Nick Lawrence

Yashad Samant

Brad Wirtz

PA2 report

Base System:

When we were first started implementing our base system, we looked into OpenCV libraries that could detect keypoints using an on center off surround methodology. We ended up coming up against the decision between SIFT and SURF. Rather than commit to one, we decided to implement our system first on the simpler to implement SIFT, then rewrite the system using the more robust SURF and compare the results.

With our base model, we wanted to find a good baseline for what SIFT could accomplish when used in its default form. In order to achieve this we used the basic SIFT constructor with no additional parameters to find our keypoints. When the keypoints were identified, the program picks a random integer to select the index of the keypoint it will highlight in that frame. After this point is selected, the program checks the keypoint against the selections from the last 30 frames to ensure no repeats are selected

Our process for avoiding repeat key frames is as follows. We first select the keypoint with the highest response value for the current frame, we then calculate the (x,y) position for the upper left and lower right corners of the bounding box. We then iterate through an array holding the corner coordinates for the selected keypoint for the previous 30 frames. We then take the maximum left and minimum right x values of the two boxes, as well as the max top and min bottom y values and construct a new bounding box which represents the overlapping area between the two boxes. If this box’s area is any more than half the size of the keypoint area, we label the selection as a repeat and discard it. We then select the point with the next highest response value and begin this cycle again.

We tested the system predominantly on 2 video clips. The first contained a mechanism constructed with Legos, which has a rotating propeller on top, this clip is 6 seconds long. The second scene begins with an empty parking lot, after a few seconds 2 trucks drive into the scene from either side of the frame and cross paths. This clip is 7 seconds long. In the first clip, our base program completes the task in just over 20 seconds, and comes up with around 430 attention windows per frame. According to documentation on the OpenCV site SIFT automatically selects the size of attention windows based on video resolution, however, our program seems fixated on tiny attention windows, producing only a few pixels which are incredibly hard to match to the original frame if viewing only the highlighted selection. This problem does not occur when testing the second clip however, which is a significantly higher resolution than the first. Although the program does prefer extremely small attention windows, there are much larger ones scattered about. However, SIFT finds around 6,500 attention windows per frame which is an enormous number, and significantly slows down the run time compared to the first clip (just under 3 minutes).

Upgraded System using SURF:

For our upgraded model, we implemented SURF as our method for identifying keypoints. In addition to using SURF, we also experimented with parameters such as hessian limits, and octave structures in order to speed up runtime. We also experimented with selecting the keypoint with the maximum hessian response in each frame, rather than selecting them randomly. According to the OpenCV documentation on the libraries, SURF implements a number of methods to increase the speed of SIFT by a factor of around 3. However, the documentation also claims that SURF handles poorly under instances where there is a changing viewport or sudden changes in illumination. For our purposes of grabbing single keypoints without concern for their contents, the downfalls in accuracy of SURF are not an issue. When it came to configuring our SURF implementation, the OpenCV documentation recommends a hessian threshold of 300-500 in order to optimize keypoint selection. However, in our implementation, when selecting the keypoint for each frame, we take the keypoint with the highest response in each frame. Because of this, keypoints generally have very high hessian response values, therefore any threshold below approximately 2000 produces the same list of selected keypoints. For that reason we use a high hessian threshold in our implementation to improve performance. Another choice we made to improve the efficiency was to lock the orientation of the keypoints, due to the keypoint orientation not being relevant to the goal of the base model. When choosing octave and octave layer values, we found it helpful to use an octave count higher than 5 to get a good coverage of important details in the video. On the other hand, we found that increasing octave layer values created more keypoints, but didn’t necessarily lead to an increase in important details detected. Rather, the program would select the same feature for multiple similar scales and lead to a much lower diversity in features selected by the base program. For this reason we chose to stick with only one octave layer to keep performance high and redundancy low.

When running on the test videos, SURF completed the Lego clip in around 8 seconds (under half the time) and produced around 100 attention windows per frame. On the second video the program completed in around 35 seconds (almost 6 times as fast as SIFT) and produced around 450 attention windows on the empty scene, and increased to around 600 when the trucks entered the frame. In both tests the keypoints produced were much larger, and therefore easier to recognize the element of the frame being used.

This implementation provided a high level of performance while processing videos while not sacrificing features detected. However, the decision to select the keypoint with the highest hessian response was not as effective as we had initially hoped. The result of this decision was the effect that our base program would frequently cycle through keypoints in regions of the image with the highest contrast, completely ignoring features that had less intense contrast, but were nevertheless distinguished.

Overall SURF with configured parameters provided a much faster and more focused implementation of the project. Although our process of highlighting keypoints in our upgraded program was not a success, the upgraded program proves that proper configuration of hessian limits and octave structures can significantly increase performance of the task.