

COSC-5207EL02: Assignment 2 Report

Group Number: 8

Group Member:



NAME	STUDENT#	EMAIL
Haoliang Sheng	0441916	hsheng@laurentian.ca
Songpu Cai	0441024	scai1@laurentian.ca

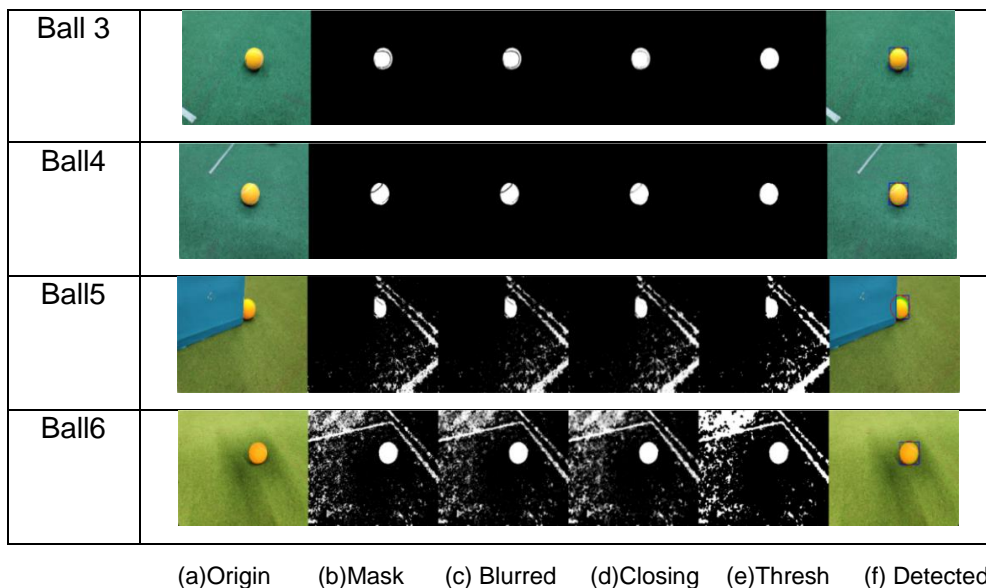
Introduction

In this report, we explore the innovative application of computer vision techniques for tracking objects and movements within videos, specifically focusing on ball tracking and arrows tracking. Utilizing Python programming, we have employed distinct scripts for each tracking task to ensure precision and efficiency. For ball tracking, the script **BallTracking.py** was meticulously developed to capture and follow the ball's trajectory across frames. Similarly, **ArrowsTracking.py** has been tailored to identify and track the directional flow of arrows, highlighting the versatility of computer vision in interpreting dynamic objects and symbols. Our approach demonstrates the potential of specialized algorithms in transforming raw video data into actionable insights. For a more comprehensive understanding, we invite readers to view our detailed demonstration on YouTube: <https://www.youtube.com/watch?v=vzsT-hkKb0s>, where we delve into the methodologies and results of our project.

Ball Tracking

1. Image pipeline

Ball1	
Ball 2	

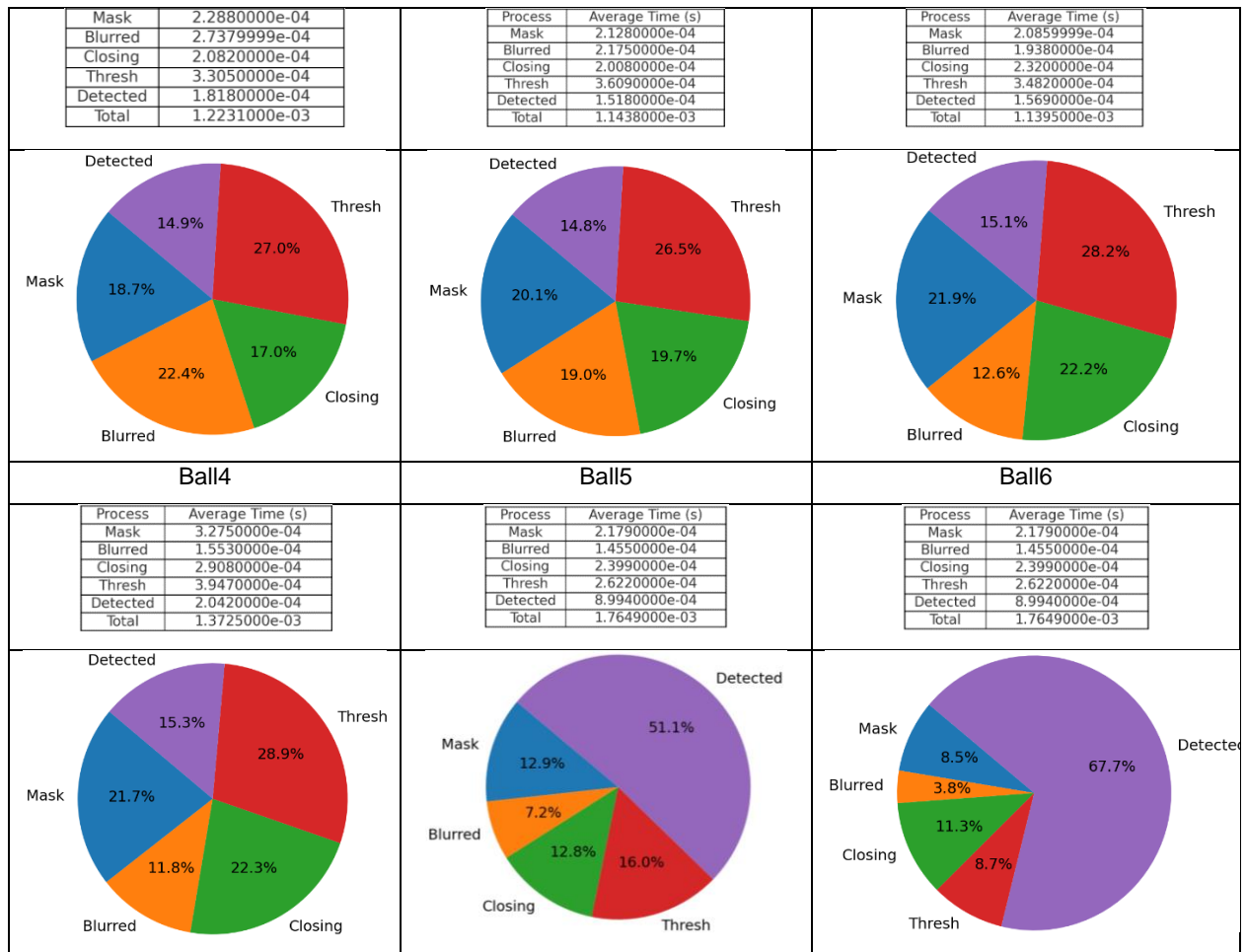


Using **BallTracking.py** to process the ball tracking task through picture and webcam.

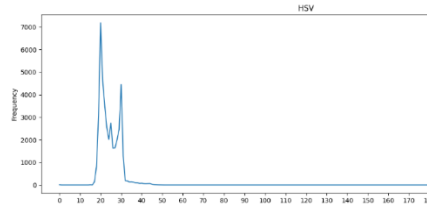
- **Mask:** In image processing, masks define areas to be processed or ignored. For tennis ball recognition, a mask created through color filtering includes only pixels matching the tennis ball's color, aiding in its isolation.
- **Blurred:** Blurring reduces an image's high-frequency content, diminishing details and noise. In tennis ball recognition, this helps minimize background distractions, making the tennis ball's outline more distinct.
- **Closing:** Morphological closing, involving dilation followed by erosion, helps seal small holes or gaps within foreground objects. For tennis ball recognition, it fills in minor gaps within the ball due to uneven colors or shadows.
- **Thresh:** Thresholding converts an image into a binary format based on pixel values. In tennis ball recognition, setting an appropriate threshold isolates the tennis ball by turning its pixels white and all others black.
- **Detected:** The final step where the tennis ball is identified and marked, possibly using contour detection, Hough Circle Transform, or other image recognition techniques.

2. Processing time and Time efficiency

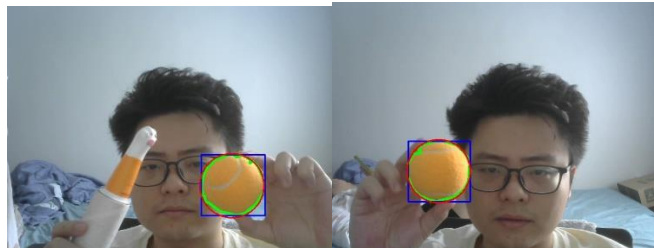
Ball1	Ball2	Ball3
-------	-------	-------



Analyzing the temporal distribution and the image processing pipeline, it becomes evident that a significant portion of the processing time is allocated to the final two balls' images during the search process, resulting in extended durations. A comparative examination of the image processing workflows for each image reveals that the underlying cause of this phenomenon is the insufficient background filtration following the initial HSV (Hue, Saturation, Value) conversion, which subsequently necessitates increased processing time in the subsequent stages, particularly in the final contour detection phase. The objective factors contributing to this issue in the last two images are the background (ground) HSV values, which closely resemble those of the balls, compounded by the presence of shadows and light spots, leading to interference.



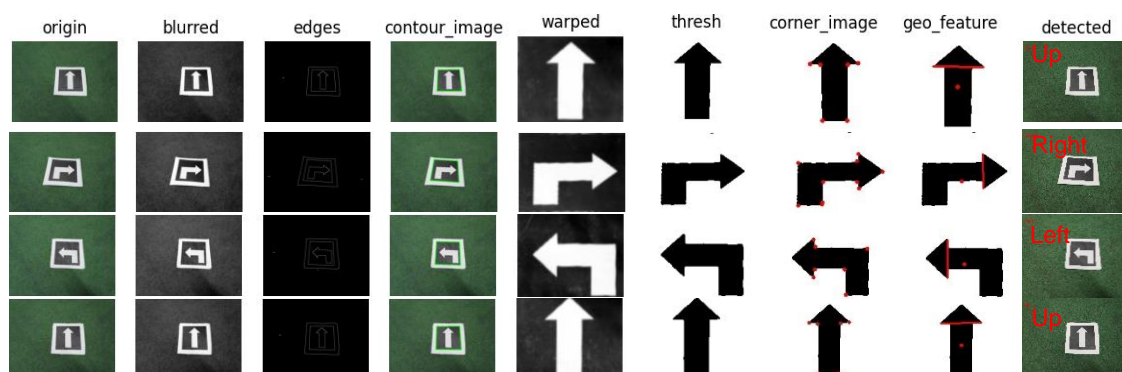
3. Repeatability tests and Real-time performance



- **Laboratory Setting:** In a controlled environment with consistent lighting, the system accurately identified tennis balls in 15 consecutive trials, demonstrating the algorithm's stability across repetitive tasks.
- **Diverse Backgrounds:** Testing across various backgrounds, including tennis courts with different surface colors and textures, the system maintained a high detection rate, showcasing its adaptability and repeatability in changing environments.

Arrow Tracking

1. Image pipeline

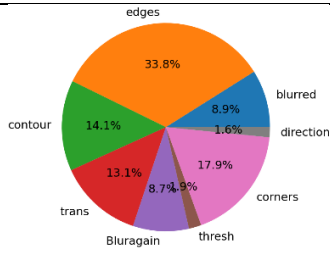
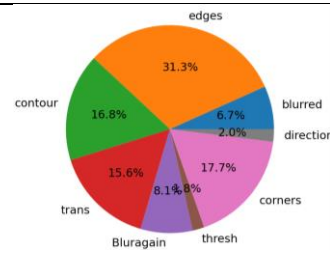
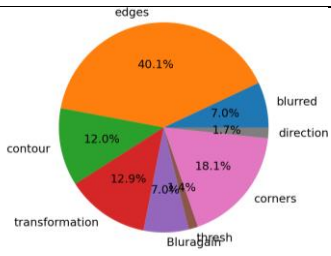
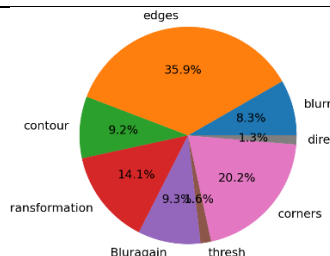


Blur again here

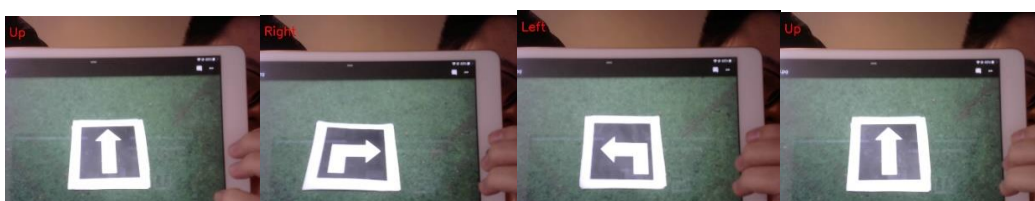
Using **ArrowsTracking.py** to process the arrow direction determination task through picture and webcam.

- **Principle:** Leveraging OpenCV's capabilities, the process begins with corner detection to identify the keypoints within the image. Subsequently, intersections that are six pixels apart are connected to form a line through their coordinates. The direction (up, down, left, or right) is then determined by analyzing the angle formed by a line drawn from the centroid to this line.
- **Warped:** This step involves a matrix transformation to alter the coordinate system, thereby converting the image into the desired planar representation. This facilitates more precise and rapid detection operations in subsequent stages, leveraging OpenCV's capabilities.
- **Blurred:** Applying blurring twice is aimed at reducing noise and artifacts, thereby facilitating smoother edge detection and enhancing focus detection convenience.

2. Processing time and Time efficiency

UP			Right		
Process	Average Time (s)		Process	Average Time (s)	
Blurred	1.2610000e-04		Blurred	9.5299998e-05	
Edges	4.7680001e-04		Edges	4.4480000e-04	
Contour	1.9840001e-04		Contour	2.3830000e-04	
Transformation	1.8469999e-04		Transformation	2.2170000e-04	
Blur Again	1.2260000e-04		Blur Again	1.1450000e-04	
Threshold	2.7300004e-05		Threshold	2.5999994e-05	
Corners	2.5310001e-04		Corners	2.5130001e-04	
Direction	2.2000007e-05		Direction	2.7799993e-05	
Total	1.4110000e-03		Total	1.4197000e-03	
Left			UP2		
Process	Average Time (s)		Process	Average Time (s)	
Blurred	9.6199990e-05		Blurred	1.3830001e-04	
Edges	5.5230000e-04		Edges	5.9820000e-04	
Contour	1.6520001e-04		Contour	1.5310000e-04	
Transformation	1.7710000e-04		Transformation	2.3530000e-04	
Blur Again	9.6100004e-05		Blur Again	1.5540000e-04	
Threshold	1.9499988e-05		Threshold	2.6200010e-05	
Corners	2.4890000e-04		Corners	3.3729999e-04	
Direction	2.2799999e-05		Direction	2.2300010e-05	
Total	1.3781000e-03		Total	1.6661000e-03	

3. Repeatability tests and Real-time performance



- **Controlled Environment Tests:** Within a laboratory setting under consistent lighting conditions, the system was able to accurately detect the direction of arrows in 10 sequential images, showcasing its reliability and precision in repetitive tasks.
- **Varied Backgrounds and Orientations:** The algorithm was tested against a diverse array of backgrounds and arrow orientations, maintaining a high detection accuracy. This underscores the system's adaptability and consistent performance, even when faced with complex scenarios.