Automating the Collection of Bird Data from Video Footage

ECE/SENG 499

Summer 2018

University of Victoria

Group ID: 3

Faculty supervisor: Dr. Alexandra Branzan Albu

Team Information:

S. No.	Name
1	Colby Timm
2	Tim Salomonsson
3	Cyrus Erfani
4	Sam Taylor

Acknowledgment

Foremost, we would like to express our gratitude to our project supervisor, Dr. Alexandra Branzan Albu, for her continuous support of our project. We would also like to acknowledge the department and technical staff as well as the chairman, Dr. Michael McGuire. Our thanks also goes out to our course teaching assistant, Philip Alipour, and our course instructor, Dr. T. Ilamparithi. Last, but certainly not least, we would like to thank our family and friends for their support throughout our education.

Executive Summary

The chimney swift is a bird that is designated as a threatened species in Canada. These birds nest and roost in cave walls but now also use man-made chimneys for this purpose. Initially, the bird population grew with the expansion of urban settlements. Recently, however, the bird population has declined.

To track the chimney swift population size, the Ontario government installed high-definition cameras to monitor two large communal chimneys in Sault Ste Marie, Ontario. The camera footage is manually examined to estimate the number of birds entering the chimneys.

Given that counting birds by hand is exhausting and tedious work, software automation would greatly reduce the time required to count the birds, thereby resulting in decreased cost and increased productivity. Furthermore, automation would remove the potential for a human counter to make errors.

Computer vision techniques implemented in Python and OpenCV were employed to detect and track moving swifts in recorded video footage. The future path of a swift is predicted based on the swift's current and previous position. The predicted path is used to determine if the swift will enter the chimney.

The bird tracking process starts with background subtraction. This technique produces an image which contains only moving objects in a frame. The objects contours within the resulting image are detected, and a tracker is applied to each new contour that does not already have a tracker. The swifts are continuously tracked until they enter the chimney. A performant bird counting application with a GUI is built using this bird tracking algorithm.

The proposed algorithm was found to be successful in most cases, but struggles to provide accurate results under high-stress situations when there are many birds to track simultaneously with the birds overlapping in the video footage.

Table of Contents

List of Figures

List of Tables

Glossary

CV **Computer Vision**
The field of computing that deals with making computers understand visual input

IR **Infrared**
Electromagnetic radiation which has a wavelength in the infrared range.

AI **Artificial Intelligence**
The field of computing that deals with making software which is capable of human-like intelligence.

CSV **Comma Separated Values**
A filetype wherein each text line consists of data separated by commas.

GUI **Graphical User Interface**
A method of interfacing with computers using windows, menus, icons and other graphical elements.

I Introduction

The chimney swift, a bird native to Ontario, has been designated as a threatened species by the province of Ontario as of September, 2009 [1]. These birds naturally nest and roost on cave walls and in hollow trees but, following the emergence of the man-made chimney, have taken on the behaviour of utilizing open-ended chimneys for this purpose. If steps are not taken to address the declining chimney swift population, the province of Ontario predicts that the chimney swift population will become an endangered species [1].

The chimney swift population is tracked via video feed. Several cameras are installed at two nesting sites in Sault Ste Marie, Ontario. Up to 2600 swifts have been observed using the chimneys during spring migration [2]. To monitor swifts at each location, two high-definition colour and IR supporting cameras with a resolution of 1920x1080 and a frame rate of 15fps or 30fps have been set up to capture the birds. One camera is installed inside the chimney, with the second camera installed at a location in the distance to monitor the chimney exterior. Cameras are operational in light and low-light conditions from May 1 to August 31, which corresponds to the chimney swift migration season.

The current implementation of counting swifts is achieved by individuals manually counting each bird. The proposed objective of this project is to eliminate the reliance on manual counting and replace it with an automated system, utilizing a computer vision approach. The completed approach writes the bird count to a CSV file with a timestamp in hours:minutes:seconds, mimicking the spreadsheets used by Algoma SwiftWatch -- a nonprofit organization that tracks chimney swifts in Sault Ste Marie. The software will save Algoma a considerable amount of time by reducing or eliminating manual counting.

II Project Goal

The goal of the project is to monitor the chimney swift population by counting those entering a chimney. This is accomplished by creating an application that analyzes camera footage and records the time when a swift enters the chimney along with an accumulated count of how many have entered. This reduces the amount of manual work required to count the birds.

III Design Objectives

The objectives can be fulfilled only if the chimney is not obstructed by extreme weather or other birds, such as a seagulls sitting on the chimney edge. Also, the video footage must have sufficient quality to identify individual birds. These are the only constraints.

The design objectives include a performance requirement: the software should at minimum process a video within the running time of the video. This requirement ensures that the computer that plays the video can also process the video in a timely manner.

The software must include a usable graphical user interface (GUI). Additionally, a non-technical user without computer vision knowledge should be able to understand and utilize the graphical elements and text within the GUI with ease.

Once a video is processed, the application must give the option to export a comma separated values (CSV) file with lines of data, including the accumulated bird count:

e.g. <file name>, hh:mm:ss, <count value>

The next section explores possible approaches to objective fulfillment from existing literature.

IV Literature Survey

The objectives require extraction of meaningful data from video footage. More specifically, the numbers of birds which enter the chimney. Thus, a computer vision (CV) approach to solving the design objectives is required. CV is an interdisciplinary field, combining machine learning, AI, image processing, and more. The process of counting the birds requires a multiple object tracker and a heuristic to count birds entering the chimney. The heuristic is context-dependent and is not found in existing literature. The multiple object tracker uses a few combined models including [3]:

- an appearance model aka. object (bird) detection,

- a motion model aka. object tracking,
- and an interaction/occlusion model.

Object detection can be accomplished with two methods: 1) background subtraction [4] with contours and 2) image thresholds [5]. Background subtraction is used to detect *moving* objects from a static camera view. Image thresholding is used to binarize an image; pixels above the threshold (colour/intensity) are black. The advantages and limitations of each object detection method are summarized in Table 1 below:

Object Detection			
Background subtraction		*Image thresholds*	
Pros	Cons	Pros	Cons
• no initialization required	• requires more processing power • weather adaption	• simple and efficient	• required threshold value • weather adaption

Table 1: Advantages and limitations of object detection methods

Both object detection methods will detect weather and lighting changes. Moving clouds are detected by background subtraction. Dark coloured clouds are detected by image thresholds. However, background subtraction does not require an initial threshold value which is usually a colour intensity value and may change from video to video. Therefore, background subtraction is more adaptable to changing lighting conditions than image thresholds.

Object tracking can be accomplished using multiple libraries including the Open Source Computer Vision Library (OpenCV) [6] and SimpleCV [7]. OpenCV is a real-time computer vision, image processing, and machine learning library. SimpleCV allows "access to several high-powered computer vision libraries such as OpenCV" [7]. However, SimpleCV only implements a subset of the trackers within OpenCV [8]. OpenCV contains numerous tracking algorithms, including Kalman [9] and MOSSE filters [10]. The advantage of OpenCV is its complete and accessible online documentation as well as its question-answer forum with a community of over 50,000 users [11].

The feasibility of creating HAAR cascades to investigated to determine machine learning could be used to detect the birds. It was determined that using HAAR cascades was infeasible since training is required to create the cascade and bird detection is unreliable due to low resolution of birds within the video. Consequently, HAAR cascades offered

no advantages over the background subtraction and were not chosen as the means of detection.

Existing Solutions

The suitability of computer vision and machine learning algorithms are reliant on the video data. The problem context creates assumptions within an implementation and controls various initialization parameters. This means one computer vision implementation likely requires modification before use in another context, even if the context is extremely similar. Two examples of different trackers are displayed in this section.

Kalman Filter Tracking

An existing multi object tracker using a Kalman filter is available within OpenCV [12] and on Github.com [13]. The tracker on Github uses background subtraction and edge detection in combination with the filter to track insects from a top-down view, as shown below:

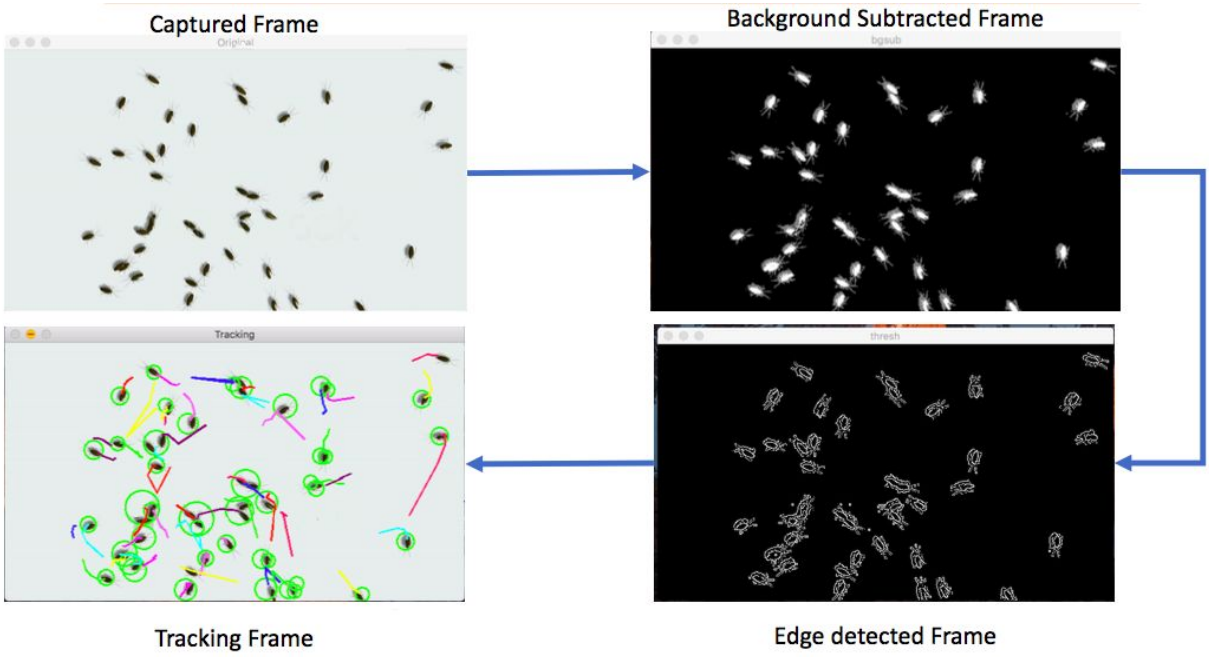

Figure 1: Kalman Filter image pipeline[13]

The Kalman filter provides continuous tracks for multiple insects in a controlled environment (see *Figure 1* - bottom left).

MOSSE Filter Tracking

OpenCV provides a Minimum Sum of Squared Error (MOSSE) filter useful for object tracking.

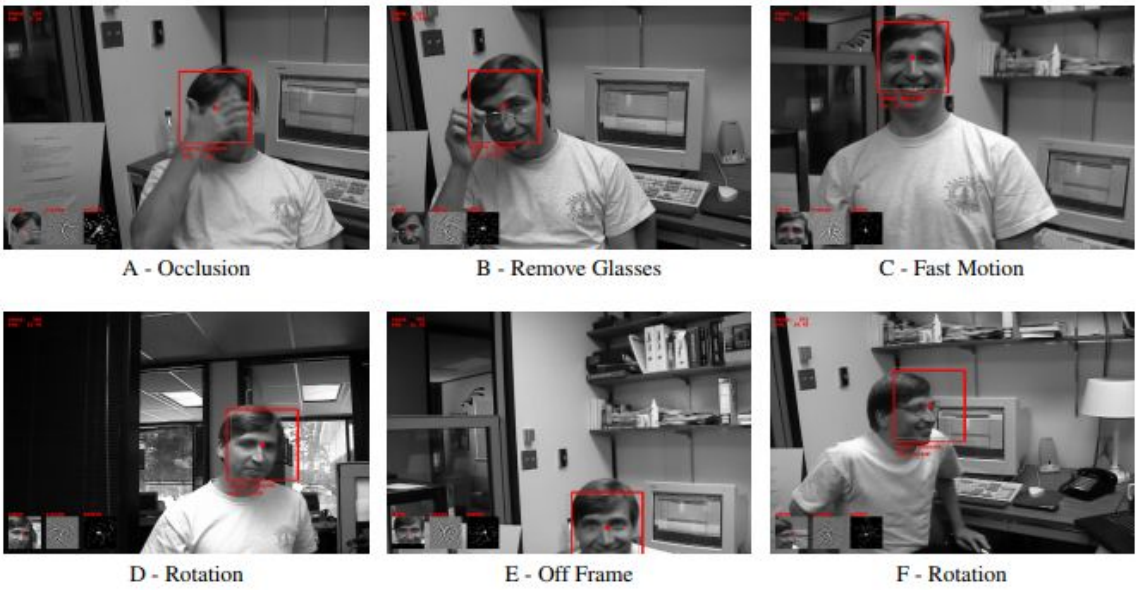

Figure 2: MOSSE filter tracking in various conditions[14]

Figure 2 above shows the MOSSE filter can track an object (i.e a face) under various conditions including occlusion (A), deformations (B), and fast motion (C).

Selected Approach

Both approaches for object detection will be used which provides greater design breadth. OpenCV is chosen as the desired computer vision library for object tracking for its size and extensive online support base. Multiple trackers will be tested and the final prototype will be refined from the most accurate design.

The selected trackers for testing include:

1. MOSSE filter
2. Kalman filter
3. no tracker; i.e. a simple bird counting heuristic without tracking

The two filters were chosen because they offer some occlusion handling. i.e both filters can reacquire occluded objects once they reappear. Occlusion commonly occurs within the bird video footage, as multiple birds enter the chimney or overlap in flight.

Figure 3: Bird occlusion

A Kalman filter uses probabilistic predictions to determine the likely location of the occluded object once it reappears. When "occlusion is detected [a MOSSE filter] enables the tracker to pause and resume where it left off when the object reappears." [14] Additionally, birds undergo non-rigid deformations during flight. This means MOSSE filters are suited to this project since "a tracker based upon MOSSE filters is robust to … non-rigid deformations" [14]. The final prototype will use the most accurate and/or most efficient tracker(s). Since background subtraction is more adaptable to changing lighting conditions, it will be used for object detection in the final prototype.

V Team Duties & Project Planning

This project was divided into two distinct phases: phase one and phase two. In phase one, each team member conducted independent research, and based on the findings, implemented a unique solution to satisfy the project goals. Ideally, research and development on the same problem by each team member will yield different approaches which can be used to determine the most effective solution. In phase two, the solutions of all team member were evaluated and compared in order to find the most accurate approach. This became the base for building the final version, which is further refined to improve usability and accuracy.

Phase One

The initial milestone for each team member is similar as each member works on a different approach to the same problem. Each team member is expected to explore different computer vision approaches to tracking swifts, which will then be used to create an implementation to count birds. Thus, one of the main deliverables is a comparison of different computer vision approaches to solving the problem. This is significant because it ensures that the final version takes the best possible approach to satisfying the project goal.

The most apparent challenge the team encountered was establishing meetings times which were acceptable for all members. This problem was resolved by creating a shared Google Calendar which contained the schedule of all team members. This simplified scheduling as the entire team could view the schedule of each member.

The next challenge was ensuring that the approaches taken by the team were sufficiently different such that a relatively broad range of possible solutions could be explored. This problem was resolved through constant communication within the team through team chat, and weekly meetings where each member provided progress updates for the rest of the team.

Phase Two

Phase two began with an evaluation of the deliverables of phase one. Each team member evaluated their solution by using the same test data, and sharing the results with the rest of the team. The average error of each solution was calculated and used to pick the most accurate solution. This step was crucial to the project, as the most accurate solution would be used as the foundation of the final version.

Once the base for the final version had been created, it was improved by features such as variable levels of erosion and dilation, and minimum and maximum contour area. Finally, a GUI was implemented and tied in with the main application. Implementing all the features of the main application as settings in the GUI and having all features work as desired proved to be much more challenging than initially expected. Some of the challenges arose because of issues involving changing the rendering from using the OpenCV library to using the PyQT [17] GUI library, reimplementing some necessary drawing functionality, and making the GUI stateful while behaving appropriately in each state. Half of the team worked on finalizing the application while the other half worked on documentation. The final deliverable was the finished application.

VI Design Methodology & Analysis

The Python programming language was chosen for this project. Python has a relatively simple syntax and is known by the entire project team. This means development can start and progress quickly; a major concern in a project with a three month deadline. OpenCV, the chosen computer vision library, is available in Python.

The selected trackers for implementation in Python are:

- MOSSE filter
- Kalman filter
- no tracker; i.e. a simple bird counting heuristic without tracking

The design of the counting heuristic involves counting the birds as they pass near the chimney. One option is to track the birds and count the birds which cross the chimney edge. A second option is to remove tracking and count birds as they cross a line or enter an area near the edge of the chimney.

The tracking counter is implemented twice with a MOSSE filter and a Kalman Filter. The non-tracking counter is implemented twice with line (*Figure 4*) and rectangle (*Figure 5*) intersection, respectively.

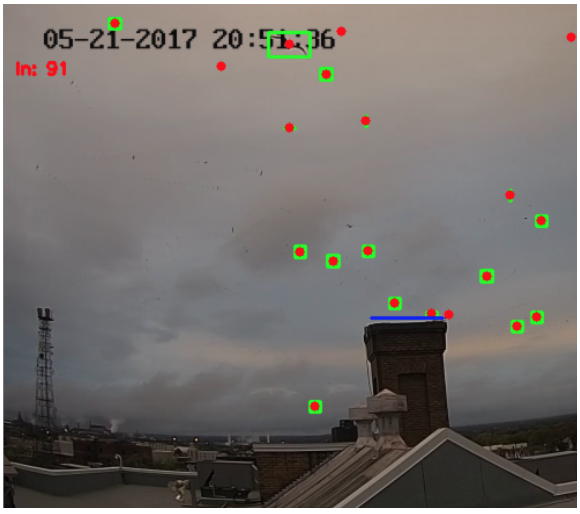

Figure 4: Count birds with line (blue) heuristic

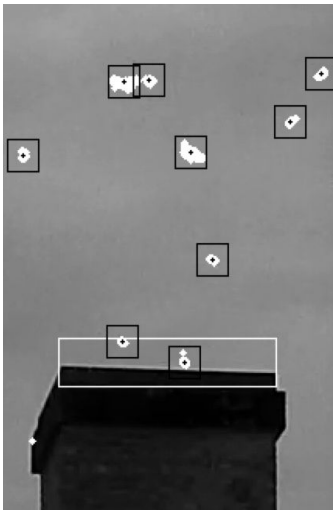

Figure 5: Count birds with rectangle heuristic

Analysis began with an evaluation of the individual solutions developed in Phase One of the project. The criteria for evaluation of the implementations consisted of 5 test samples which were representative varying conditions that a user would experience when using this application. Three of five sample videos included a flock of swifts entering the chimney, and two of five videos consisted of no birds entering the chimney.

The sample videos had varying weather and lighting conditions, as well as other variables that could impact the bird count. The weather conditions included cloudy and rainy. The lighting conditions included those seen around dusk or mid-day or night. Broadly speaking, there were both normal-light and low-light conditions in the sample videos. Two other variables that could possibly affect bird count were obstruction of the chimney ledge and infrared lighting. To evaluate the implementations under these conditions, sample videos were added which included a seagull obstructing the camera's view of the chimney ledge and another video where the lighting switches to infrared. Screenshots of various scenarios from the samples are shown in the figures below.

Figure 6: Normal lighting

Figure 7: Dusk lighting + obstruction

Figure 8: IR lighting

These videos are used to analyze the accuracy of the software implementations. The range and average error of all implementations is shown below:

Implementation	**Error Range**	**Average Bird Count Error**
MOSSE	28 - 77%	48%
Kalman	29 - 200%	163%
no tracker (line)	9 - 864%	215%
no tracker (rectangle)	35 - 266%	206%

Table 2: Analysis of design error; average bird count compared to the known count

The line heuristic has more error than the rectangle heuristic. This is because most chimney swifts fly quickly, and will not be detected intersecting the line. A rectangle allows the bird to be within the rectangle on at least a single frame. If the rectangle is too large, a single bird may be within the rectangle on multiple frames, and thus counted multiple time. In contrast, if the rectangle is too small, the bird will fly past the rectangle without being captured inside within a frame.

The analysis of the implementations show that the implementations which use tracking perform better than the implementations without tracking. As demonstrated, the MOSSE filter performs best overall, with the Kalman filter being the next best tracker.

Tracking is best accomplished with a MOSSE filter, exemplified by the lowest average error and the smallest range of error. The MOSSE filter performs best due to its support for deformable objects and occlusion [14]. Extreme occlusion occurs when a swarm of swifts enter the chimney together. The Kalman filter cannot handle the extreme occlusion as well as the MOSSE filter. As the MOSSE filter performed better than the other implementations, it was chosen as the tracker method for the final implementation for the project.

VII Final Design Details

The first aspect of the swift counter application that the user is exposed to is the user interface. This has been designed to be very user-friendly, satisfying the project objective that the application should be usable by those with limited computing experience. This has been accomplished through the use of a GUI.

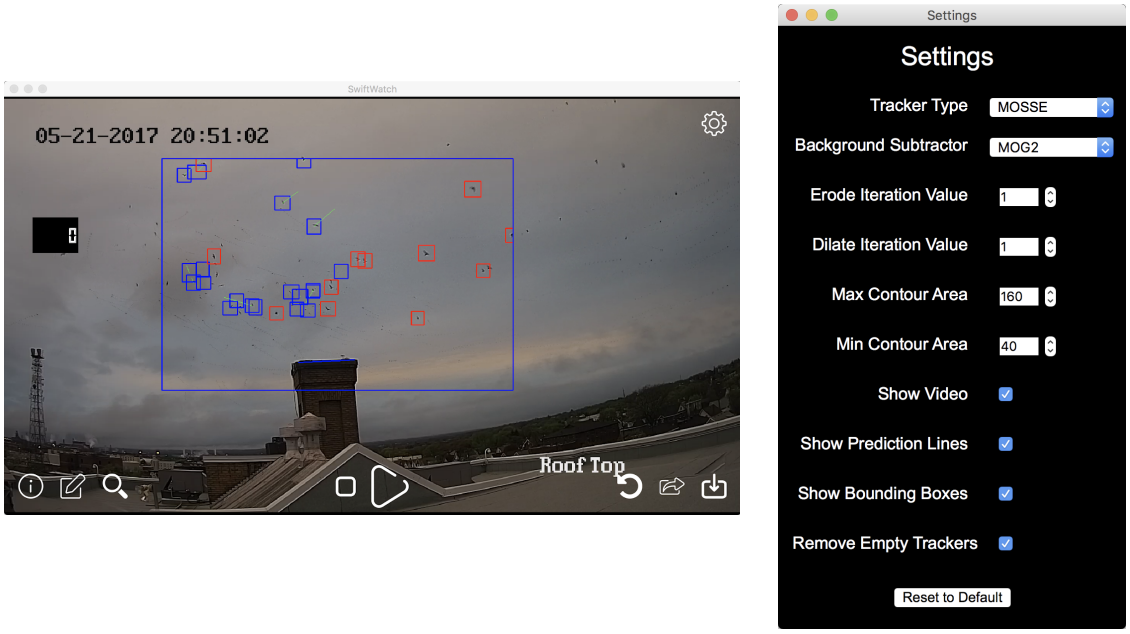

Figure 9: Main user interface (left) and settings dialog (right)

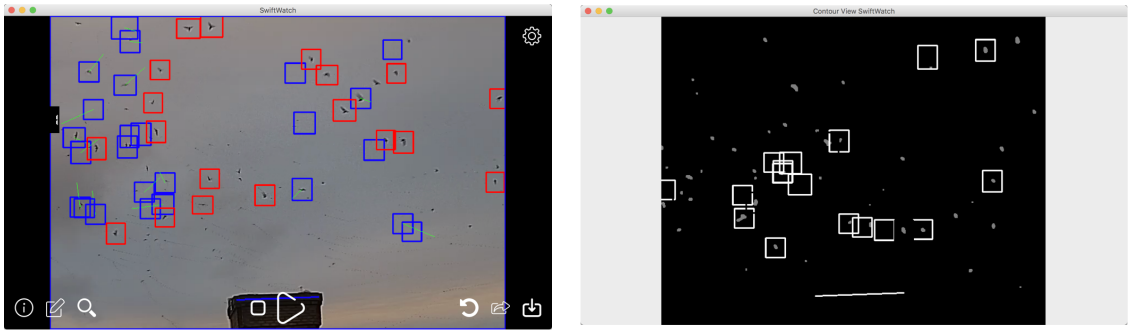

Figure 10: Zoom-in view in main user interface (left) and contour view dialog (right)

The GUI has many features which make it user-friendly. The GUI has buttons which feature standard media control symbols which are popular across various applications. For example, the gear symbol has been used as the settings button and the play symbol has been used as the play button. Tooltips help the user discover the function of a button if they are unfamiliar with the symbols used. Additionally, on-screen directions are

presented to the user to guide them through the process of using the application. The figures below show these features as seen in the GUI.

Figure 11: Tooltip directions

Figure 12: Standard settings symbol

Figure 13: On-screen

Another project objective was satisfied by making the runtime of the application less than the runtime of the video. The runtime is variable and partially depends on the number of birds in the frame. This is because more processing is required to detect and track the more birds. In videos with representative bird swarms, the runtime was measured to be roughly ⅓ of the runtime of the input video, which is well below requirement. When the number of birds of increases, the running time of the application increases to roughly ½.

The third and final objective is the output of a CSV file with the relevant data, i.e. bird counts with associated timestamps. An example is shown in the figure below.

Filename	Time	Swift Entering
ch04_20170531233754.mp4	23:39:01	2
ch04_20170531233754.mp4	23:39:59	2
ch04_20170531233754.mp4	23:40:09	2
ch04_20170531233754.mp4	23:40:15	1
ch04_20170531233754.mp4	23:40:19	1
ch04_20170531233754.mp4	23:40:37	1

Figure 14: Screenshot of example CSV output file

The final bird counting process is highlighted in the Figure 15.

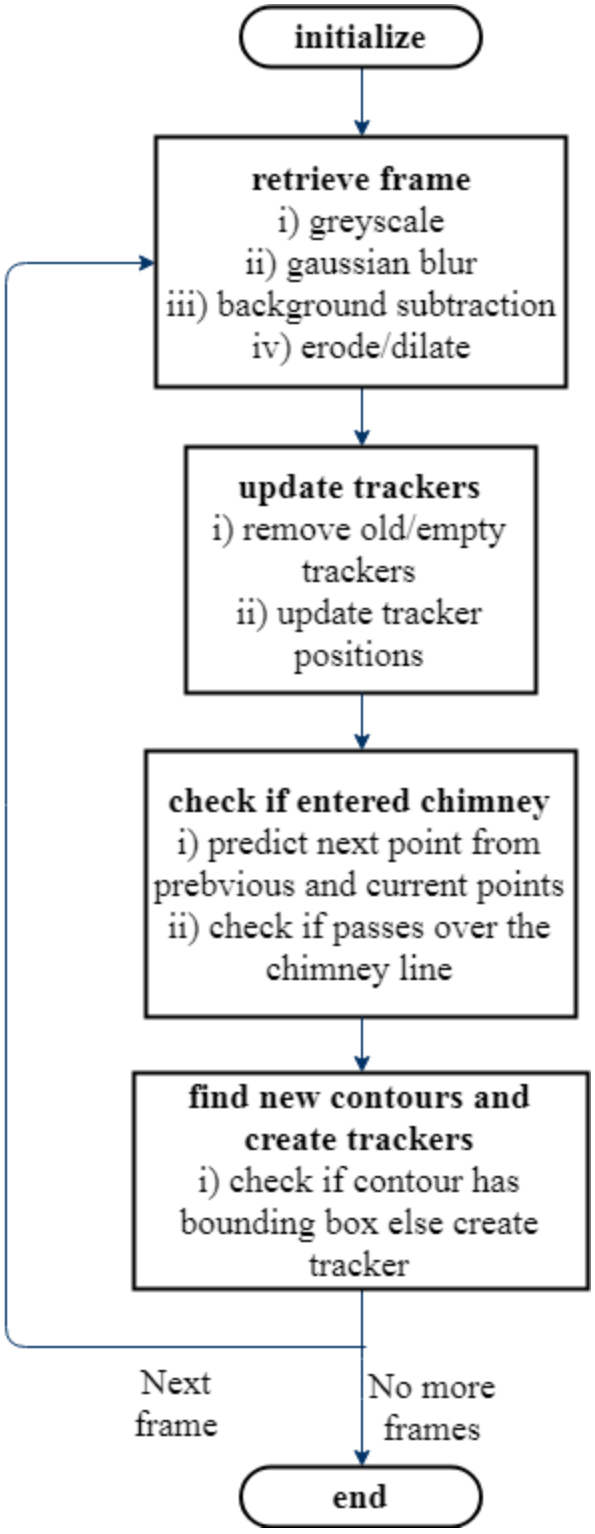

Figure 15: Bird counting process diagram

To summarise, the final design implementation iterates through each frame of the imported video and implements the retrieve process, then updates the trackers, checks if

the object (bird) entered the chimney, and then finds the new contours and creates the trackers. Once all of these processes have run, the program checks for the next frame. If it finds a next frame it returns to the retrieve frame process again and reruns through all the processes that follow it. If no more frames are detected, the program terminates.

The predicted point is calculated by creating a vector from the tracker's previous point to its current point, then multiplying the vector by 2 to extend it in the direction that the bird is currently moving. The predicted point is used to determine if the swift will enter the chimney. The algorithm first checks if the bird is currently within the chimney x-range where it could actually enter the chimney. The current point is then tested to determine if it is above the chimney entrance line, and the predicted point is tested to determine if it is below the chimney point. If all the checks evaluate as true, the bird is said to have entered the chimney.

VIII Testing & Validation

Since occlusion is a major problem for tracking birds, the prototype application will be stress tested when multiple birds are occluded, i.e. when a bird swarm enters the chimney. The test involves measuring the precision and recall of the tracking algorithm.

First, a few representative samples of bird swarms are found. Next the following metrics are manually determined from the samples:

- true positives (bird enters, count increments),
- false positives (no bird enters, count increments)
- false negatives (bird enters, count doesn't increment)

These values are included in the table below:

Sample name	Start Frame	End Frame	Entering Actual	True Positives	False Positives	False Negatives	Precision	Recall
birds_busy1	252	271	17	10	2	7	0.83	0.59
birds_busy1	273	400	80	42	8	38	0.84	0.53
birds_busy2	1088	1285	95	64	7	31	0.90	0.67

Table 3: precision and recall of bird swarm tracking

The precision metric means that on average 85.7% of counted objects are actual birds. The recall metric means that on average 59.7% of birds which enter are counted. These metrics meet expectations. Occasionally moving weather patterns or objects near the chimney will be detected as a bird entering, and reduce the precision. Also, bird occlusion near the chimney reduces recall since multiple bird contours merge into one as they enter the chimney.

The performance objective can be tested by comparison between the processing time of sample videos and the video length. This comparison was done on a system with the following specifications:

- Type: MacBook Pro 13 (2018)
- OS: MacOS 10.13.6
- Processor: 8th gen i7 2.7GHz
- Memory: 16GB 2133MHz LPDDR3
- Graphics: Intel Iris Plus Graphics 655 1536 MB
- Storage: 512 SSD
- Display: 13.3-inch (2560 x 1600)

The results of runtime testing are shown in Table 4. It should be noted that the runtime tests meet the performance objectives over multiple runs. Additional testing was performed on an older MacBook from 2015, wherein the code processing time never surpassed the video runtime.

File name	Realtime Time (mm:ss)	Code Time (mm:ss)	Code Time	Number Birds
nobirds_052117213436.mp4	05:01	01:31	30%	0
birds_052117205059.mp4	05:01	02:08	43%	627
birds_20170523212858.mp4	05:02	01:13	24%	53

Table 4: Performance (time) metrics

The performance results show the videos are processed within 33% of the video length. This is expected since the software can process the videos at a rate greater than the human eye can process a movie. This mean the final process achieves the performance objective.

IX Discussion & Recommendations

The test results detailed in Section VIII meet the objectives. The prototype remains performant during changing lighting and weather conditions and provides a usable GUI.

Although the final implementation reaches the specified objects, there were a number of challenges that were encountered during implementation and testing of the counting algorithm. The challenges that were immediately present were that the birds small and fast moving, which initially makes them more difficult to track. As the project progressed, one main challenge was dealing with the video footage in its given frame rate. The frame rate is sufficiently low and the birds are sufficiently fast that they move a relatively large distance between frames, which provides a major challenge for the tracking algorithms. With a higher frame rate, the trackers would have a much easier time

continuously tracking their target because the birds would move smaller distances between frames. Extending on this point, the birds are able to change directions quite drastically between frames, providing a further challenge for the trackers.

Because the birds can move a relatively large distance between frames, tracker bounding boxes must be sufficiently large to contain the birds after advancing to the next frame. If the bounding box does not contain the bird after the frame advances, the tracker will be removed as it can not longer continue tracking its target. As the tracker box size is increased, the possibility of the bounding box containing more than one bird increases, which is undesirable. As birds approach the chimney entrance, they cluster together and often overlap. The implemented trackers have useful but limited occlusion handling to deal with this issue. All these challenges combined make the tracker less effective.

One limitation of the testing and validation phase of the project is the known bird counts for test cases are obtained with manual counts. This is a difficult task due to the distances birds move each frame.

One recommendation for improvement of the prototype is build an algorithm to identify the relevant chimney edges. This means the user doesn't have to manually select the chimney edge to count the birds. Additionally, more types of trackers could be tested and added.

X Conclusion

In conclusion, all objectives were met. Python was selected as the language of implementation for the project, and OpenCV was chosen as the computer vision software package. The application has a usable GUI to make the application easy to use. The GUI features on-screen instructions, standard media control symbols, and tooltips. The application is performant in the sense that the time required to process input videos was less than the runtime of the videos. Lastly and most importantly, the application exports a CSV file which contains bird counts, fulfilling the main requirement of the project. Counting of birds is accomplished through the use of background subtraction for tracking birds and the MOSSE tracker for tracking the birds.

The major limitation of the prototype is its limited ability to handle occluded birds near the chimney entrance. The minor limitation of the project is that the ground truth values for the bird counts used to measure application accuracy were not exact values.

References

[1] "Chimney Swift", *ontario.ca*, 2018. [Online]. Available: https://www.ontario.ca/page/chimney-swift. [Accessed: 30- Jul- 2018].

[2] "2016 Ontario SwiftWatch Report", *SwiftWatch and Bird Studies Canada*, 2018. Available: https://www.birdscanada.org/volunteer/ai/resources/2016_Ontario_SwiftWatch_Annual_Report_EN.pdf [Accessed: 30- Jul- 2018]

[3] W. Luo, J. Xing and A. Milan, "Multiple Object Tracking: A Literature Review", pp. 1-14, 2018. Available: https://arxiv.org/abs/1409.7618

[4] A. Mordvintsev and A. K., "Background Subraction - OpenCV-Python Tutorials", *Opencv-python-tutroals.readthedocs.io*, 2013. [Online]. Available: http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_video/py_bg_subtraction/py_bg_subtraction.html. [Accessed: 30- Jul- 2018].

[5] A. Mordvintsev and A. K., "Image Thresholding - OpenCV-Python Tutorials", *Opencv-python-tutroals.readthedocs.io*, 2013. [Online]. Available: http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_thresholding/py_thresholding.html. [Accessed: 30- Jul- 2018].

[6] "OpenCV Library", Opencv.org, 2018. [Online]. Available: https://opencv.org/. [Accessed: 30- Jul- 2018].

[7] "SimpleCV - Computer Vision Platform using Python", *Simplecv.org*, 2018. [Online]. Available: http://simplecv.org/. [Accessed: 30- Jul- 2018].

[8] "SimpleCV.Tracking package", *Simplecv.readthedocs.io*, 2011. [Online]. Available: http://simplecv.readthedocs.io/en/latest/SimpleCV.Tracking.html. [Accessed: 30- Jul- 2018].

[9] "OpenCV: Object Tracking", *Docs.opencv.org*, 2017. [Online]. Available: https://docs.opencv.org/3.4.0/dc/d6b/group__video__track.html. [Accessed: 30- Jul- 2018].

[10] "OpenCV: Tracking API", *Docs.opencv.org*, 2017. [Online]. Available: https://docs.opencv.org/3.4.0/d9/df8/group__tracking.html. [Accessed: 30- Jul- 2018].

[11] "Users - OpenCV Q&A Forum", *Answers.opencv.org*, 2018. [Online]. Available: http://answers.opencv.org/users/. [Accessed: 30- Jul- 2018].

[12] "OpenCV: cv::KalmanFilter Class Reference", *Docs.opencv.org*, 2018. [Online]. Available: https://docs.opencv.org/3.4/dd/d6a/classcv_1_1KalmanFilter.html. [Accessed: 30- Jul- 2018].

[13] S. Ananthakrishnan, "Multi Object Tracker Using Kalman Filter & Hungarian Algorithm", *GitHub*, 2017. [Online]. Available: https://github.com/srianant/kalman_filter_multi_object_tracking. [Accessed: 30- Jul- 2018].

[14] D. Bolme, J. Beveridge, B. Draper and Y. Lui, "Visual Object Tracking using Adaptive Correlation Filters", Colorado State University. Available: http://www.cs.colostate.edu/~vision/publications/bolme_cvpr10.pdf

[15] "Python Programming Tutorials", *Python Programming*, 2018. [Online]. Available: https://pythonprogramming.net/loading-images-python-opencv-tutorial/. [Accessed: 31- Jul- 2018].

[16] "What is PyQt?", *Riverbank Computing*, 2018. [Online]. Available: https://riverbankcomputing.com/software/pyqt/intro. [Accessed: 31- Jul- 2018].

[17] "Qt Designer Manual", *Qt Documentation*, 2018. [Online]. Available: http://doc.qt.io/qt-5/qtdesigner-manual.html. [Accessed: 31- Jul- 2018].