SOFTWARE TASK

DEADLINE: 31-01-2024

TOPICS

1) Getting started using OpenCV
    a) Loading images of different formats
    b) Displaying images
    c) Basic filtering operations
    d) Saving images
    e) Color Space (cover RGB<->BGR, HSV)
2) OpenCV in Python and NumPy
3) Digression: working with NumPy arrays
4) Essential Operations
    a) Reading and Editing Pixel Values
    b) Retrieving and understanding image dimensions
    c) Working with Regions-of-Interest
    d) Channels: Splitting and merging
    e) Adding, subtracting, and blending images
    f) Overview of mathematical tools (FFT, SVD/PCA)
5) Filtering and morphological operations
    a) Low Pass/smoothing filters
    b) High Pass/edge-detection filters
    c) Adaptive thresholding
    d) Erosion/dilation
    e) Floodfilling
6) GUI Features
    a) Creating windows
    b) Getting keyboard input
    c) Using the mouse
7) Inpainting demo
8) Template matching
9) Image segmentation
    a) Watershed algorithm
    b) GrabCut
10) Image transformations
    a) Translation/rotation/scaling
    b) Affine and perspective transforms
11) Features
    a) Finding lines and circles with the Hough Transform
    b) Corner detections (ie Harris)
    c) Advanced features: SURF/SIFT, BRIEF/ORB, HoG
12) Demo: feature-matching to compute homography between images

13) Working with video
    a) Loading videos of various format
    b) Playing/controlling videos
    c) Extracting frames
    d) Saving videos
14) Video analysis and object tracking
    a) Meanshift and Camshift tracking
    b) Background subtraction
    c) Kalman filtering
    d) Optical flow

RESOURCES

DOCUMENTATION
https://docs.opencv.org/3.1.0/d6/d00/tutorial_py_root.html#gsc.tab=0
YouTube
https://youtu.be/WQeoO7MI0Bs?si=YMaTXlRvrB8FOvYp
https://youtube.com/playlist?list=PLS1QulWo1RIa7D1O6skqDQ-JZ1GGHKK-K&si=sKwI7kNXkx8jDe1p

SUBMISSION

Provide the program's drive link and a video demonstrating the program in action within the following Google Form link: https://forms.gle/r3WJTMFfWiGNg3CD8

Task #1: Color Detection (EASY)

Time required: 2-3 hours
Prerequisite knowledge: Basic Python, OpenCV
Problem statement: Develop a program that detects and highlights specific colors in an image or video using OpenCV. The application should allow users to specify the target color(s) and visualize the detection.
Evaluation Criteria: Accuracy in color detection, flexibility in choosing target colors, and efficiency of the implementation.
Submission format: Submit the Python code and a video showcasing the color detection on various images or video clips.

Task #2: RGB to Grayscale Video (EASY)

Time required: 2-3 hours
Prerequisite knowledge: Basic Python, OpenCV
Problem statement: Develop a program using OpenCV to convert an RGB video to grayscale. The application should efficiently process each frame of the video, producing a grayscale output.
Evaluation Criteria: Accuracy in color conversion, efficiency in video processing, and clarity in presenting the results.
Submission format: Submit the Python code and a video showcasing the RGB to grayscale conversion on different video clips.

Task #3: Add Subtitle to Video (EASY)

Time required: 2-3 hours
Prerequisite knowledge: Basic Python, OpenCV
Problem statement: Implement a program using OpenCV to add subtitles to a video. The application should allow users to input text, specify the position and appearance of subtitles, and overlay them on the video.
Evaluation Criteria: Accuracy in subtitle placement, flexibility in text customization, and efficiency of the implementation.
Submission format: Share the Python code and a video demonstrating the addition of subtitles to different video clips.

Task #4: Shape Detection (MODERATE)

Time required: 3-5 hours
Prerequisite knowledge: Basic Python, OpenCV
Problem statement: Implement a program that detects and outlines different shapes (e.g., circles, rectangles, triangles) in an image or video using OpenCV. The application should be able to distinguish between various shapes.
Evaluation Criteria: Accuracy in shape detection, robustness to different shapes, and clarity in presenting the results.
Submission format: Share the Python code and a video demonstrating the shape detection on various images or video clips.

Task #5: Barcode and QR Code Scanner (MODERATE)

Time required: 4-6 hours
Prerequisite knowledge: Basic Python, OpenCV, pyzbar library
Problem statement: Develop a program that utilizes the pyzbar library and OpenCV to scan and decode barcodes and QR codes from images or video streams. The application should be able to accurately detect and decode codes, providing relevant information or output.
Evaluation Criteria: Accuracy of code detection, efficiency in decoding, and user-friendly interface (if applicable).
Submission format: Submit the Python code along with a demonstration video showing the scanner in action. Ensure the video includes scenarios with different types of barcodes and QR codes.

Task #6: Angle Detection (MODERATE)

Time required: 3-5 hours
Prerequisite knowledge: Basic Python, OpenCV
Problem statement: Implement a program that detects and measures angles in an image or video using OpenCV. The application should highlight or output the angles found in the given input.
Evaluation Criteria: Accuracy of angle detection, robustness to different scenarios, and clarity in presenting the results.
Submission format: Share the Python code and a video demonstrating the angle detection on various images or video clips.

Task #7: Document Scanner (DIFFICULT)

Time required: 6-8 hours
Prerequisite knowledge: Advanced Python, OpenCV
Problem statement: Create a document scanner using OpenCV that processes an input image containing a document, extracts the document, and rectifies it to produce a clear, straightened output.
Evaluation Criteria: Accuracy in document extraction, quality of the rectified output, and efficiency of the implementation.
Submission format: Provide the Python code and a video demonstrating the document scanning process on different documents.

Task #8: Lane Finder (DIFFICULT)

Time required: 5-7 hours
Prerequisite knowledge: Advanced Python, OpenCV
Problem statement: Develop a program using OpenCV to detect and highlight lanes in a video recording of a road. The detected lanes should be clearly outlined to aid navigation.
Evaluation Criteria: Accuracy in lane detection, robustness to different road conditions, and clarity in presenting the results.
Submission format: Provide the Python code and a video demonstrating the lane detection on various road scenarios.

Task #9: Car/Ball Counter (DIFFICULT)

Time required: 4-6 hours
Prerequisite knowledge: Basic Python, OpenCV
Problem statement: Implement a program that counts the number of cars or balls in a video using OpenCV. The application should be able to accurately count objects of interest.
Evaluation Criteria: Accuracy in object counting, robustness to different scenarios, and efficiency of the implementation.
Submission format: Share the Python code and a video demonstrating the object counting on various video clips.

Task #10: OMR Grading (DIFFICULT)

Time required: 6-8 hours
Prerequisite knowledge: Advanced Python, OpenCV
Problem statement: Create a program using OpenCV to process and grade Optical Mark Recognition (OMR) sheets. The application should be able to detect and analyze marked answers, providing a grading summary.
Evaluation Criteria: Accuracy in OMR sheet processing, reliability in grading, and efficiency of the implementation.
Submission format: Provide the Python code and a video demonstrating the OMR grading process on different OMR sheets.
