University of Missouri - Columbia Department of Electrical Engineering and Computer Science

CS 7650 Digital Image Processing - FS2021  
Final Project Report

Object Detection and Tracking Using Background Subtraction and Connected Component Labeling

Kaitlyn Zahn

December 14, 2021

Supervised by Prof. Kannappan Palaniappan & Feliz Bunyak

TAs: Imad Toubal

Abstract

As the digital world becomes more and more prevalent, the need for object detection and tracking only grows. Object tracking can be used for security purposes, intelligence gathering, combating crime, and many individual needs.

This project implements a solution to this problem by taking a video input and implementing object detection and tracking using background subtraction and connected component labeling. This project is based off the “Object Detection and Tracking using Background Subtraction and Connected Component Labeling” research by Abdul Malik in 2013 [1].

The goals of this project are discussed in detail in the ‘Image Processing Tasks and Goals’ section of this report. More details on the implementation of this program, including the chronological steps (including handling video, background subtraction, connected component labeling, object selection, component masking, and dynamically drawing on an image) follow in the ‘Algorithm and Implementation Details’ section of this report. This section also includes code segments. The results of this program are discussed in detail later in the report, found in the ‘Results: Demo and Evaluation’ section. Finally, the report is closed by reflecting on the project and looking forward on extending this solution in the ‘Summary and Future Work’ section of this report. Resources used for this program can be found in the ‘Bibliography’ at the end of the report.

Overall, this program is unable to process every frame from a video efficiently, but identifies and tracks objects as well as built in OpenCV trackers do for the frames selected.

Introduction: Motivation and Background

This project is a study of background subtraction and connected component labeling to implement object detection and tracking. This project aimed to detect objects in different frames of a video and detect whether there was movement between the two frames. The main objectives of this project were to successfully implement background subtraction on frames of a video, apply connected component labeling to that same frame, and compare the identified objects between frames to identify movement.

This is an important problem to solve because of the increasing need for object tracking in this digital age. The world is increasingly more digital and object tracking can provide resources for security, data collection, and can help prevent and fight crime. The researcher implemented this problem because of increasing interest in connected component labeling and the possibility of working with video footage rather than still images.

Solutions like this research can be seen in security at malls, airports, and homes (including Ring Doorbell). Another primary use is in stores where object tracking can help store owners gain a better understanding of their customer’s spending habits.

The goal of object detection and tracking was implemented by taking an input video, freezing specific frames from the video, and implementing background subtraction on each of those frames. The background subtracted image is then inputted into a connected component labeling algorithm that identifies objects as a whole rather than as individual pixels. The program selects the largest object identified by connected component labeling, uses component masking to remove objects besides that which is being tracked, and draws a rectangle around the tracked object. The selected frames are then displayed in chronological order with a rectangle that tracks the identified object through each frame of the originally inputted video. This program was tested on Ring Doorbell footage [8].

Image Processing Tasks and Goals

The goal of this program was to detect objects and track their movement using background subtraction and connected component labeling. The expected output from any given video input is a series of images with the background subtracted and the largest object from the foreground selected and tracked over the series of output images.

For the program, the researcher studied how to implement background subtraction by hand but was unsuccessful in finding methods that did not use built in methods [5]. The researcher implemented connected component labeling by hand. Innovative methods in the project include working with video files rather than images, subtracting the background, the selection of the object, component masking, and dynamically drawing the rectangle on the selected object.

The researcher used video input, so files are not attached to this report, but freeze frames from each video can be seen in the ‘Results: Demo and Evaluation’ section of this report.

The researcher had to study a lot about working with video inputs, background subtraction, connected component labeling by hand and the built-in functions, component masking, and drawing on an image dynamically. This is all discussed in further detail in the next section of this report, ‘Algorithm and Implementation Details’.

Algorithm and Implementation Details

This program was implemented using Python 3.9.7.

For a more specific solution, the researcher tested this implementation using Ring doorbell footage [8]. This allowed for unmoving cameras, relatively unchanging backgrounds, and lack of changing shadows/lighting. This also allowed the researcher to focus on single object tracking rather than venturing into the field of multiple image tracking.

*Diagram

Description automatically generated*The figure below illustrates the program’s pipeline in making Object Detection and Tracking possible. As you can see, a video input is taken, frames from that input are frozen and background subtraction is applied to those frames. To the frames with a background subtracted, connected component labeling is applied. Then, the object is selected from the connected component labeling image, component masking is applied, and the object to be tracked is circled by drawing a rectangle around it.

Figure 3.1 *Program Pipeline*

Below is a breakdown of the main implementations of each part of this pipeline including segments of code and important information about the implementation of each of these.

*Video Input & Frame Freezing*

OpenCV’s built in functions for capturing a video made working with .mov files rather than image files relatively easy. The researcher used OpenCV’s function ‘cv2.VideoCapture()’ to capture the video [7]. This function ensures every frame in the video is successfully read in. A while loop was used to process through each frame. Every 50th frame is saved to the device and then pushed through the rest of the pipeline [7].

*Background Subtraction*

The researcher looked for ways to implement background subtraction by hand but was unable to find any implementation that did not use OpenCV’s built in functions. This was implemented using ‘cv2.createBackgroundSubtractorMOG2()’ [5]. This mask is applied on each of the saved frames and then sent to the connected component labeling functions.

Text

Description automatically generated

Figure 3.2 *Video Capturing and Background Subtraction Code*

*Connected Component Labeling*

The researcher had previously implemented connected component labeling using C++ and transformed that code into python for this program. The labelRegion() function uses the unionFunction() and findFunction() to correctly label every pixel in the image. Then, the labelColorize() and unique() functions correctly assign a color to each object identified in the image. Finally, the shapeProperties() function calculates the area of each object in the image.

Upon further research, the user found the ‘cv2.connectedComponentWithStats()’ function from OpenCV [2], [4]. This function allowed the program to run more efficiently and returned the centroid of each object. This proved useful in the object selection. Thus, while connected component labeling is implemented by hand, the built-in function was used for efficiency.

*Object Selection, Masking, and Circling*

The object to track was selected based on the object that returns with the greatest area (not including the background of the image). Everything besides the selected image was then masked and the object was circled by drawing a rectangle around it using OpenCV’s ‘cv2.rectangle()’ function [6]. This was all implemented using the statistics given by the connected component labeling function.

Text

Description automatically generated

Text

Description automatically generatedFigure 3.3.1 *Object Selection Using Greatest Area Code*

Figure 3.3.2 *Component Masking and Object Circling Code*

Results: Demo and Evaluation

The program performed relatively well on the given test dataset. While lighting and singular objects not seeming connected (impacting the Connected Component Labeling results) resulted in some imperfect object selections, in general the program was able to detect when an object came into the frame and track its movement over time.

The below figures display input, intermediate, and output images as a result of implementing this program on a video. In each figure, each row represents a different frame that was evaluated. These do not represent every frame evaluated per video. The leftmost image in each row is the original frame extracted from the video. The following image is the background subtracted image. As you can see, lighting impacts Background Subtraction and sometimes parts of the background are not subtracted as a result. The third image in each row is after Connected Component Labeling is implemented. Shadows and lighting impact how some objects are identified. The final image in each row is after the correct object is selected, component masking Background pattern

Description automatically generatedis applied, and a rectangle is drawn around the centroid of the given object. Results below.

Figure 1.1 *Program with Ring1.mov*

Graphical user interface, application

Description automatically generated

Figure 1.2 *Program with Ring2.mov*

A picture containing background pattern

Description automatically generated

Figure 1.3 *Program with Ring3.mov*

Since the publication of the research article that is the basis of the researcher’s program, OpenCV has implemented trackers as well [1], [3]. These trackers allow the user to draw a box around the object to track and tracking the video is the output. Results from those trackers are displayed below.

A picture containing text, different

Description automatically generatedFigure 2.1 *OpenCV Tracker with Ring1.mov*

A picture containing text, outdoor

Description automatically generatedFigure 2.2 *OpenCV Tracker with Ring2.mov*

Summary and Future Work

Overall, the researcher is content with the implementation and results of the project. The original goal of being able to detect whether movement exists between frames was exceeded as the researcher was able to track more specifically a certain object between frames.

Some problems experienced when implementing this program include the question of how many frames to evaluate from a given video, what threshold to use universally when converting from an RGB image to a grayscale to a binary image, and implementing connected component labeling by hand in Python.

The researcher learned a lot while implementing this, but one of the greatest lessons is the importance of research and pre-planning that goes into implementing a solution like this. Much of the time spent implementing this project consisted of research, trying things that didn’t work or didn’t work well, and finding a better solution. Putting in the work on the front end was extensive, but necessary. Some other lessons learned include using OpenCV on videos, the importance and relevance of background subtraction and component masking, and the importance of lighting, shadows, and noise in digital image processing.

In the future, the researcher like to be able to implement this method on an entire video without isolating specific frames and working on still frames. The researcher would also like the program to be able to identify an object and know whether it’s the same or a different object from a previous frame. The researcher would love to be able to explore Deep Learning methods that contribute to object tracking. With this updated technology, tracking methods exist that are much more consistent than what the researcher implemented. Finally, some of the noise that exists because of lighting in a background subtracted image could have been eliminated using adaptive median filtering and the researcher will implement this in the future to achieve more efficient results.

Bibliography

[1] AbdulMalik, A., Khalil, A., & Ullah Khan, H. (2013). Object detection and tracking using background subtraction and connected component labeling. *International Journal of Computer Applications*, *75*(13), 1–5. https://doi.org/10.5120/13168-0421

[2] Joshi, Y. (2020, April 28). *Connected component labeling: Algorithm and python implementation*. OpenGenus IQ: Computing Expertise & Legacy. Retrieved December 15, 2021, from https://iq.opengenus.org/connected-component-labeling/

[3] Martinez, J. C. (2021, October 11). *Object tracking with opencv*. LCS - Learn and grow with us. Retrieved December 15, 2021, from https://livecodestream.dev/post/object-tracking-with-opencv/

[4] *OpenCV connected component labeling and analysis*. PyImageSearch. (2021, April 17). Retrieved December 13, 2021, from https://www.pyimagesearch.com/2021/02/22/opencv-connected-component-labeling-and-analysis/

[5] *Python: Background subtraction using opencv*. GeeksforGeeks. (2021, July 6). Retrieved December 13, 2021, from https://www.geeksforgeeks.org/python-background-subtraction-using-opencv/

[6] *Python opencv: Cv2.rectangle() method*. GeeksforGeeks. (2019, August 13). Retrieved December 15, 2021, from https://www.geeksforgeeks.org/python-opencv-cv2-rectangle-method/

[7] *Python: Program to extract frames using opencv*. GeeksforGeeks. (2018, May 15). Retrieved December 15, 2021, from https://www.geeksforgeeks.org/python-program-extract-frames-using-opencv/

[8] *"Ring alarm helped save our lives"*. Ring. (n.d.). Retrieved December 15, 2021, from https://tv.ring.com/detail/videos/