Independently-Moving Object Detection

Andrew Morato

Abstract

In any given video sequence, there may be a moving object of interest. It could certainly be advantageous to detect the presence of a specified object and track it throughout a video. This paper presents a lightweight approach to this problem and the results of its application on a set of simple testing data.

1. Introduction

Due to the prevalence and continued increase of technology and robotics in everyday life, there is a sizeable demand for autonomous and self-correcting machines. Clearly, any kind of automaton follows the process of perceiving its environment, understanding it, then acting based on what it has seen. Central to perception is making sense of the data presented by the robot's sensors. Detecting moving objects in a video sequence and keeping track of their location through all the frames of the video is a clear example of understanding the information presented by sensors, in this case a camera. How can this be achieved?

1.1. Motivation

There are several solutions to this problem already in circulation, the most common ones involve training a system to recognize objects in an image by scanning a very large dataset of pre-existing samples of the desired object and then applying that knowledge to detect the same object in a new image. Modern implementations are successful with a fairly high level of confidence. Once sufficiently trained, a system can be instructed to detect a particular object in every frame of a video sequence, thus "tracking" the object in the video.

I was motivated to approach this problem without going the route of heavy training, but rather detecting and tracking the object based on several patterns found by extracted key features. In other words, the idea behind this approach is to reduce runtime while sacrificing as little accuracy as possible.

1.2. Applications

A good solution to this problem is essential to any kind of moving robot. Keeping track of other objects surrounding the robot is key to avoiding collisions or amending its behavior due to the changing external factors. Perhaps the clearest example of this is in self-driving cars. Knowing the location of moving objects in close proximity to the car is paramount as some adjustments may be necessary to ensure passenger safety. In another example, to avoid manually parsing a very large dataset, say security footage, a robot could be trained to detect objects of interest and track their movements/change of position to see if anything interesting has been done. Another application could be in an automaton's pursuit of a moving target. Computer vision is essential to autonomous robotics and can assist in almost any task assigned to an intelligent agent.

1.3. Dataset

The data used to test the implementation of this method, both during incremental testing and after implementing the entire system, is a collection of simple image sequences. Each image sequence is essentially a dissected video, consisting of 600 frames. The videos are relatively simple, there is not much obstruction and the individual frames are not crowded with objects. Samples include a boat moving in the water, a child on a bike, a jet flying through the sky, and an ice skater dancing.

2. Approach

The general algorithm can be divided into three sub problems. To detect an object in an image, the robot has to have an idea of which object to "look for". The robot could be trained with a huge training set so that it can find a similar object with great accuracy, but this approach relies on feature extraction of a small number of very similar images (i.e. attempting to isolate a small pattern in a view of the object). Once there is a pattern to identify in memory, the next step consists of scanning the first frames of the video sequence for the same pattern. If this was found, then the robot detected the object and now knows its initial location in the video. The final step is a repetitive process where the location of the object is tracked frame by frame by essentially checking the next frame for the object in a similar location to the previous frame and taking note of its slightly changing locations. This process is explained in further detail below. A diagram of this process is depicted below in Figure 2.1.

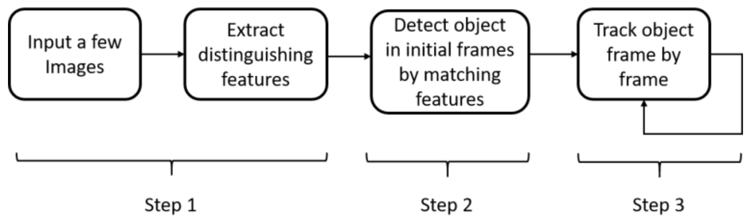
Input a few Images
Extract distinguishing features
Detect object in initial frames by matching features
Track object frame by frame
Step 1
Step 2
Step 3

Figure 2.1

2.1. Feature extraction

Initially, this solution worked with a bounding box drawn around the object to track. This eliminated the need for feature extraction and object detection in the first few frames and only the problem of tracking the object remained. After implementing a solution for the frame-by-frame tracking problem, I attempted to develop a solution to replace drawing a bounding box. As seen in Figure 2.1 above, this solution required a few input images that contained a pattern that could be matched with the object in the video. If there was no such pattern, then this implementation failed at detecting the object in the first frames in the video sequence.

The successful detections worked first by extracting the distinguishable features of the input images using the Harris corner detector or SIFT feature detection. With these features extracted (an array of x-y points of the features' location), we are able to move to the next stage in this algorithm.

2.2. Initial object detection

After extracting the features, this solution turns to applying the same feature extraction techniques to the first frame in the video sequence where the object is to be tracked. This yields a list of key features that can include some distinguishable points belonging to the object and some belonging to other objects in the frame or the environment in the image. In some instances, where there are too many extracted features from the first frame, I applied background subtraction in an attempt to simplify the problem. In other words, I extracted the key features of several frames and removed the features that appeared too many times, i.e. removing points belonging to objects that did not move throughout the video (likely the lifeless environment).

Now that there is a list of key features found from an input image and a list of key features extracted from the first frame, both lists are searched for a matching set of key features. If there is a convincing match, then there must exist a correspondence between the input images and the first frame, which indicates that the object has been detected. This can certainly lead to false positives if a pattern from the input images just so happens to appear elsewhere or false negatives if the pattern is present but not accurately matched. Using this system, however, there was a reasonable margin of success on the simple testing set used. An example of this is shown in Figure 2.3. A very similar pattern is shown in both bikes, which led to a successful match. Note: this process is similar to the homework assignment from class about sift point correspondences.

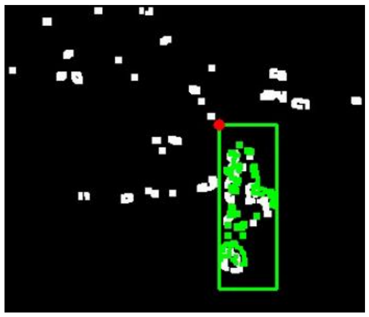

Figure 2.2

2.3. Frame by frame tracking

Once the object's location is found in the first frame of the video, finding the object in the second frame becomes an easier task. Presumably, the object in the second frame will only change a little from the object just encountered in the first frame. Assuming a rigid body, the object can move along the x or y axes, scale itself by moving closer to the camera or further away, and rotate along any axis. If the object is not a rigid body, it can change itself in even more ways. Regardless of the change, it will be miniscule from any particular frame to the subsequent one. This observation is key to this approach to track the object frame by frame.

The feature detectors are once again used but only on the area where the object was detected. This produces a set of distinguishable features belonging to the object as shown in the first frame, making a kind of mask. This mask is then overlaid over the second frame in the location that the object was found in the first frame and the

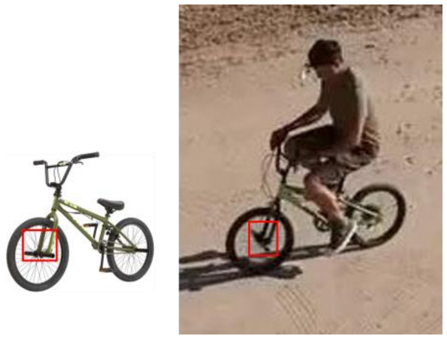

Figure 2.3 - An example of a matched set of features (a pattern) found from the left input image and the right frame

surrounding areas. There will not be a perfect match since the object has changed, but the location of the best match is recorded as the location of the object in the second frame. The mask is updated to represent the key features found by running the detectors on the new location in the second frame. The same process is applied on the third frame with the second frame's mask and location, and so forth until the last frame. An example of the mask being superimposed over a frame is shown in Figure 2.2.

The advantage to this approach is that the mask continuously evolves to match the shape of the latest version of the object. If the object were to slowly rotate so that a completely new face is presented to the camera, this algorithm could still track the object even though the initial key features (of the input image/first frame) and the key features of the current pose of the object are effectively a disjoint set. Unfortunately, while this approach can track an object even if it significantly changes throughout the course of a video sequence, it is vulnerable to producing a faulty mask if the object is obstructed for a time.

3. Results

After implementation, several results were produced by applying the program to some of the video sequences from the dataset. This solution worked very well when the changes of the object in the video were translations or slow rotations. The program struggled when the objects underwent rapid rotation or scaling in the form of quickly zooming in and out.

3.1. Confidence levels

Every instance of superimposing the mask and matching its pattern with the pattern found in the subsequent frame produced a confidence level, an indication of how much of the pattern was matched. If the confidence levels were too low in all cases, the program concluded that the object disappeared from view entirely. In the average case, overlaying the mask resulted in confidence levels of 70 to 85 percent. The minimum threshold to consider the object present in tracking its location was a confidence level of at least 50 percent. Anything above that was accepted and the mask was amended as a result.

3.2. False positives

In testing with the dataset, there were some edge cases where the mask kept adapting to what it thought was the object but was actually some other passing object that just barely met the match percentage requirement. The mask then adapted to this false object and reported that the object was successfully tracked when in fact it was not the case. This issue may be reduced by increasing the minimum percentage requirement, however the mask's flexibility to track a rapidly changing object would decrease as well.

3.3. Failure to track

In addition to the false positives, this solution failed to track some objects in videos that rotated too rapidly or moved extremely quickly. For example, in one of the video sequences in the dataset, an ice skater rotated very quickly and changed position from facing the camera to looking away from it in the space of about 3 frames. This proved to be too quick of change for the adaptive mask to adjust to, and as a result, it declared that the object disappeared from view. Fortunately, the ice skater moved back into the mask's frame and tracking was able to resume.

Additionally, if there is too much noise in the frames or there is significant obstruction, it becomes difficult to isolate and track the desired object. With too many moving objects, it becomes difficult to detect and track an object, and background subtraction offers no help since it only eliminates objects that are not in motion.

4. Discussion

Although this implementation works in the majority of test cases, it is not quite robust enough to rely upon. Too many cases exist where this solution yields false positives or fails to track the object altogether. The flexible mask that allows for change in tracking an object has no way of measuring its own reliability.

4.1. Improvements

This solution could have been improved in several ways. In the current implementation, there are several instances when the mask departs from the object and begins tracking some other object. This happened because the passing object was barely similar enough to current object (enough to meet the confidence interval). Due to the adaptive mask, the confidence level does not stay low, so it becomes very difficult to detect this error. A possible solution is to keep a frame of reference of the key features found from the input images and several masks from random frames in the past so that the program can occasionally verify, with those references, that the object it is currently tracking is the desired one. If there is no correspondence (i.e. not the original object), the program can “backtrack” to a point where the confidence level was low (where it might have gone wrong) and try a different path or conclude that the object was lost, which is preferable to a false positive.

Of course, this could be solved much easier with a large training set so there never need be much doubt. Once the object has been lost in any given frame, the algorithm could be restarted so that the current frame is completely rescanned to detect the original object. This approach may improve the robustness of the solution presented in this paper but would sacrifice runtime, as this procedure can be very expensive, which is precisely what this solution seeks to avoid. Based on tests with the current dataset, my algorithm, although promising in some direction, lacks accuracy and reliability to be considered over current solutions to this problem.

4.2. Implementation

This solution was implemented with use of OpenCV 3.4 in Python 3.7.0. The code can be found in github.com/andy9kv/Projects/