Space-Engineering Final Project
Ground Station

Department Of Computer Science

Ariel University

Dor Getter, Oren Lacker and Eldar Takach

13/06/2021

Abstract

In our final project in space engineering course at Ariel university The goal was to build a self oriented optical ground station, which is capable of detecting and tracking moving objects in the sky i.e satellites. To achieve this goal we chose the NEXSTAR 8SE COMPUTERIZED TELESCOPE which is the Celestron's signature orange-tube telescope.
The telescope combines advanced features and excellent optics in one easy-to-use system. One of the advantages of the chosen telescope, is that it's open source code is suitable for controlling the telescope mount, using a python script.
For that matter of building the self-oriented ground station, we divided our mission into two main tasks:

1) Motion detection algorithm - which is based on comparing two frames in order to detect changes in pixels location, which will indicate that there was a movement. This ability will be useful since the objects we want to locate are very far away, yielding that identification using the naked eye is problematic.

2) Object Tracking - Enables to lock on specific targets and use optical flow concepts of estimation, using Lucas-Kanade method to keep on tracking the object we chose.

Figure 1: Nexstar Telescope on the test site.

Contents

Chapter 1

Introduction

We decided to use Python, as this is the easiest and most suitable programming language for this task of image processing and controlling the Nexstar 8SE Telescope[2]. For the Nexstar Telescope, an API has been given to us by Semyon Pikolov [4].
Modifications has been made in order to suit our needs.

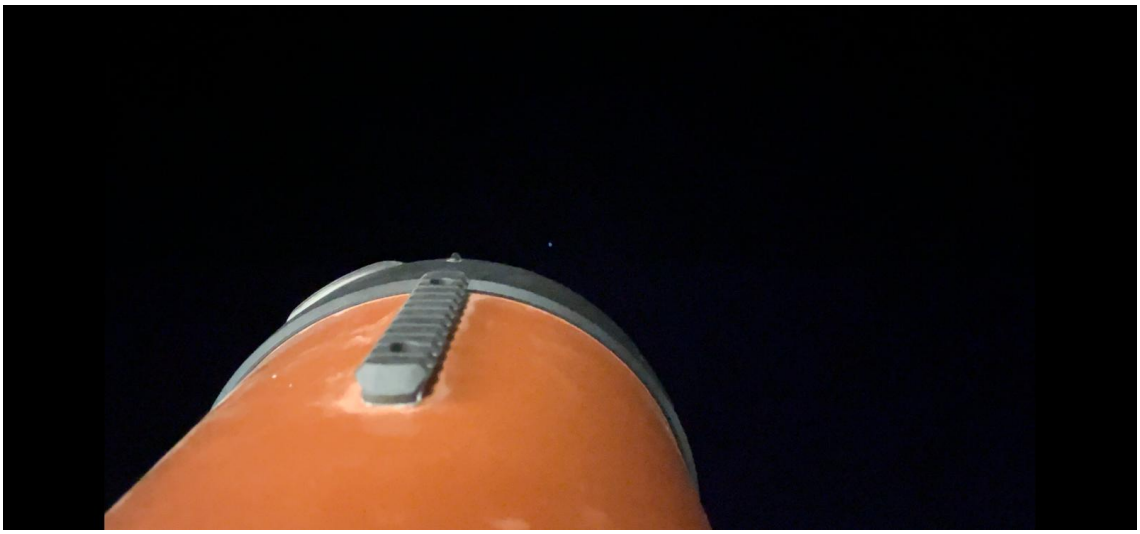

Figure 1.1: Nexstar Telescope.

Chapter 2

Algorithms

§2.1. Motion Detection Algorithm

The goal of motion detection is to recognize motion of objects found in the two given frames. Moreover, finding objects motion can assist in object recognition for tracking abilities. At present methods used in moving object detection are mainly the Frame Subtraction method, the Background Subtraction method and the Optical Flow method. In the Optical flow method we calculate the image optical flow field, and do cluster processing according to the optical flow distribution characteristics of the image. This method can get the complete movement information and separate the moving object from the background better.

2.1.1 *Proposed Method*

Detection of moving objects from a sequence of frames captured by a static camera is widely performed by the frame difference method. The objective of the approach is detecting the moving objects, based on the difference between the existing frame and the reference frame. The frame difference method is the common method of motion detection. This method relays on pixel-based differences to find the moving object[5]. Difference of Two Consecutive Frames:

Definition 2.1 *I_k is supposed to be the value of the K_{th} frame in image sequences. I_{k+1} is the value of the $(K+1)_{th}$ frame in image sequences*

The absolute differential image is defined as follows:

Definition 2.2 $I_{d(K,K+1)} = /I_{K+1} \text{-} I_K/$.

When there is a movement in the frame:
When there is a movement in the scenes, the binary image of the difference informs there has been a movement by white mark. when there is no change,it paints with black color.
Using the OpenCV[1] implementation, returns a set of contours that indicates the moving objects location in the frame.

```
>>> diff = cv2.absdiff(frame1, frame2)
>>> gray = cv2.cvtColor(diff, cv2.COLOR_BGR2GRAY)
>>> blur = cv2.GaussianBlur(gray, (3, 3), 0)
>>> _, thresh = cv2.threshold(blur, 20, 255, cv2.THRESH_BINARY)
>>> dilated = cv2.dilate(thresh, None, iterations=5)
>>> contours, _ = cv2.findContours(dilated, cv2.RETR_TREE,
>>> cv2.CHAIN_APPROX_SIMPLE)
```

§2.2. Object tracking algorithms

2.2.1 *Proposed Method*

For the Object tracking algorithm the Optical Flow Pyramid (LK)[3] was chosen. The Lucas–Kanade method is a widely used differential method for optical flow estimation developed by Bruce D. Lucas and Takeo Kanade. It assumes that the flow is essentially constant in a local neighbourhood of the pixel under consideration, and solves the basic optical flow equations for all the pixels in that neighbourhood, by the least squares criterion. By combining information from several nearby pixels, the Lucas–Kanade method can often resolve the inherent ambiguity of the optical flow equation. It is also less sensitive to image noise than point wise methods. On the other hand, since it is a purely local method, it cannot provide flow information in the interior of uniform regions of the image. We used this technique to track our objective and feature it's x and y coordinates in the frame.

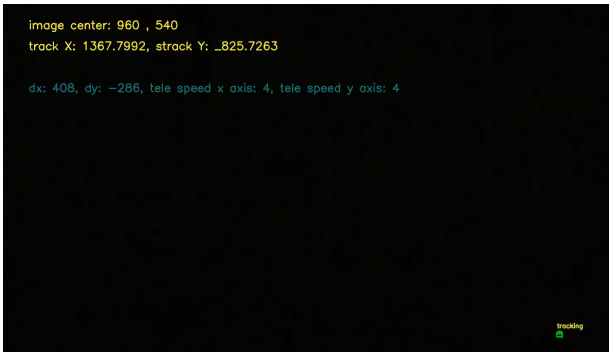

Figure 2.1: Object tracking.

§2.3. Telescope control

To keep on tracking the objective, an algorithm for controlling the telescope mount speed and direction was needed. The problem of moving the mount was divided into two sub-problems:

1) The speed of movement - One problem we encountered, is that fast movements of the mount generated a noisy image, which interferes with the algorithm responsible for tracking the objective. For this manner an adaptive speed algorithm was introduced.
This algorithm sets the mount's speed according to the object speed and distance from the center of the frame, so it will be fast enough to keep the object in the frame, and slow enough to keep the image noise free as possible.

2) The movement direction - As we keep tracking the object, the x and y coordinates of the center of the object is given to an algorithm, which calculates the amount of pixels needed for the camera to move to center the object in the center of the frame. This is done by subtracting the frame center coordinates from the x and y location of the object. In order to pass those instructions into the telescope DC motor, we used an API written in Python which uses a serial port.
An API provided to us by Semyon[4] has been modified to suit our needs.

Chapter 3

Experiments

3.0.1 *Initial experiment: testing the algorithms on video*

In the initial phase, the testing of the algorithms was done using a video of the ISS station moving at an altitude of about 870 miles. It can be seen that the algorithm is able to identify and track the space station despite the great difficulty of the distance.

3.0.1.1 Results

The algorithms were performed and tested in daylight videos and also night-time videos. The ability to identify movement and track small objects seemed to be working as expected.

Figure 3.1: Tracking objects on video.

3.0.2 *Secondary experiment: testing the algorithms on live balloon experiment*

In the second experiment, we took part in an experiment conducted by Professor Ben Moshe, lecturer of the course, in which a balloon flowering was conducted to simulate a satellite. We tested our code and algorithms on videos from the launch. Also in this attempt we were able to present the object's x and y coordinates.

3.0.2.1 Results

Good tracking and recognition were achieved on different quality videos (cell phone recording, and video camera).
We were able to detect the object in high altitude and with gust winds which caused the balloon to change direction rapidly.

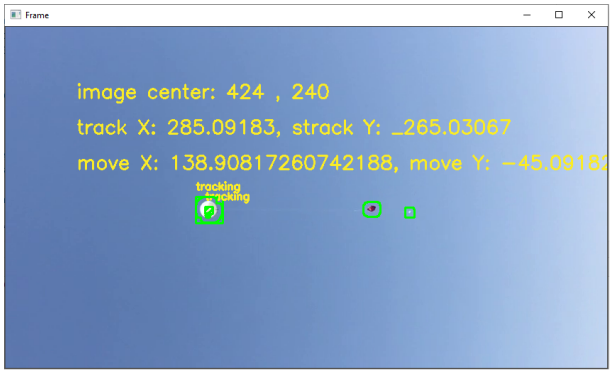

Figure 3.2: Tracking the Balloon.

3.0.3 *FINAL EXPERIMENT: TRACKING SATELLITE NIGHT-TIME*

In the second stage we went from testing the algorithms by video, to testing it in the real world. To do this, we built an interface for working with the tripod motors of a telescope and integrating the algorithms. The algorithm detects an object (probably a satellite or a plane at high altitude) and after a "follow" command the algorithm tracks the object while using the telescopic tripod motors to keep it in the center of the frame.

3.0.3.1 Results

We can see the object data in the frame as well as the motion data for the motors on the X and Y axes as well as the speed of movement in each of the axes.

Figure 3.3:
Detecting movement of satellite

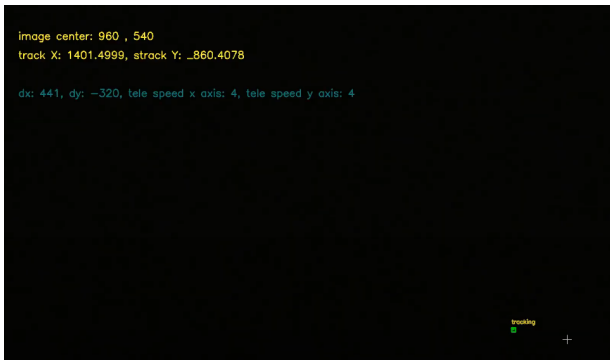

Figure 3.4: Tracking the satellite

Chapter 4

Conclusion

Object Detection and Tracking are one of the most common problems in Computer Science in general and in Computer Vision in particular.
There are many solutions to both of the problems above, in this project we did not try to find new ones, but used the common solutions in specific use-case. Our goal was to detect various objects in the sky, and track a selected object through its movement in the sky. We achieved our goal.

Figure 4.1: Visit our GitHub Repository

Bibliography

[1] Intel. *OpenCV module for Computer Vision*. Vol. 4. Intel, 2021.

[2] Celestron LLC. *Telescope specs*. 2007. URL: https://www.celestron.com/products/nexstar-8se-computerized-telescope#specifications (visited on 09/30/2010).

[3] OpenVX. *Optical Flow Pyramid (LK)*. 2007. URL: https://www.khronos.org/registry/OpenVX/specs/1.1_SC/html/d0/d0c/group__group__vision__function__opticalflowpyrlk.html (visited on 09/30/2010).

[4] Semyon Pikolov. *Telescope control*. Vol. 1. Semyon Pikolov, 2020.

[5] Nishu Singla. *Motion Detection Based on Frame Difference Method*. 2007. URL: https://www.ripublication.com/irph/ijict_spl/ijictv4n15spl_10.pdf (visited on 09/30/2010).