Rahil Dedhia, Ankeet Mutha, Erika Weiler

Computer Vision Project Write Up - Object Tracking

What was the goal of your project?

The goal of our project was initially track cars from a video from a street camera. We planned to do this by implementing either mean shift or cam shift object tracking algorithms from scratch. We also hoped to investigate a few different ways of finding similarity in order to do the object tracking. Our goals shifted slightly when Ankeet's SCOPE project shifted.

We tracked objects in different videos using an approach similar to mean shift, which relied on the comparison of color histograms. We wanted to make an effecient and reliable tracking algorithm for stationary video cameras. We also compared our own algorithm against opencv's mean shift and cam shift algorithms.

Describe how your system works. Make sure to include the basic components and algorithms that comprise your project.

To object track we chose to use a self implemented algorithm that is based on mean shift. We choose an object within the first frame of the video and created a box just large enough to fit it. We then made a color histogram of that space. For successive frames, we partitioned the frame into boxes of that object's size, and calculated a color histogram in each of those boxes. Then we figured out which histogram most closely matched the original box. In order to more heavily weight closer boxes, we also subtracted the normalized (from 0 to 1) distance that the box was away from the original frame from the weight of the match. From the weights, we were able to find the location of the object in the new frame.

In order to improve our efficiency, we reduced the size of the red, green, and blue bins of our color histogram as much as possible. When we repeated the algorithm using hsv values instead of rgb, we did the same thing for hue, saturation, and value. In terms of efficiency, our algorithm was also heavily affected by the number of boxes we had to iterate through for each frame. For tracking small objects, like a zoomed out bottle cap, we found that a small box was necessary for accuracy, but significantly slowed down the tracking. To address this, we looked into skipping frames in order to speed up our algorithm. On the specific videos we implemented them on, our algorithm turned out to be more accurate than openCV's mean shift and cam shift algorithms.

Describe a design decision you had to make when working on your project and what you ultimately did (and why)? These design decisions could be particular choices for how you implemented some part of an algorithm or perhaps a decision regarding which of two external packages to use in your project.
When we initially implemented our mean shift algorithm, it was very slow at tracking a bottlecap, because it was so small, and we had to adjust the box size we were iterating over accordingly. The image was not playing in real time. We set up timers to figure out what was taking longer, creating histograms all across the image, or comparing histograms. Comparing histograms took a much longer time. We realized this made a lot of sense because histograms are dictionaries and creating dictionaries is a constant time operation. In order to fix this problem, we found two solutions. Solution one was to only look at every fourth frame. This still remained remarkably accurate and also was much faster.

Solution two was to increase the size of the “boxes” we had split the image into. This caused the histogram to be much larger, but as we had just rediscovered, histogram creation is constant time. The issue with this solution was that increasing the box size made the algorithm unreliable for small objects. The solution worked great when we tried it on a video where we were tracking a larger object, and could track a large object. Following a larger object enabled us to compare fewer histograms making the tracking also run much faster.

How did you structure your code?
We have four scripts, two of which are mostly from openCV’s documentation, and two of which are written entirely by us. The functionality of each of the scripts is described in more detail in the readme in our github repository. The most important one is `mean_shift_video.py`, which we developed based on `mean_shift.py`, but has more functionality to actually track an object through a video instead of just a few “frames”.

In this script, we defined an ObjectFrame() class, for which we created instances for each frame of a video. The class had several methods for storing the parameters of each frame and creating and comparing histograms. We tried to separate functionality into many methods to make them more self explanatory and modular.

There was then a main loop outside the class where we defined global variables, initialized each frame, and displayed each frame to the screen. There are only two global variables that need to be changed to switch between videos and between rgb and hsv. Changing these variables changes the hardcoded initial position of the object, and the bins we use for rgb or hsv values, which are changed by passing the `rgb` flag.

What if any challenges did you face along the way?
One challenge we faced when implementing our computer vision was balancing the speed of the video and the accuracy of the object tracking. To solve this we attempted to change the bin numbers, the number of frames looked at, and the size of the histogram block. We also faced some challenges figuring out exactly how mean shift worked and how to best implement it. We wanted to figure out how exactly mean shift worked without looking at code that implemented mean shift. We were able to eventually figure this out by looking at papers and talking to Paul. We ended up with a tracking algorithm that works, but is not quite identical to mean shift.

We also faced issues trying to make our code perform optimally, and that is discussed in our design decisions section. We also had to deal with some strange math, because pixels were referred to in a numpy array. In order to iterate over pixels, we occasionally had to mod by one coordinate and divide by another coordinate. We initially had bugs from this, but were able to resolve them by being more explicit with our variables.

What would you do to improve your project if you had more time?
We would have implemented code on a neato. We were thinking about doing this, but we realized that using a mean shift algorithm on a neato would be less useful, because mean shift doesn't deal well with objects having depth. Mean shift assumes the size of a histogram remains constant, which causes problems when objects appear to grow or shrink based on distance. If we had more time, we could have implemented a cam shift algorithm to put on the neato, so that we could track and follow an object. Cam shift has a changing histogram size, which is much nicer for following objects that move closer and farther away, thereby changing size.

Did you learn any interesting lessons for future robotic programming projects? These could relate to working on robotics projects in teams, working on more open-ended (and longer term) problems, or any other relevant topic.
This project ***was*** directly related to Ankeet's SCOPE project, so the knowledge gained from this object tracking ***would have been*** useful when designing his new SCOPE project. (#*notbitter*). It still might be useful for the SCOPE project. We can possibly use this on our future compRobo project. In addition, Rahil is also working on a robotics SCOPE team, and knowing about object tracking will likely be useful to him.

Additionally, this was a good project to get experience dealing with computational efficiency. We had to weigh the speed of our video against a number of other factors, and make informed design decisions about how to improve the computational time of our algorithm without making in unreliable. In this vein, we had to do a lot of work iterating over large data sets in interesting ways, which is always a useful skill.