Object Speed Tracking

Tech Review

Alex Bailey, Ben Wick, Dylan Washburne

CS 461, Fall Term

Abstract

While we could consider whether we would use a singular camera or multiple, there is really no debate here. Stereoscopic cameras carry such an advantage for a project like this, we are making that decision.

CONTENTS

I Introduction 2

II Technologies 2

II-A Live Data Feed . . . 2

II-B Synchronization . . . 3

II-C Long-Term Compression . . . 4

II-D Long-Term Storage . . . 5

II-E Computer Vision Library . . . 6

II-F Object Tracking Methods . . . 7

II-G Live Compression . . . 8

II-H UI Overlay . . . 9

II-I Video to Velocity Formulas . . . 10

III Conclusion 11

References 12

I. Introduction

This document is a detailed review of multiple technology options considered for our project. Each piece of technology has three options that is being compared. It includes reviews of live data feed, synchronization, and long-term compression written by Alex Bailey. Also reviewed, are options for long-term storage, computer vision libraries, and object tracking algorithms written by Ben Wick. Finally, written by Dylan Washburne, includes live compression, UI overlay, and video to velocity formulas.

II. Technologies

A. Live Data Feed

The options for Live Data Feed are processing both images simultaneously, both images plus distance information, and a single image with distance information. The goal for this piece is how the information should be passed into the velocity algorithm. The velocity algorithm will need to know how the object is moving perpendicular to the camera as well as how it is moving parallel to the camera. Because of this, the algorithm will need to know the distance the object is from the camera and whether or not it is changing as the object moves. There are a couple criteria for this piece. The first is simplicity and ease of use, as we have a limited timeframe for this product. Second is the amount of data that is being transmitted. The more data being transmitted, the longer the process will take.

For processing both images simultaneously, the program will pass the images from both cameras to the velocity algorithm and the velocity algorithm will have to compute the distance to the object as before computing the velocity of the object. Computing the distance will add some complexity to the this method, but not a lot, as the distance will have to be calculated at some point, but this may not be the most efficient place for this computation. The amount of data being transferred is significant because 2 video feeds is a lot of information, but the lack of the extra distance information means that it is not the worst it could be.

For processing both images plus distance information, the program will pass the images from both cameras to the velocity algorithm as well as the distance to the object. Having the distance computed before calculating the velocity will reduce the complexity of this piece by a fair amount, though the distance will still have to be calculated at some point. The amount of data being transferred is very significant in this option as two video streams is a lot of data and then the distance information of top is a lot.

For processing a single image with distance information, the program will only pass the video feed from one of the cameras, probably an arbitrary choice of always the left or the right camera, to the velocity algorithm. With the distance already being computed this will have a small complexity as the algorithm will only need to compute the velocity. The amount of data being transmitted will be significantly less than the other two options, having only one video feed with extra data.

Option	**Difficulty**	**amount of data**
2 Image	Hard	Medium
2 Image and Data	Easy	High
1 Image and Data	Easy	Low

Because the single image with distance information is tied for the lowest difficult but also has a significantly less data transmission it is the best choice

B. Synchronization

The options that we will be looking at for this technology are having two cameras and having no synchronization, having two cameras and using a global shutter to synchronize the camera, and using a stereoscopic camera which will self-synchronize. The goal of this piece is to decide the best method for synchronizing, or not synchronizing, the two cameras we will be using the track the objects. This is important because we will be performing calculations based on the images from both cameras. Since the objects we will be tracking are moving, we will need to know if the two images were taken at the same time. If the cameras are synchronized, then we can perform calculations knowing the object is in the same place. If the cameras are unsynchronized then the objects will be in different locations and we will have to take that into account. The main criteria that our group will be looking at is ease of use. Due to our projects limited timeline, we need a method that will be simple and fast to pick up.

The first method is have to two cameras with no synchronization. If we were to use this method then we would have to take into account the difference in position between cameras. Depending on the velocity of the type of object this could be significant. If we choose people as our object, this difference will be fairly little. However, if we to use vehicles as our object, then the difference will be more significant. This will require a fairly complicated algorithm to account for the difference in either case. The framerate of the camera will also play a role in this. The lower the framerate the greater the amount that the two cameras can be off.

The second method is to have two cameras that have a global shutter, most likely set by the computer controlling the cameras. This could be fairly difficult. Depending on the type of camera, it may not be possible to set the shutter from the computer running the cameras. This would mean that we would need to attempt to start them at the same time and hope that there is no variation in the real shutter speed, otherwise there could still be variation. Alternatively, we would have to find a hardware work around, but as our group has limited knowledge of video camera hardware, this would be very difficult, especially with our limited timeframe.

The third method is to use a stereoscopic camera. This would be a simpler solution as the camera itself will deal with the synchronization of the shutters. However, this would come at the price of not being able to choose how far apart the camera lenses are. Though there could be other benefits, depending on the camera.

Type	**Difficulty**
No Sync	Hard
Global Shutter	Medium
Stereoscopic camera	easy

Due to us needing the easiest solution to implement, the stereoscopic camera is the best choice.

C. Long-Term Compression

The options that we will be looking at are Xvid, FFV1, and OpenH264. The goals for this piece of the project is to compress the video after it has been displayed to the user, to be kept for long term, should they be needed at a later date. With our product, it is likely that the user will be leaving our product running for a significant amount of time, possibly hours. When this happens, video files can become rather large. This will become a problem for long term storage if our product is used often. So, the answer to this problem is to use a compression encoder/decoder, codec. This will reduce the size of the video as much as possible. There are several factors to consider when looking at video compression codecs. The first is whether or not it is lossless. When compressing information, especially pictures and videos, the compression codecs will often save space by removing data, often in the form of merging pixels, leading to a lower resolution, these are called lossy codecs. While a lossless codec is obviously preferred, there are not many lossless video compression methods available and they often have limited compression. Lossy codecs generally offer a greater reduction of file size, with the amount of data lost, depending on the codec. This could be an acceptable tradeoff, depending on the size of the file and amount of quality lost. Second is the amount of compression, often expressed as a ratio. This is the ratio of the size of the original video file to the size of the compressed file. This can be difficult to judge as some videos compress better than other depending on what is being recorded. Third, price is almost always a factor, as it is in this case. If there is an open source alternative that is comparable to a paid version, then the open source would be more favorable.

Xvid is a an open source codec alternative to a commercially sold codec, DivX [1]. Xvid claims to be able to ”compress video at a ratio of 200:1 or more” [long2]. While impressive, this is likely only under certain conditions. Xvid is a ”’lossy’ compression but aims at removing just those picture details that are not important for human perception” [2].

FFV1 is a lossless video codec that is a part of FFmpeg a ”leading multimedia framework, able to decode, encode, transcode, mux, demux, steam, filter and play pretty much anything that humans and machines have created” [3]. Two sources have reported FFV1’s compression ratio as roughly 100GB per hour to 45-50GB per hour [4], approximately 2:1, to roughly 1.2:1 to 2.5:1 [5].

H.264 is a video codec that was created by International Telecommunication Union [6], that has become ”an industry standard for video compression” [6]. Cisco has recently decided to release an open source version under a BSD license and cover the royalties for anyone using their binary files [7]. While this would allow our product to use the H.264 codex, it would restrict us in the need to only use their binaries and always keep them up to date. Should there be an issue, this could cause significant legal trouble. H.264 has a lossless version that has a ratio of roughly 2:1 [8].

Codec	**Lossless**	**Compression Ratio**	**Open Source**
Xvid	No	200:1(theoretical)	Yes
FFV1	Yes	2:1 (roughly)	Yes
OpenH264	Yes	2:1 (roughly)	Yes (with caveat)

While Xvid has the best compression ratio, since it is not lossless, it is not the best choice. OpenH264 has the roughly the same abilities as FFV1, but because of the potential legal problems and potential royalty payments this is also not the best choice. This means that FFV1 is the best option for the long term compression codec.

D. Long-Term Storage

The three option we have considered for video storage is Audio Video Interleave (AVI), Matroska Multimedia Container (MKV) and MP4. The goal of the storage is to use the best storage container based off of features and for our user needs. The container used is important because we plan on storing the videos so the user is able to go back and view the video feed. This however, creates problems for storage because raw video files can tend to be large in size and storing all of our video would just be impossible. Once the video is compressed with a codec, the container is going to be used to package the video. We are looking for a container that is capable of handing compressed files and storing them.

AVI is an older container created by Microsoft. It features support almost any video format and audio format. Because it is an older container, it does lack a few features. The only supported devices are Microsoft devices. This means it won't be able to play on any apple device. AVI also tends to be larger than most video formats because of the lack of video compression features [28].

MP4 is another container that is developed by the Motion Pictures Expert Group. The videos inside MP4 files are encoded with H.264 [29]. MP4 is compressed using AAC encoding or lossy which allows for great storage [29]. MP4 is compatible with devices like IPad, IPod, Android Phone and many others which gives it an edge on MKV and AVI [29]. This reason alone makes it usually one of the most used containers. MP4 is also capable of handling high quality videos. However MKV tends to win in that category.

MKV is another container that is used that can hold an unlimited number of video, audio, picture or subtitle tracks that is developed by CorCode, Inc [30]. MKV does a great job of handling high quality video [30]. It is very flexible a It can't be played on portable devices including phones and tablets [30]. However, MKV does support H.264/AVC which creates for efficient HD content playback.

	Video formats supported	**Codecs supported**	**Files size**
AVI	Almost anything	DivX, Xvid, Cinepak, Indeo, DV and Motion JPEG	Tend to be larger
MP4	MPEG-2 Part 2, MPEG-4 ASP, H.264/MPEG-4 AVC, H.263, VC-1, Dirac	AVC M	Smaller than both AVI and MKV
MKV	Almost anything	Almost any	Tend to be very large

Choosing a container is a little challenging because many offer different benefits. MKV and AVI tend to have larger file sizes. However, this means it's quality is lot higher. Because we aren't necessarily looking for super high quality video playback, this may not be the best options for us. MP4 is also better supported on devices. This is a huge positive as it creates less chance for issues when playing back video. This means that MP4 is the best option for us.

E. Computer Vision Library

The three options for Computer Vision (CV) libraries include OpenCV, LTI-Lib, and VXL. Selecting a good CV library is essential for our project. The goal of the CV library is to provide us with a large array of functions that we are able to utilize. Every CV library we are looking at is open source and is free to use. The library selected must be able to support our needs of being able to identify and track objects in real time through our live video feed. It must have many available tools to simplify the identification and the tracking of objects. The criteria that will be evaluated are languages available, features available, and performance. The languages available is important because choosing a language we are already familiar with will give us an advantage. We are hoping to use C/C++. Feature available is also another very important criteria. Specifically, we are looking for features that are able to simplify object tracking and detection. Performance also plays a huge large factor in our choice. We are looking for library that is as efficient as possible. These libraries tend to be written in C/C++.

OpenCV is one of the most commonly used libraries for computer vision. The OpenCV libraries are written in C/C++. According to their website, they offer over 2500 algorithms [18]. The purpose of OpenCV is to offer a large number of function that the user is able to use to simplify difficult tasks. This includes algorithms that are capable to identify and track objects, recognize faces, follow eye movements and many more [18]. OpenCV is used by many large companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda and Toyota [18].

LTI-Lib is also another Computer vision library that has many algorithms and features. LTI-Lib's main goal is to provide an object oriented library in C++ [19]. It includes libraries for linear algebra, classification and clustering, image processing, as well as visualization and drawing tools [19]. There are a few research projects that utilize LTI-Lib including one at the University of Liege, Belgium where they do many things like machine learning, medical imaging, radar imaging, as well as sports video analysis.

VXL is another great computer vision library utilized by a lot of people. Similar to OpenCV, the VXL libraries are written more in C++ [20]. The main libraries within VXL include numerics (VNL), imaging (VIL), geometry (VGL), and streaming I/O (VSL) [21]. They also have libraries for image processing, camera geometry, stereo, video manipulation, 3D imaging and many more [21]. VXL is written to be light, fast, and portable over many platforms [vision5].

	Languages available	**Features available**	**Performance (rank)**
OpenCV	C, C++, Java, Python	Many	1
LTI-lib	C, C++	Many	2
VXL	C, C++	Numerics, Imaging, geometry, streaming I/O	3

Based on the criteria needed for computer vision library, OpenCV has a large number of algorithms that we will be able to use as well as performs faster than the other libraries. The speed performance test was done by Utkarsh Sinha. The comparisons done were 2D DFT, resizing, optical flow, and neural net [22]. His findings show that OpenCV performs faster than both LTI-Lib and VXL with LTI-Lib coming in second [22]. Also, Based on reviews, OpenCV is just a lot more popular and widely used. This is a huge positive because it means more references and documents will be available for use. For our project I think OpenCV would be a lot more useful based on the criteria tested.

F. Object Tracking Methods

The three options for object tracking algorithms are Haar Cascades, background subtraction, and Speeded Up Robust Features (SURF). Each method can be implemented using OpenCV libraries. The goal of each method is to offer an accurate and efficient way of identifying an object. To compare each option we will look at accuracy and speed of the method. Accuracy is a crucial criteria because a method that does not correctly identify an object will fail to calculate the speed. We are looking for the best option that will not produce a large number of false-positives as well as miss objects. The speed of the method is also very important because the method used should be capable of detecting an object that is moving quickly through the video feed.

Haar Cascades is a great object detection method. It is capable of facial recognition, vehicle tracking, animal tracking, and many other objects [24]. Haar cascades is most accurate with facial recognition with a precision rate of 80 percent [24]. However, vehicle tracking precision is sub optimal with a precision rate of just below 50 percent [24]. The accuracy of cars does go up when viewing cars from the side [24].

Background subtraction is also another common method for object detection. According to documentation on OpenCV's website, "Background subtraction calculates the foreground mask performing a subtraction between the current frame and a background model." This means it is able to detect the background and then use that to detect the moving object. This is done in two steps, background initialization, and background update [25]. This method is easy to apply when an image of the background alone is available [26]. Because the method relies on moving objects, objects that are moving more slowly have a higher chance of error. Shadows can also cause error because shadows move as well but can be marked as foreground [26].

Speeded Up Robust Features or SURF is another method capable of facial recognition and car detection. SURF relies on an image to determine key points. These key points are then used to detect the objects on the live feed. This means it will rely on a selection of images to create key points and then compare it to the live video feed to recognize objects, so creating a library with enough images is crucial to increase accuracy.

	Acuracy	**Speed**
Haar cascades	Accurate for facial recognition, less accurate for car detection.	Very fast
Background subtraction	Accurate for fast moving objects	fast
SURF	Accurate (%80)	Slightly slower than both Haar cascades and background subtraction

This option is a little more difficult to choose from. All options are viable and have positives as well as negatives. The accuracy testing was done in a research paper testing for the speed and accuracy of Haar cascades and SURF [27]. The deciding factor is whether accuracy is more important than the speed of the detection. As long as the method is capable of detecting fast moving objects with a low rate of error, then accuracy should be the main deciding factor. Because Haar cascades is more accurate for facial detection, and SURF being less efficient, our best bet is to explore the background subtraction method.

G. Live Compression

When the program is running, the camera will be actively generating data. Videos by their basic nature require large amounts of data to be transferred. A method to reduce the load of the video passing through is to compress it at some step of the way. However, compression carries with it potential dips in video quality which may be required to accurately identify objects.

The most basic approach to the problem is not to compress the video at all. This ensures the entire image is received by the computer which is doing the processing. This however is not an ideal solution, because while it is receiving the maximum quality image, it could also has to transmit the entire image to the display port, which often is smaller than the resolution the video was taken at. This also means that while it has the highest visual fidelity, it has to parse that much more imagery before it can spit out a response [15].

A second method to consider is that the computer does video compression in addition to its other duties, every frame. This offers the option to output significantly less imagery to the display port each frame, at no loss to incoming video. Alternatively, the computer can also compress the image before it begins its scans to lessen the amount of work required to scan the entire image. It is fully possible that the image will not degrade in a significant enough manner under compression that it would affect the object recognition [16].

If the image getting compressed would not affect the object recognition, then it would be beneficial to offload as much processing the computer would perform as possible. To this end, it is possible that modern video cameras can compress the frames before sending the images back to the computer. Doing so would offload a decent amount of stress from the computer when processing the image, as well as when receiving the data from the camera in the first place [17].

Name	**Processing Strain**	**Image Quality**	**Speed**
No Compression	High	High	Slow
Computer Compression	Very High	Moderate	Moderate
Camera Compression	Low	Moderate	Fast

No compression, while the ideal of receiving perfect image quality is certainly a respectable endeavor, is unfeasible for this project. In addition, the quality loss from compression does not seem to my human eye to be significant in any measure. To that end, it seems most beneficial to move as much strain off the computer as possible and, therefore, attempt to get a camera which will automatically compress the frames before they're sent for processing.

H. UI Overlay

The UI overlay we select can be generated by one of the three following software packages: OBS, Camtasia, or XSplit. The goals for this software is to run with minimal overhead because it must perform its function while the backend is also doing the heavy lifting. A major function we require from the selected software package is the ability to create an overlay on top of the incoming video feed. In addition, depending on how the backend is set up, we will also need the overlay to place boxes around positions specified by the backend. This is an effort to show the user where identified objects are in the image feed.

Open Broadcaster Software, frequently referred to as OBS, is an open source project designed for use in video editing and streaming. OBS is also currently the most used software for live-streaming video feeds over the internet. It is made to be as lightweight as possible while providing all required power to the user for the purposes of editing the video feed [12].

Camtasia is, at its core, a video editing software. Since the rise in popularity of live streaming, Camtasia's creators (TechSmith) have added the capability to support live video streams. They have an extensive toolset to allow you to manipulate what is shown on-screen due to their roots in video editing, but that also comes at the cost of marginally more overhead when running compared to other modern solutions [13].

XSplit is a software designed primarily for use in web streaming a video. As a result, it is designed to be run in the background while more strenuous tasks simultaneously run on the computer. Popular add-on packages allow it to generate pop-in elements on the overlay, primarily for Twitch donation notifications, but it can potentially be retooled for dynamic placement [14].

Name	**Overhead**	**Dynamic Overlay Placement**	**Open Source**
OBS Studio	Low	Yes	Yes
Camtasia	Moderate	Yes	No
XSplit	Low	Yes	No

Based on the capabilities of review technologies, OBS Studio is the selection for this project. Camtasia has too much overhead due to its roots as a video editor. XSplit is much more similar in performance reviews, but OBS has more power available to the user. While XSplit can install packages to match this power, each package has an overhead cost, which places OBS on top.

I. Video to Velocity Formulas

In a sense video to velocity formulas already exist, however they exist to view their targets from a known, fixed distance and take advantage of hard-coded values. In that way, we can certainly make use of what has been made, but we will have to heavily modify it for our purposes. We can do so by using one variety of block-matching algorithm. Our options include SDSP, Adaptive Rood, and Phase Correlation. Because we are expecting to receive many frames every second, the computer should be able to calculate the velocities of located objects with minimal overhead and maximum speed.

A Small Diamond Search Pattern (SDSP) is a heavily simplified version of what is referred to as an exhaustive search. In a sense, an exhaustive search is a brute-force application to find the object's motion between frames, in any possible direction. SDSP approximates the boundary conditions of an exhaustive search by searching in 8 directions around an object to where it may have traveled. It repeats this any number of times until it believes the new location is approximately correct. This method has very small variance from an exhaustive search while offering very reliable velocities when set up correctly [15].

The Adaptive Rood Pattern Search is an algorithm that takes an object's previous motion and assumes it will approximately continue to use that motion between frames. This allows it to potentially jump to the answer immediately, though in reality it will self-correct in a large number of cases. This self-correction takes place through the use of a Diamond Search pattern, a variant of which was discussed above, after which it returns a velocity [16].

Phase Correlation is used to determine how far an object has moved on screen by comparing the two images and using the fourier shift transform to locate the change in x and y which the object traveled. This gives highly accurate results when it works, but its reliability comes at the cost of higher processing time required [17].

Name	**Overhead**	**Speed**	**Accuracy**
SDSP	Low	Fast	High
Adaptive Rood	Low	Fast	High
Phase Correlation	Low	Slow	High

Phase Correlation is clearly not an option we will wish to pursue for this project, based on the conditions listed above. The SDSP had the lowest overhead of any method available, however Adaptive Rood was notably faster in many scenarios. It is worth noting Adaptive Rood has parts based around Diamond Search patterns which would make it more advanced than the pattern by itself, however it is questionable whether the objects in frame would move fast enough that we would need to predict their motion. That said, I am selecting the Adaptive Rood method for this project.

III. Conclusion

After researching multiple options for each technology, we believe we have a good start starting point of which option we will need to use create our project. The above information shows what we will prioritize when developing the project.

REFERENCES

[1] B. Clark, 'All You Need to Know about Video Codecs, Containers, and Compression', 2015. [Online]. Available: http://www.makeuseof.com/tag/all-you-need-to-know-about-video-codecs-containers-and-compression/ . [Accessed: 15-Nov-2016]

[2] xvid.com, 'Home', 2016. [Online]. Available: https://www.xvid.com/ . [Accessed: 15-Nov-2016]

[3] ffmpeg.org, 'About FFmpeg'. [Online]. Available: http://ffmpeg.org/about.html . [Accessed: 15-Nov-2016]

[4] E. Lorrain, 'A Short Guide to Choosing a Digital Format for Video Archiving Masters', 2014. [Online]. Available: https://www.scart.be/?q=en/content/short-guide-choosing-digital-format-video-archiving-masters . [Accessed: 15-Nov-2016]

[5] CS MSU Graphics & Media Lab, *Lossless Video Codecs Comparison*, Moscow: CS MSU Graphics & Media Lab, 2004, p.14[Online]. Available: http://compression.ru/video/codec_comparison/pdf/lossless_codecs_test_en.pdf . [Accessed: 15-Nov-2016]

[6] Vcodex, 'H.264 Advanced Video Coding'. [Online]. Available: https://www.vcodex.com/h264-resources/ . [Accessed: 15-Nov-2016]

[7] openh264.org, 'FAQ'. [Online]. Available: http://www.openh264.org/faq.html . [Accessed: 15-Nov-2016]

[8] H. Lewetz, et al., 'Comparing Video Codecs and Containers for Archives', 2013. [Online]. Available: http://download.das-werkstatt.com/pb/mthk/info/video/comparison_video_codecs_containers.html#codec_tests . [Accessed: 15-Nov-2016]

[9] compression.ru. [Online]. Available: http://compression.ru/video/codec_comparison/pdf/msu_lossless_codecs_comparison_2007_eng.pdf . [Accessed: 15-Nov-2016]

[10] microsoft.com. [Online]. Available: https://msdn.microsoft.com/en-us/library/windows/desktop/dd743961(v=vs.85).aspx . [Accessed: 15-Nov-2016]

[11] vcodex.com. [Online]. Available: https://www.vcodex.com/news/how-to-stream-better-quality-video-part-1-basics-and-good-practices/ . [Accessed: 15-Nov-2016]

[12] obsproject.com. [Online]. Available: https://obsproject.com/ . [Accessed: 15-Nov-2016]

[13] techsmith.com. [Online]. Available: https://www.techsmith.com/camtasia.html . [Accessed: 15-Nov-2016]

[14] xsplit.com. [Online]. Available: https://www.xsplit.com/ . [Accessed: 15-Nov-2016]

[15] ncbi.nlm.nih.gov. [Online]. Available: https://www.ncbi.nlm.nih.gov/labs/articles/18255398/ . [Accessed: 15-Nov-2016]

[16] semanticscholar.org. [Online]. Available: https://pdfs.semanticscholar.org/7f78/41b7b9c98b1e1f52282bdc3c2710cf83d3f0.pdf. [Accessed: 15-Nov-2016]

[17] semanticscholar.org. [Online]. Available: https://pdfs.semanticscholar.org/2c6c/6d3c94322e9ff75ff2143f7028bfab2b3c5f.pdf. [Accessed: 15-Nov-2016]

[18] "ABOUT — OpenCV", Opencv.org, 2016. [Online]. Available: http://opencv.org/about.html. [Accessed: 16- Nov- 2016].

[19] "LTI-Lib", Ltilib.sourceforge.net, 2016. [Online]. Available: http://ltilib.sourceforge.net/doc/homepage/index.shtml. [Accessed: 16- Nov- 2016].

[20] "VXL: 1. Introduction", Public.kitware.com, 2016. [Online]. Available: http://public.kitware.com/vxl/doc/release/books/core/book_1.html. [Accessed: 16- Nov- 2016].

[21] "VXL - C++ Libraries for Computer Vision", Vxl.sourceforge.net, 2016. [Online]. Available: http://vxl.sourceforge.net/. [Accessed: 16- Nov- 2016].

[22] "Computer Recognition System For Detecting And Tracking Objects In 3D Environment", Csus-dspace.calstate.edu, 2016. [Online]. Available: http://csus-dspace.calstate.edu/bitstream/handle/10211.9/1249/Introduction%20to%20computer%20vision.pdf?sequence=2. [Accessed: 16- Nov- 2016].

[23] U. Sinha, "OpenCV vs VXL vs LTI: Performance Test - AI Shack - Tutorials for OpenCV, computer vision, deep learning, image processing, neural networks and artificial intelligence.", Aishack.in, 2016. [Online]. Available: http://aishack.in/tutorials/opencv-vs-vxl-vs-lti-performance-test/. [Accessed: 16- Nov- 2016].

[24] "Object Detection Using Haar-like Features" 2016. [Online]. Available: http://www.cs.utexas.edu/ grauman/courses/spring2008/slides/Faces_demo.pdf. [Accessed: 16- Nov- 2016].

[25] "OpenCV: How to Use Background Subtraction Methods", Docs.opencv.org, 2016. [Online]. Available: http://docs.opencv.org/trunk/d1/dc5/tutorial_background_subtraction.html. [Accessed: 16- Nov- 2016].

[26] "OpenCV: Background Subtraction", Docs.opencv.org, 2016. [Online]. Available: http://docs.opencv.org/3.1.0/db/d5c/tutorial_py_bg_subtraction.html. [Accessed: 16- Nov- 2016].

[27] "Learning SURF Cascade for Fast and Accurate Object Detection"2016. [Online]. Available: http://www.cv-foundation.org/openaccess/content_cvpr_2013/papers/Li_Learning_SURF_Cascade_2013_CVPR_paper.pdf. [Accessed: 16- Nov- 2016].

[28] "AVI vs MP4 , difference between AVI and MP4 format", Iorgsoft.com, 2016. [Online]. Available: http://www.iorgsoft.com/compare/avi-vs-mp4-comparison.html. [Accessed: 16- Nov- 2016].

[29] "All You Need to Know about Video Codecs, Containers, and Compression", MakeUseOf, 2016. [Online]. Available: http://www.makeuseof.com/tag/all-you-need-to-know-about-video-codecs-containers-and-compression/. [Accessed: 16- Nov- 2016].

[30] L. Case, "All About Video Codecs and Containers", TechHive, 2016. [Online]. Available: http://www.techhive.com/article/213612/all_about_video_codecs_and_containers.html?page=2. [Accessed: 16- Nov- 2016].