The purpose of this research is to determine the most efficient and effective way to configure a raspberry pi and a camera to be able to detect motion and record video.

There are two main methods that one could use when undertaking this project. Method one would be to configure the camera to constantly be recording at all times. While this would allow for the most accurate footage and make sure everything was always captured this poses a series of problems.

The first issue is that this method will require a large amount of storage in order to be able to record hours and hours of footage from each day. This problem is confounded when one takes into account the nature of our cloud security system where each pi must upload its footage to a central webapp. Uploading large amounts of footage from constant recording will be both a strain on storage resources and network bandwidth.

The second issue is cost. When the footage is uploaded to the webapp, it will again need to be stored for a period of time. With all cloud providers there is a cost that will quickly amas when storing large amounts of data, the kind that result from recording hours and hours of footage. Combine this with multiple feeds from multiple systems and you quickly exhaust any budget that would be reasonable for such a project.

To help overcome these challenges we have selected method two, which is to only record video when something is actually happening. This means that recording hours of static footage will not be a problem. Our system will be able to detect when its environment is changing and only then begin to record footage. This allows us to use a much smaller amount of storage, both on the pi and the webapp backend, as well as much less bandwidth as there will be less video to record. Method two is not without its disadvantages however. While this method saves space, there is also the risk of not recording important footage because of poor software that fails to detect when the environment is changing(i.e. Something is taking place). This make the choice of software that we use a very important one.

After extensive research there are three main libraries/projects that would allow us to more easily achieve our goal of being able to sense a changing environment in our pi's camera stream:

- OpenCV (open source computer vision)
    - http://opencv.org/
- Motioneyeos
    - https://github.com/ccrisan/motioneyeos
- Motion
    - https://github.com/Motion-Project/motion

Each option offers its own advantages and disadvantages over the others. While there are still many other options to choose from these three were deemed to be the only serious contenders. The following table compares each of the options to the others.

Computer vision library comparison

	OpenCV	**Motioneyeos**	**Motion**
Amount and quality of documentation	Excellent - comprehensive documentation for each version	Poor - little documentation from project maintainers	Medium - Large amount but outdated
Quality of code	Excellent - maintained by industry experts	Good - code is well maintained and up to date with patches	Poor - unsupported branches and no central direction
Performance	Excellent - written in C/C++, multicore processor support. Supports hardware acceleration	Medium - no multicore support or hardware acceleration seems to be available	Medium - large amounts of dependencies make codebase large and difficult to manage
User friendliness	Poor - code is optimized for performance and features, not user friendliness	Excellent - all in one solution that comes with entire operating system	Excellent - simple installation and configuration
Fit to project	Good - no extra cruft attached to code, does what is needed and does it well.	Poor - all in one solution means little flexibility	Medium - a good compromise between ease of use and features

After weighing the options is was decided that OpenCV would be the best choice for the project. OpenCV is a very advanced and capable library that has interfaces for C, C++, Python, and Java and can run on our selected platform of choice. Furthermore it will be able to leverage the multicore processor in the pi as well as possible hardware acceleration.

Aside from the performance advantages of OpenCV, another strong point that lead to it being chosen over the rest was the quality of the documentation. While OpenCV may be more complex than the other options the documentation is very adept at detailing each of the components. There are also many active project members willing to help if need be if one were to get stuck on a particular issue.

By using OpenCV and our own custom software, our cloud security system will be able to accurately sense changes in its environment in real time allowing for an accurate and efficient security system.