Suspect Report Generator

William Christie

SID: 810915676

University of Colorado Boulder

Computer Vision Final Project Report

**Suspect Report Generator**

**Introduction**

Consider a system by which a snap shot image or short video file taken during a criminal encounter could lead to a synthesized suspect report providing invaluable synthesized information regarding the suspect and the crime committed. The initial idea for this final project would be to apply facial detection and image processing to the field of criminal justice and create an application that would help law enforcement synthesize information collected from images and videos into a standardized “Suspect Report”. My main goal in creating this system is to help law enforcement manage the wealth of digital information that is collected through the course of an investigation and automate some of the report generation process to save investigator time and resources. I believe that the features described in this application could be combined with other computer vision and machine learning techniques to generate documents that could be acceptable for submission as evidence in court cases. This application was written in MATLAB version 2016a and utilizes several functions described in the Computer Vision Toolbox as well the document model objects and document part templates from the MATLAB Report Generator package.

**Algorithm**

The Suspect Report Generator application’s algorithm takes as inputs either an image or video file and returns the generated “Suspect Report” PDF report detailing the department information, suspect’s name, given suspect identification number, investigation number, investigation status, and observations regarding the video or image. The algorithm prompts the user to upload a video or image file that henceforth will be referred to as the *frame*. After creating the frame, the algorithm creates a *vision.CascadeObjectDetector*  object via inbuilt MATLAB functions. The algorithm then performs facial detection using the Viola Jones Algorithm (Viola and Jones, 2001) by calling the MATLAB *step* function on the *frame* and detector object to locate all image features representing a frontal view face in the frame and returns a matrix containing the set of bounding boxes around every detected face in the *frame*. The algorithm must then iterate over the set of bounding box coordinates and plots an indexed box onto the frame which is displayed to the user as a figure for “suspect selection.” After displaying the annotated *frame*, the user is prompted to select the indexed bounding box surrounding the face of interest. The selected indexed face is then extracted from the *frame* and filtered to sharpen all edges prior to resizing to fit within the report as the *reportPicture*. In the event that facial detection fails via incorrect feature identification or a failure to detect any facial features, the user is allowed to either continue without a referenced suspect or quit the application entirely. During the report generation stage, the algorithm collects user input regarding the suspect name, suspect identification number, investigation status and observations which is stored in a cell array named suspectInfo. The PDF report is created using a series of document objects as described in the MATLAB report generator documentation and a predefined HTML document part template stored within the */myPDFtemplate/stylesheets* directory. Once the collected *suspectInfo* and *reportPicture* are placed into the report the PDF document is saved into the same directory as the application. Once report generation is complete the user is returned to the opening menu prompt to either finish and quit or run the application on another image or video file.

I originally hypothesized that I would be able to create a system using functions from the Computer Vision Toolbox that would able to perform facial recognition with a high success rate regardless of sex provided that the face in question was fully in the image and not disguised by glasses, facial hair or clothing. I additionally suspected that strange illumination, shadowing, or facial orientation might also inhibit the success of the facial recognition portion of the algorithm. Finally, I hypothesized that because the application was using the Viola Jones Algorithm that the facial recognition portion of the algorithm could be expedited fast enough that no undue wait time overhead would be expected regardless of the number of features in an image or video frame.

**Data and Results**

During the development of Suspect Report Generator algorithm it was necessary to test the facial recognition portion of the algorithm. The data set chosen for testing was Collection of Facial Images (Spacek, 2016) consisting of seven thousand nine hundred 24-bit color JPG images of both male and female subjects of various racial origins and various ages. Facial hair and glasses were present and lighting was an artificial tungsten and fluorescent overhead. The images were organized into four directories based on the lighting, facial expression changes, scale, and background type. The figures below are representative examples of the dataset (from running testFacialRecog.m) and also show the results of running Suspect Report Generator. Statistical results of running testFacialRecog.m can be found in Appendix A.



**Figure 1: Faces94 “Anpage”:** Considered to be a simple case given the one tone background and sufficient illumination with few illumination changes. Subjects exhibited few facial expression changes. System is easily able to perform facial recognition under theses conditions.

**

**Figure 2: Faces94 “doraj”:** Facial recognition can be difficult when glasses or facial hair are present in an image. The system is able to perform facial recognition under optimal environmental conditions when facial hair and glasses are present.

**

**Figure 3: Faces95 “Jserai”:** Considered to be at a medium level of difficulty given multiple illumination changes due to shadowing and facial translation (even to the point of leaving the frame). As can be seen above, cutting out part of the face can result in failed facial recognition with high levels of shadowing.

****

**Figure 4: Faces96 “mystery”:** Considered to be at a high level of difficulty given the confusing glossy poster background. Features obscured by glasses, illumination changes at high level and facial translation evident. The second image in the sequence likely was not recognized due to the increased glare from the glasses and the large background disturbances.



**Figure 5: Grimace “Will”:** Considered to be a high level of difficulty given particularly poor illumination, obscure facial features, illumination changes and facial translation. As can be seen, the system is generally resilient to low illumination and obscure facial expressions including closing of the eyes.



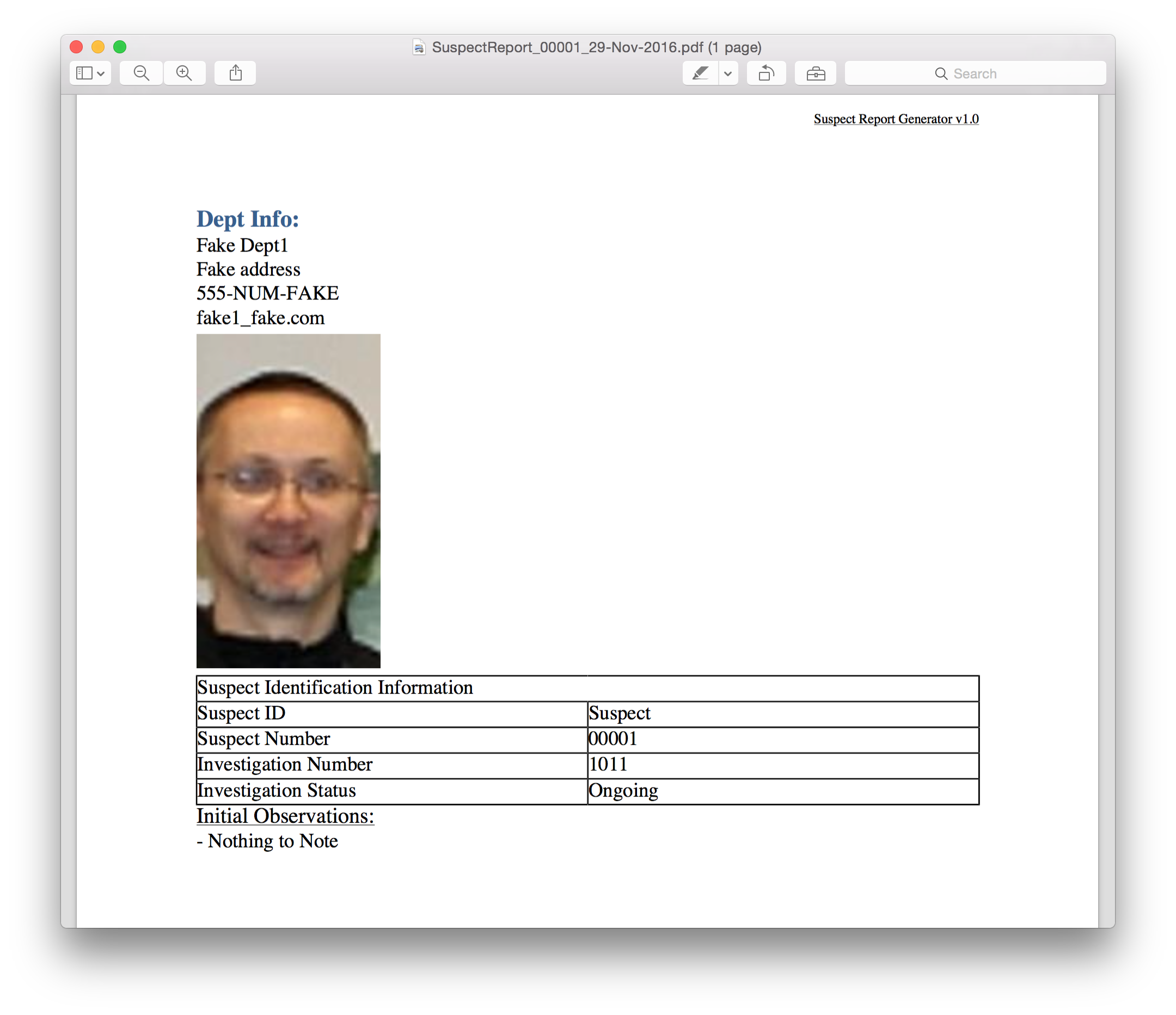
**Figure 6:** Result of detecting face in video frame using ‘tilted\_face.avi’ provided by MATLAB.



**Figure 7:** Result of detecting multiple faces using “vision\_team.jpg” provided by MATLAB. As can be seen, the system is able to recognize multiple faces within a single image frame and annotates them in an indexed fashion for user selection.

**

**Figure 8:** Result of image extraction after user selected “Face 1” for report generation. Although an edge-sharpening filter is applied prior to face extraction, features can become blurred during resizing if the face detected in the image is particularly small. Bi-cubic Interpolation was used for pixel resampling.

**

**Figure 9:** Basic sample of suspect report.



**Figure 10:** An interesting case in which the yellow colored masked man is identified. A mouth and eyes are clearly identified in every face but the color intensity of the yellow mask allows for facial recognition. Generally unhelpful should a suspect wear any other color of mask but an interesting case nonetheless.

**

**Figure 11:** An interesting example of failed facial recognition. Failure could be a result the pixels in the image all having nearly the same intensity.

**Computer Vision Methods and Tools**

The key piece of this application is the success or failure of the facial recognition portion of the algorithm. For this reason I utilized the MATLAB Computer Vision Toolbox vision.Cascade.ObjectDetector system object and step function ("Detect Objects Using The Viola-Jones Algorithm – MATLAB," 2016) in order to run the Viola Jones Algorithm using a well trained, written implementation of an algorithm that would be both time efficient and result in high level of successful detection. While it would have been possible to code an implementation without using the Computer Vision Toolbox, I required a system that would have a very high success level of facial detection in spite of different illumination conditions, facial hair, hairstyles, glasses, expressions and the resources offered by the toolbox fit my needs perfectly. Additional computer vision methods used in this project revolved around image processing and indexing in MATLAB. The imresize and imsharpen inbuilt functions in the Computer Vision Toolbox were valuable during resizing of the extracted indexed face and indexing techniques used constantly during this course were utilized in both extracting the face and formatting the images in figures one through six. The final tool used during the Suspect Report generation step was the MATLAB Report Generator. Using the Document Object Model API offered by the Report Generator I was able to create and format a PDF report document containing all pertinent information using a custom HTML document part template.

**Known Limitations and Future Extensions**

**Contributions**

I completed this project on my own without any other assistance aside from the resources cited in the above section. Please see all code for particular resource citations.

**Learning**

**Advice**

Below is the list of advice that I would offer to future computer vision students. Please feel free to share this list as an example of how I believe that an undergraduate student should approach this course in order to be successful.

1. Start early on assignments!
2. This class requires a large time allocation. Expect to spend between 15-30 hours on assignments and significantly more time on the final project. Budget your time as best you can and make sure that you understand what you are trying to accomplish. Start with small examples and work up.
3. Office hours are necessary to success but getting individual appointments are more helpful to understand and complete the task. Developing a personal relationship with the instructor is necessary for this course.
4. We did not have available TA’s for this course. Should this change I would also recommend them as a resource.
5. Not going to class is not an option. Helpful hints are presented that can make the difference between success and failure on a programming assignments.
6. Use the Internet to better understand concepts. Power point slides from other schools can contain helpful hints/basic code examples to help you with assignments. Cite every source.
7. The book explains algorithms and concepts at a very high level. This can be good in order to understand the basic concept but makes it more difficult if you are struggling with specific aspects of a topic. See (iii) above for the solution when you are confused.
8. You can succeed in this class if you choose to. However, it is easy to feel overwhelmed in both lectures and during assignments. The professor is there to help you succeed and you will realize that 15-20 minutes of office hours can make the difference in understanding and completing an assignment.
9. There is a definite sense of accomplishment that comes with completing an assignment in this course because in every case you end up with a working implementation of a useful application of computer vision. Take pride in this and enjoy the exploration process that will inevitably take place while completing an assignment.

**References**

"Detect Objects Using The Viola-Jones Algorithm - MATLAB". *Mathworks.com*. N.p., 2016. Web. 29 Nov. 2016.

"Face Detection And Tracking Using The KLT Algorithm - MATLAB & Simulink Example". *Mathworks.com*. N.p., 2016. Web. 29 Nov. 2016.

Spacek, Dr. Libor. "Face Recognition Data". *Csessex.ac.uk*. N.p., 2016. Web. 29 Nov. 2016.

Viola, Paul and Michael Jones. "Rapid Object Detection Using A Boosted Cascade Of Simple Features". *ACCEPTED CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION 2001* (2001): 1-9. Web. 29 Nov. 2016.

**Appendix A**: **Data**

From the Face Recognition Data set. Designed and maintained by Dr. Libor Spacek of the University of Essex.

|  |  |  |
| --- | --- | --- |
| **Faces94 Facial Recognition: Lowest Difficulty** | | |
| Background Green Screen | | |
| Head Scale: none | | |
| Head turn/tilt/slant: minor changes | | |
| Position of face: minor changes | | |
| Image lighting variation: none | | |
| Expression variation: considerable | | |
| **Subject** | **Images** | **% recognition** |
| anpage (f) | 20 | 100 |
| asamma (f) | 20 | 100 |
| elduns (f) | 20 | 100 |
| 9326871 (m) | 20 | 100 |
| admars (m) | 20 | 100 |
| dioann (m) | 20 | 100 |

|  |  |  |
| --- | --- | --- |
| **Faces95 Facial Recognition: Medium Difficulty** | | |
| Background: Red Curtain | | |
| Head Scale: Large variation | | |
| Head turn/tilt/slant: minor changes | | |
| Position of Face: some translation | | |
| Image lighting variation: significant | | |
| Expression variation: some variation | | |
| **Subject** | **Images** | **% recognition** |
| adhast (m) | 20 | 100 |
| boylee (f) | 20 | 100 |
| darda (m) | 20 | 100 |
| gstamo (m) | 20 | 95 |
| jserai(m) | 20 | 85 |
| lidov(m) | 20 | 95 |

|  |  |  |
| --- | --- | --- |
| **Faces96: Highest Difficulty** | | |
| Background: Complex glossy posters | | |
| Head scale: large variation | | |
| Head turn/tilt/slant: minor variation | | |
| Position of Face: some translation | | |
| Image lighting variation: significant | | |
| Expression variation: some variation | | |
| **Subject** | **Images** | **% recognition** |
| acatsa (m) | 20 | 100 |
| cprice (m) | 20 | 100 |
| dfarre (m) | 20 | 100 |
| jcgrij (m) | 20 | 100 |
| jlrums (m) | 20 | 100 |
| shpill (m) | 20 | 100 |
| mystery (m) | 20 | 65 |

|  |  |  |
| --- | --- | --- |
| **Grimace Facial Recognition: Highest Difficulty** | | |
| Background plain, very poor lighting | | |
| Male/Female | | |
| Head turn/tilt/slant: Considerable Variation | | |
| Face Position: Some translation | | |
| Expression Variation: Extreme | | |
| **Subject** | **Images** | **% recognition** |
| Will (m) | 20 | 100 |
| And (m) | 20 | 100 |
| Ant (m) | 20 | 100 |
| Chr (m) | 20 | 100 |
| Dah (m) | 20 | 100 |

**Appendix B: Code**

\*Application was written using MATLAB version 2016a utilizing functions and classes from the Computer Vision Toolbox and MATLAB Report Generator.

\*All code may be found in /FinalProject folder

1. **userInterface.m:** A MATLAB script to create a menu driven application to allow user to select their department information, upload an image or video file, and perform facial recognition for the purpose of generating a Suspect Report PDF document.
2. **findFace.m:** A function that performs facial recognition on either an image or video *frame*, calculates the bounding box coordinates around all detected faces, and annotates and displays to the user the *frame* with bounding boxes around the detected faces for user selection.
3. **formatFace.m:** A function that takes as inputs the un-annotated *frame* and the matrix containing the bounding box coordinates of the detected faces and returns the extracted indexed face as the *reportPicture*.
4. **generateReport.m:** A function that takes as inputs *deptInfo* and *reportPicture* and collects user input regarding the suspect identified in the image or video frame. This information is stored in *suspectInfo*.
5. **reportBuilder.m:** A function that takes all user input from *generateReport.m* and returns a formatted a PDF suspect report containing *suspectInfo*, *deptInfo*, and *reportPicture*.
6. **testFacialRecog.m:** A function to run a series of tests on the facial recognition portion of the application using an altered version of findFace.m. Runs statistics on recognized faces in image sets, creates mosaics of image frames with annotated facial recognition frames,
7. **usage\*.m:** Three simple functions to print usage instructions to command window for user.
8. **myPDFtemplate2.pdftx:** Contains HTML and CSS styling templates to format the PDF Suspect Report. Utilized in reportBuilder.m.