Team Cuties
Dasha, James, Gilvir
APCS2 Pd 3

Cuties Computer Vision (CCV)

Description: A program that tracks motion, colors, or objects in a live camera feed or video. When you start the program you will decide whether to use a pre-recorded video or a live camera feed from your computer. The video will displayed on screen or a webcam video will be displayed. By clicking on an object on screen, you are selecting that object as the one that the program will track. You will be prompted with calibration steps such as adjusting contrast and color threshold to help the CV track your object to its best ability. It will draw an outline around the object through each frame. It will be able to track objects even when they are out of view and returned into view.
Post MVP_V1 (post MVP): Tracking multiple objects
Post MVP_V2: A mode where you can interact with on screen shapes and such with your object such as hitting a ball around the screen with a tracked pen.
Post MVP_V3: Tracking using motion

Technical Description: The program takes in a feed of frames(the CV program may frameskip if the computer is too slow to run the program at the feed). Each frame is turned into its basic form of a 2d array where each element of the outer arrays is a byte array which contains the 3 color values, red, blue, green. When the mouse is clicked, it selects the pixel at that mouse position. it then begins to accumulate surrounding pixels to construct the object. It finds the object's pixels using a color threshold(3d color distance in the r g and b axis) and adjusts contrast to isolate the image into a black and white image(for CV tracking, the user still sees the original frame) and enters this collection of pixels into a tracked object class. This class' draw method will draw a polygon outline to the screen on the current frame using the edge pixels to show where the computer "sees" the object. The program then pushes this frame onto a deque and a new frame is taken in. It finds the object in the new frame using relative positioning and the color of the object. If the object isn't found, it will pop off the next frame to see if that provides better information on the object. This strategy can be used to show a motion only view of the screen as well. When the object goes off the screen, it will no longer push frames, but instead will search the frame for the object. It will have a timeout period so it isn't performing useless matrix searches since you will need to check the entire frame because the object could come in from anywhere. Once the timeout period occurs, you will have to select the object again or select a new object.

Limitations: The setting in which you run the program heavily influences the ability of the program. The color of the object relative to the background and lighting are the two biggest factors. Additionally, as for right now we will assume that the object will only be of one color unless we have time for image smoothing and advanced techniques. The speed of the computer greatly affects how quickly each frame is executed which may make the user experience poor since a low frame rate is not aesthetically pleasing.

User Experience: Must know when the object is being best tracked during calibration. Must know how to run Java program in terminal.

Classes:

- CCV: Driver class for the program. Constructs the TrackedObject, Camera, and Frame objects.
- Camera: Uses the Camera Library included with Processing in the libraries folder and converts it into a (ie a 2D row-column array made of Byte[]s (in reality it's a 3D array, but as 2D picture it's easier to think 2 dimensionally with the 3rd row being for a property)
- Video: Serves the same functionality as using a camera feed, but sequentially traverses the frames of the pre-recorded video.
- CVMath: In case of excessive use of costly math functions, will implement near accurate approximations (ie trig functions, sqrts). Will contain shortcuts for operations we find to be used a lot.
- TrackedObject: Composed of pixels that make up the object on the frame. Will have a draw method that displays an outline of the object on the screen using the extreme values of the array.
- Frame: Contains a 2D array made of byte[]s, with indices 0-2 standing for red, green, blue.
- <u>For post MVP</u>: Ball: Contains Ball object that the user can manipulate on screen with the object tracked.

Term Concepts Used:

- Data Structures
    - Linked Lists:
        - Deque
            - (psuedo) Stack: push frames of the object tracked (for comparisons, not for displaying)
            - Remove from the bottom to maintain a certain amount of frames so we don't take up too much memory
    - ArrayList: used to hold the points of the object tracked.
    - Trees:

        - to create an object that encompasses the color of the object tracked.
        - To divide the screen into quadrants to better search for object pixels (since we're working with 2d arrays, likely quadtrees)
    - Sorting: Sorting object pixels to find the extremes on each row of pixels. Most likely using the partition algorithm or heaps.
- Matrix Searching: Traversing a matrix to identify object points based on their colors
- Recursive Backtracking: Once part of the object has been located, we can use recursive backtracking to search around that part until we have fully found the borders of it.