# Things that worked

There are several highlights that describe our project’s progress. The best one is that it produces usable results! Installation instructions can be found in README.md. For STI method *Copying Pixels*, run the script, select an avi file, and type *A*. This processes the clip (in an efficient single pass) simultaneously writing the two STI files to the root directory and displaying them in new windows. Using the first video referenced Wildlife.avi, the following is produced:

STI\_column\_CP.png



STI\_row\_CP.png



Similarly, by typing *B* for *Histogram Differences* a similar process happens with the new algorithm. After applying a threshold mask (80% was used in the figures) to the output, the transitions are even more evident.

STI\_column\_HD\_80.png



STI\_row\_HD\_80.png



While the row method for *Copying Pixels* seems to be more smooth, scene five introduces an apparent (vertical) wipe in *Histogram Differences*. The cuts are characterized by vertical bars of low temporal measure, less than 3 frames.

However, some border differences are not greatly distinct causing discontinuities in the bar. Both row and column *HD* versions suffer from this. They also have different noises exemplified by scenes one and two. A proposed way to reduce this noise even further for cuts would be to multiply these two STIs together. That should result in the intersection of them removing the extra random noise like that found in scenes one and three. Unfortunately, that would also reduce the definition of some other cuts like the transition between scenes six and seven. It would still be distinct though, as there is very little otherwise.

Our project is written in Python, using OpenCV, numpy, and matplotlib. It is a familiar language, being used in some of our other classes. Unlike Java it has great built-in UI functionality.

# Not as successful

No project is without its shortcomings. Development was done on Windows, so that was the main target operating system. While Linux and OSX are supported by both OpenCV and Python, this joint program is untested on those systems and thought unlikely to work. Additionally, the CSIL machines have few permissions making installing the necessary libraries more difficult. A different problem was in the pre-development phase, evaluating the requirements posed by the assignment. Specifically, *HD* was hard to grasp exactly how it was supposed to work. Handling the bins and differences needed more information, so emails were sent to the TA for clarification helping work past this. Similarly, the array dimensions and relative directions are easy to mix up, especially when doing both versions. They were not distinct enough to follow easily. As for the structure of the project, it is not conductive to an equitable workload between a pair. The complexities between parts 1.1 and 1.2 are fairly different. Group dynamics lead to one member doing more as a result. Environmental issues were a small problem too. Trying to compile the libraries and their dependencies was a hassle. Their varying version requirements were not fun to mitigate. Uninstalling them was even worse when one realizes they have the wrong version number

# Going forwards

Here are some things we would like to do, given more resources.

1. Checking and correcting the cross-platform compatibility of our project would be nice to have, since that makes the solution much more usable.
2. Have a fuller GUI for selecting files and transforms. Maybe add the ability to change the STI output name/directory.
3. Importing a batch of videos. Why go through manually selecting every option each time for a whole folder? Going along with this automation, detecting timecodes of transitions and writing it in an adjoining csv file. Something like (transition start, transition end).
4. Adding the IBM algorithm mentioned in the extras. It would be interesting to see what the colour similarity does to the first row cut, going from blue to blue.
5. More robust error handling, so that it does not fatally crash on any unhandled execution path. Working off the new GUI, a repeatable *Open File* window would do for invalid files. Clicking on the STI version instead of parsing text removes those problems.

# Suggested assignment improvements

Three key features could be revised to aid in clarity. The *HD* paragraphs are extremely dense. Going through the process slower and with a more substantive analogy to follow should help here. While the figures help a lot, making them more tangible would help show both what is expected of the output and how the STI is created. Specifically, a gif or video of the STI being procedurally generated over time side-by-side with the source video. That would get the dimensions in check and handle some rotational issues. It also helps explain Figure 3, of which (b) which was extremely confusing, like how it progresses from green to basketball back to green.