Tracking Humans Using a Distributed Array of Motorized Monocular Cameras.

Abstract: *Blob tracking was performed on two concurrent PTZ video feeds to graph motion history. Tactics to bypass proprietary protocols used with PTZ are discussed.*

With our software you are able to minimize operator time by employing the power of a computer to automatically track humans moving within an area with a distributed monocular motorized camera array. Movement can then be represented on a historical perspective in which paths are highlighted; enabling the viewer to make informed placement decisions. Additionally, the system would minimize the learning curve for a distributed array of motorized cameras by maintaining field of view data for all cameras.

Target Audience:

The primary target of the application would be for surveillance for both security and athletic entertainment. Possible surveillance applications range from theft prevention systems used in retail environment to locating infant children at home. The system could also be optimized to track non-human objects such as motor vehicles.

Equipment used

Our solution will utilized existing motorized network cameras on the ISU campus. The software is written to run on any machine equipped with OpenCV. No actions were made to comprise the cameras for future use. The cameras are Canon VB-50i.

Approach

Canon webcams are serviced with a software package called WebView. Webview uses a propitiatory protocol called WV-TCP/HTTP. WV protocol encapsulates the video, audio, coordinates and requests between the camera and a client. Given is proprietary nature we were left with two avenues. Acquire and use software specifically written to interact with the camera or develop our own. Clearly we choose to purse locating a download of the Canon

Three tactics where used to acquire the Canon Video LAN SDK kit. First the web-based request form was entered. Then a follow up email was to customer support asking why I didn't receive my authentication email. Then several calls where placed to customer support. Finally several calls were placed to sale representatives with the notion that a perspective customer wanted to preview the SDK prior to purchasing sixteen cameras. As with the previous emails every attempt failed. The requests would be forwarded to someone else who 'would be able to help'. Eventually the request would reach someone who either didn't reply or didn't know what to do.

Additional attempts to proceed in difficulty. Reverse engineering the dynamically linked libraries included with the publicly available Client Viewer, screen wrapping the Client Viewer and finally screen casting the Client Viewer. The Client Viewer provides non-authenticated client to control the cameras for a short period of time. The user is able to see the PTZ coordinates, video stream, and audio stream.

Reverse Engineering DLLs

In an effort to obtain programmical control of the Canon webcam attempts were made to decompile the dynamically linked libraries. DLL or dynamically linked library is a standard for precompiled functions to be shared between processes. Often DLL libraries are non-encrypted; they can then be decompiled to the logical equivalent of original source code. However

knowledge of the original code is desired since the absence of names and documentation make readably not intuitive. The only information I had was seven filenames similar to 'LVTCPCamCon.dll'. Hence this approach endured for just a week before moving on.

Create own network stack with PCAP Library

PCAP is a library used to for encapsulating and decapsulating data in packets. My plan was to sniff enough packets that I could determine the protocol and then create my own interface to manipulate the camera. Analysis of the packets exchange revealed no obvious structure. The real-time nature of these packets made decoding them difficult since each packet contained a continually changing timecode and ID.

Screen Wrap Client Viewer

Screen Wrapping the Client Viewer would entail capturing the video on screen and using keyboard shortcuts to send commands. VideoLan commonly known as VLC is capable of screen casting, capturing video from on screen and then transcoding to another format. WebView Client Viewer accepts keyboard shortcuts: arrow keys, + (plus) & - (minus) for pan, tilt, zoom. However the problem with this approach would be acquiring confirmation that a command to move the camera was successful. To do this we would have to read the PTZ coordinates then covert them to text with an OCR. Hence the program would be highly inefficient, overly complex and of limited value as future editions of WebView are released. Additionally it was learned that C++ does not handle key commands the same as higher languages like VB and java. The interface and type of keyboard affect how commands are sent. Already complete libraries do exist, however this does little to mitigate the complexities of this approach.

Screen Cast Client Viewer

Instead of going directly to a screen wrap. It was decided to start simple and first do a screen cast. VLC was used with the given commands:

```
screen:// :screen-fps=29.001 :dshow-fps=29.001 :nooverlay
:sout=#transcode{vcodec=mp2v,vb=10240,scale=1}
        :duplicate{dst=std{access=file,mux=mpeg1,dst="D:\ouput2.mpeg"}}
```

Then the client viewer was set full screen and the cameras were manually manipulated via the keyboard shortcuts. Although less functional then what the project initially called for, this approach the most results this late in the project. Therefore we decided to hold off on controlling the camera for now and focus on motion history. Until I can complete the PCAP stack or get the SDK.

Motion history

Motion history is the continued detection of a specific objerect over time. To do this OpenCV is equipped with blob detection and face recognition. The problem with face detection is the limited haar cascades. Much of the emphasis with OpenCV was placed on single angle face detection. Instead blob detection works with the edges and surface area of an object independent of a precompiled haar. Therefore by creating an overlay matrix where the object was last located and then adding that to the next frame it is possible to create a history. The problem with this approach would be PTZ movement, but if we know the PTZ coordinates we could mitigate for movement and assume that if the camera moves into a new field of view that the overlay would have to be reinitiated.

Additionally this approach would also provide a bases for motion history and indirectly velocity. The very strengths also served as weaknesses when dealing with noise. To eliminate false

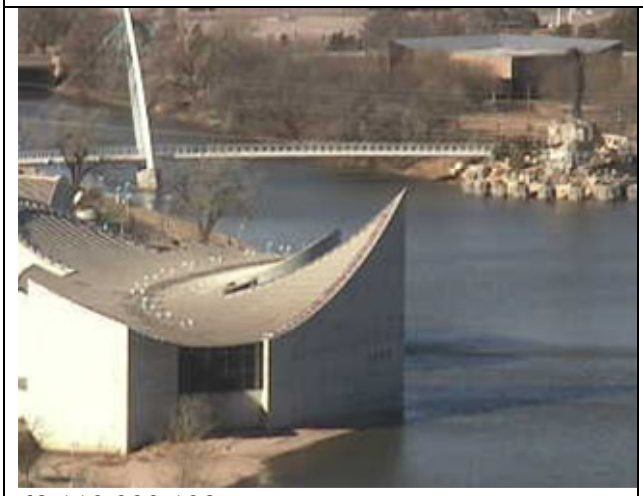
triggers a max velocity was implemented, in which objects that moved more than a specified number of pixels would be rejected.

Alternative Cameras

Among other options we considered were other networked webcams. Network sniffing showed these cameras actively using the wireless network. Their MAC addresses are identified as being Panasonic. Upon initial setup Panasonic cameras ask for a default password.	
129.186.126.8 129.186.93.33 129.186.93.32 129.186.54.202 129.186.54.204 129.186.115.253 10.10.33.49	10.10.52.46 129.186.65.45 129.186.65.41 129.186.126.9 129.186.65.129 129.186.65.117 129.186.235.240

Undisclosed ISU Campus	
 10.10.80.14	

	10.10.80.15

Wichita, Kansas Downtown Skyscraper with cameras on either side	
 68.110.223.198	 68.110.223.198

129.186.95.226

This is one viewing Alumni Hall.

129.186.168.78

This is the Veterinary Medicine building.

129.186.47.55

And this one is of the Memorial Union.

129.186.4.41 and 129.186.8.46

Differences:

When we were planning the project, we intended to have a map of where motion had taken place and have different colors represent the speed of motion of an object. However, in our current

program, we were forced to make it all just 1 color, and the map of all motion is stored in a video file.

Previous Approaches:

Motion tracking is a well developed area of computer vision. Current technology allows for real-time tracking and recognition of a person or object.

The earliest developments which used a distributed array of cameras was in the late 90's. Approaches have sense been developed for both an overlapping and non-overlapping array of fixed cameras. Little development has been made with motorized cameras in a distributed array. Standardization with PTZ functionality has not been implemented.

Q. Cai and J.K. Aggarwal, "Tracking Human Motion Using Multiple Cameras," Proc. Int"l Conf. Pattern Recognition, pp. 68-72, Vienna, Austria, Aug. 1996.

Q. Cai and J.K. Aggarwal. Tracking human motion in structured environments using a distributed-camera system. IEEE Transactions on Pattern Analysis and