and geometric structure of objects, making it effective for tasks where clear and distinct object outlines are present. It is widely used in

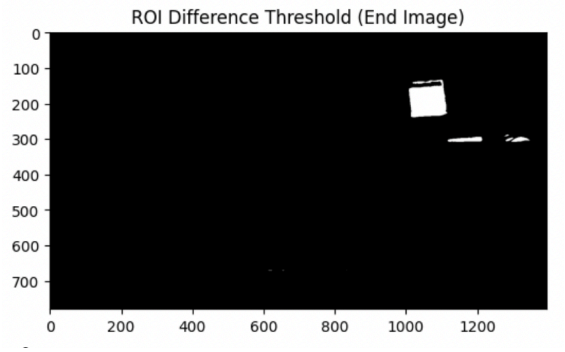
applications like shape recognition, object segmentation, and tracking in controlled environments.

Methods:

1. Extract frames of the start and end of the video.
   a. 
2. Convert the images to grayscale to simplify processing and reduce complexity
3. Compute the absolute difference between the grayscale images to highlight areas where blocks have moved or changed
4. Convert the difference image to binary, isolating the areas of change. Pixel values above a threshold of 50 are set to 255 (white), and the rest are set to 0 (black) to highlight the areas of change.
   a. 

5. Define the region of interest
   a. Calculate the sum of pixel intensities in each half of the box to determine which region was initially empty.
   b. Highlight the region of interest in both the start and end images
   c. Crop the binary difference threshold image to forums on the identified ROI
   d. 

6. Use cv2.findContour() to detect contours in the binary region of interest image.

a. 

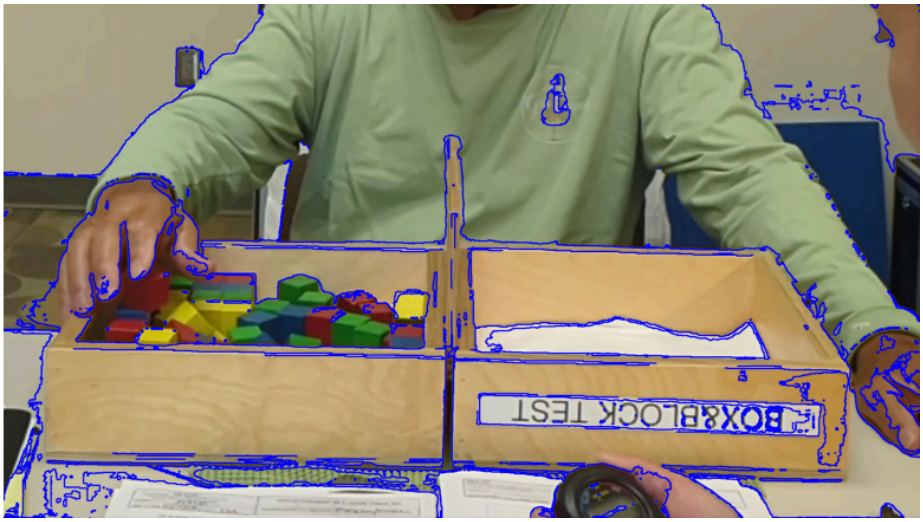
7. The number of detected contours represents the blocks moved into the initially empty region.

With the contour based object detection method, we are able to effectively detect and analyze contours of an image. From the start frame to the end frame of the video, we detect the empty region of the box and identify contours within the region of interest in the end image. However, flaws of this method are that lighting issues of the setting may cause a block to have various contours, and each of the contours may be counted as a separate box. Another obstacle encountered is that at specific camera angles, some blocks may be hidden by the rim of the box. The contours of these blocks will not be identified, and therefore be miscounting. Furthermore, there is a cloth lining the bottom of the box, and when the subject moves the block from one side of the box to another, shifts in the cloth may also result in contours that become identified as a block. Overall, while the contour-based object detection method is able to effectively identify contours of our image, it requires improvements to be used for box detection in the Block and Box test.

Color-based box detection

The color based box detection method was initially designed to search an image based on color to find the edges of the wooden box to then detect when one of the colored cubes crosses the box's edges. At this point in time, we could implement a method to calculate the motion of the box until it passes back into the space of the wooden box. As the colors of the boxes and cubes in the set of videos seemed similar enough, we believed that we could set a range of values separating the wooden box and the red, green, blue, and yellow cubes for easier detection. As such, the steps to follow to accomplish this detection method were as follows:

1. Create a range of values that would encompass the wooden colors of the box
2. Generate the edge of the wooden box based on the range of values

a. 

3. Determine the outer edge of the wooden box and do not check within the box
4. Search for colored cubes outside of the box based on 4 set HSV ranges depicting the 4 colors
    a. Determine center point of the cube for motion tracking
5. Track motion of cube until the cube passes back past the edge of the box
6. Repeat steps 4 and 5 for all blocks to then calculate average speed of motion for this test

These steps were the goals of this method, however, as this was being worked on, there are quite a few issues that should be addressed when considering the viability of this method. The first issue was determining the proper range of color values. As can be seen with the picture included when determining the edge of the wooden box, multiple parts of this freeze frame are selected. With the HSV values, to make sure that all potential boxes, given different lighting conditions and angles within the video, are within our programmed boundaries, a wide range of values are required which can lead to other objects within the images being selected. For example, the wall in the background of this image could be the same color as the box if the box were in poorer light conditions as was seen in another video example. As there are a lot of variables to take into account, issues with determining the specific boundaries of the box.

If the issue of the edge of the box is resolved, then the other issue of detecting the cubes arises. Due to the same issue with the previous part, the range of HSV values required span multiple colors that show up within the images. For example, parts of the box that are more brightly lighted coincide with the range of needed HSV values for when the yellow cubes are more dimly lighted or when camera quality isn't as great. Due to these issues, proper separation of the cubes and other objects within the frame of the image are needed. One method to possibly resolve this is to be able to extract the colors of the cubes at the start of the video to remove the need to program set ranges of values, however, implementing this potential solution was unsuccessful. Overall, until these issues are resolved, this method is not preferable.

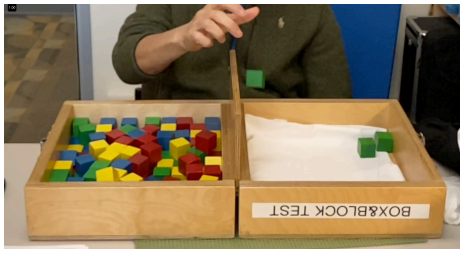
Box detection using the Segment Anything Model (SAM)

The Segment Anything Model is usually used to segment an image into different objects. The technology is commonly encountered when using the "Make a sticker" feature on mobile devices, where users can

make a sticker of an object in the image by doing a long press on it. Since this model has been proven to outline objects well, we wanted to see if it could be used to bound the box.

The following steps were taken to extract the height of the box:

1) Extract a frame from a video where the whole box is visible.



2) Divide the frame into fourths and take the bottom-right piece to take a cropped image
3) Run SAM on the cropped image to segment the image into masks.
4) Score the masks based on the following criteria:
    a) Closest to the target point (¼ of the width and ½ of the height of the cropped image
    b) Largest area
    c) Predicted_iou and stability_score, which are metrics SAM outputs along with the masks
5) Select the mask with the highest score. For all videos in the test set, this successfully filtered out the box like the example image below:

Mask only:

6) Find the right-most white pixel on the mask
7) From the y coordinate of the green dot, descend by 1 pixel on the y-axis at a time, collecting the rightmost pixel for the current y value
8) Stop collecting rightmost pixels when the newest rightmost pixel differs from previously collected rightmost pixel by more than 10 pixels on the x-axis. We do this to prevent pixels that make up the bottom side of the box from being collected

Result:

9) Take the first and last pixel from the collected pixels and calculate the euclidean distance between the two in pixels.
10) We have the approximate pixel count of the side of the box.

We see that by utilizing SAM to bound the box and then using heuristics on the obtained mask of the image we are able to have a tight bound of the height of the box.

Discussion

Out of all three options, we propose using the SAM model as the most effective method for our specific use case. While computationally expensive, the method works well regardless of camera angles and lighting conditions. The other two methods (box detection and color based detection) were especially weak to variations in lighting.

Overall, we propose an interesting method to repurpose SAM—for the translation of pixels into real-life distance given the existence of an object with a known size.

Contributions

Jun Min Kim: Intro, SAM Model, Discussion
Emily Mao: Contour-based object detection method
Francisco Carcamo: Slides and presentation video
Gavin Fountain: Color based detection