“Peace Train”
Vision for the CCR

By Josh Domina

Project Summary

- Use a webcam to identify trains and report their positions in a meaningful way
- I started with a webcam, plenty of code from Professor Blahnik, and access to the CCR room in the PAC

Requirements

- Detect up to 5 trains with cars
- Results displayed analytically or graphically
- Results accurate and delivered in reasonable time ~0.25 sec
- System supports any track configuration
- Noise is filtered
- Broken Trains are detected
- System supports web reporting

Solutions

- Use a logical data structure of “track points” to represent the track
- Any number of trains is permissible so as long as the physical track can support it
- Use motion detection to detect the trains
- Train locations are reported visually and analytically (in terms of track points)
- Running time varies based on lighting
  - Good lighting is ~.15 sec, bad lighting is ~.40 sec
- Noise is filtered as an inherent side effect of the data structure
- Because the track structure is user defined and it is logical, any physical track layout is useable

Exceptions

- Due to the “when you want it” nature of finding trains, identifying broken trains isn’t feasible
- Currently track switches are not supported

Limitatons

- The camera
  - Small aperture
  - Cannot see the entire track
  - Variant frame rates
    - When a room gets darker the camera automatically decreases shutter speed (to allow more light in)
    - Additionally color variety can affect frame rates
    - This results in lower performance
- The Computer
  - The PC equipment available to me (and most PC equipment in general) isn't really powerful enough to process any resolutions above 160x120 quickly

Methodology

- Image Capture
  - How do we get images?
- Track Point Generation
  - How do we generate the points in between user defined track points?
- The Data Structure
  - How is the track represented virtually?
- Motion Detection
  - How do we see if something is moving?
- Train Detection
  - How do we see if this motion is a train?
- Streamlining Algorithms
  - Reduce running time (and quantity) of code

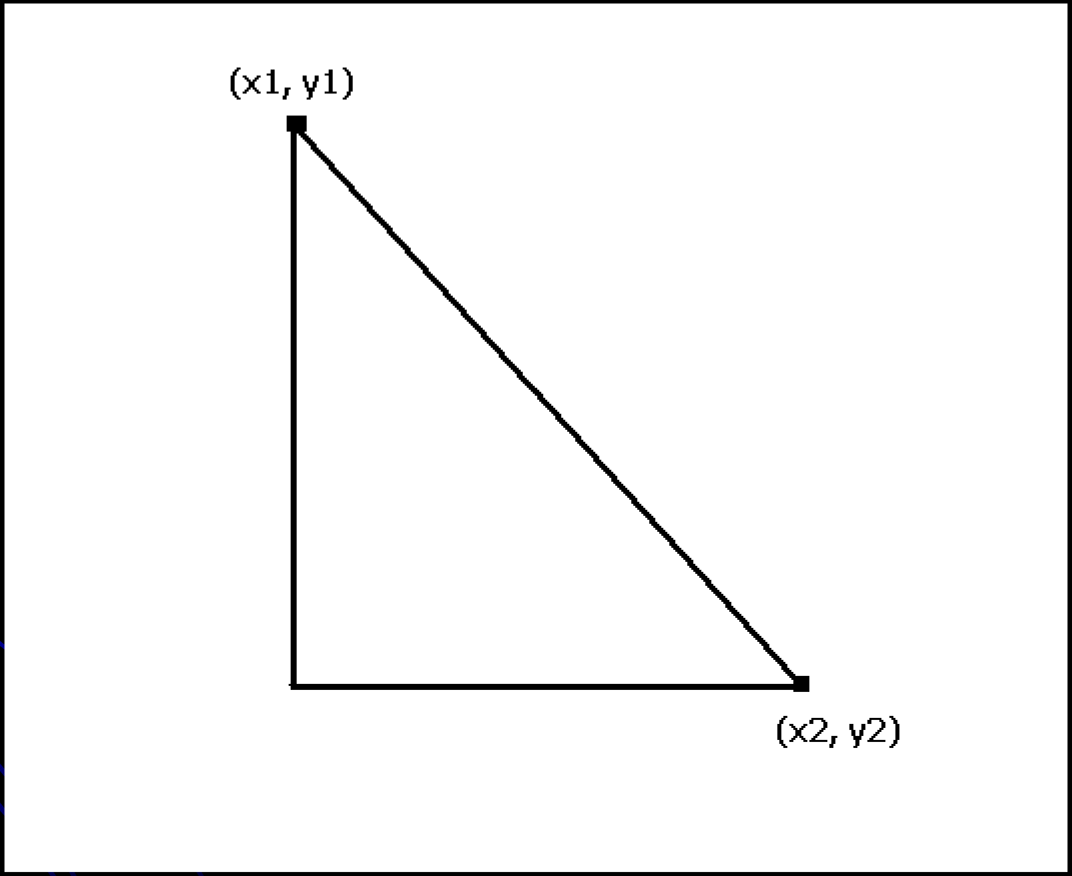
Image Capture

- Done with a frame call back system
- A new frame is grabbed when previous one was processed

Track Point Generation

- We are given 2 points,(x1,y1) and (x2,y2)
  - We need to determine how many track points (if any) fit in between the given points
- We are also given 2 important values
  - Box "Radius" = determines the size of a track point in pixels
  - Box Buffer = distance between points in pixels
- To start we need the distance between the two points
  - Use some basic trigonometry

Track Point Generation



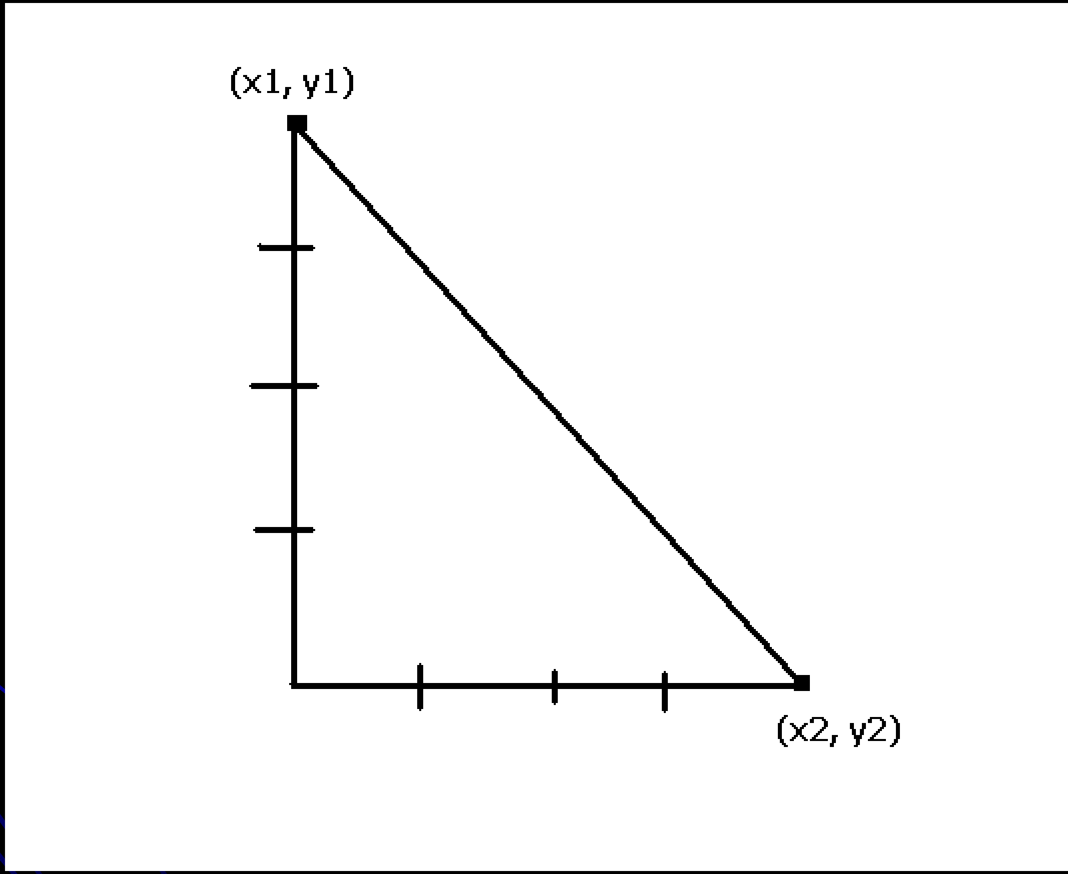
Track Point Generation

- Now we find how many points we will need to add by taking our C (distance between the points) value and using the following equation
  - NewPts = C / (BoxRadius*2 + BoxBuffer*2)

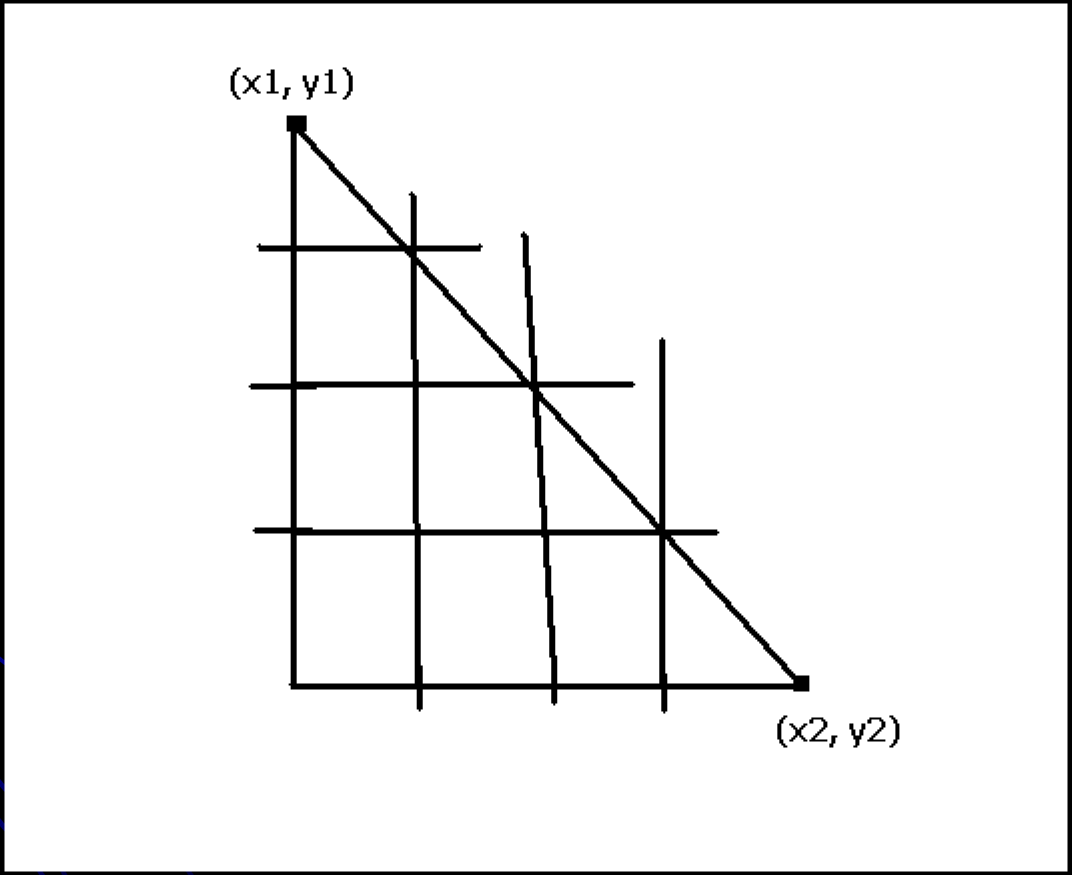
Track Point Generation

- Now we can generate the points
- Split the a and b legs of our “triangle” into NewPts subsections
- Draw horizontal/vertical lines through them
- They will intersect on the hypotenuse, this intersection is the centerpoint of a new track point

Track Point Generation



Track Point Generation



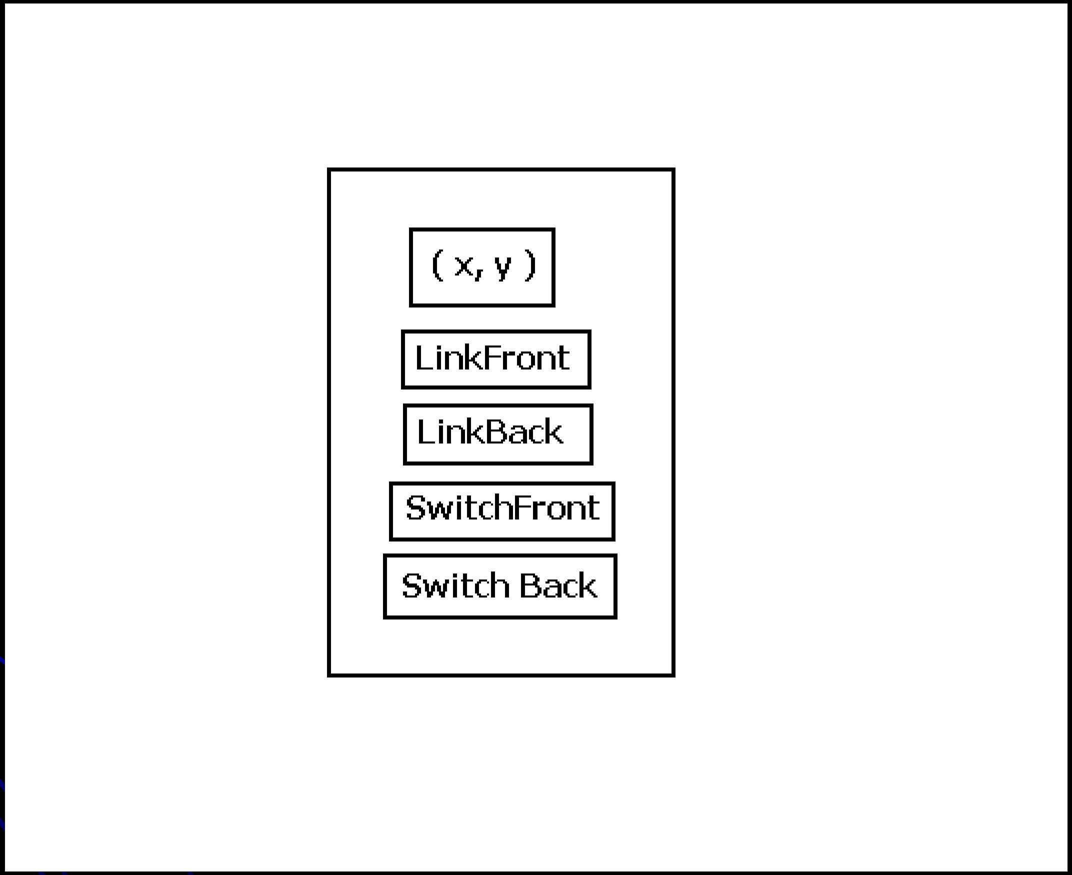
Track Point Generation

- All points are saved to the data structure

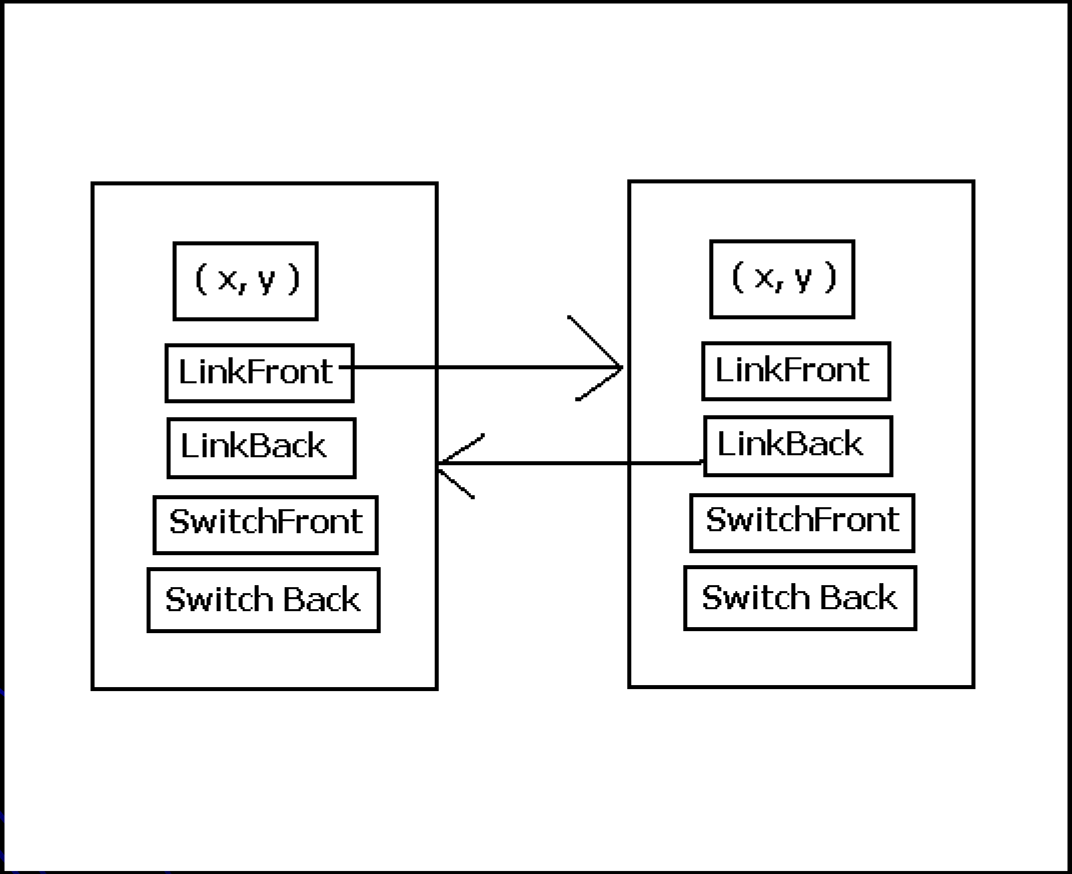
The Virtual Track

- Structure is essentially a linked list represented as an array

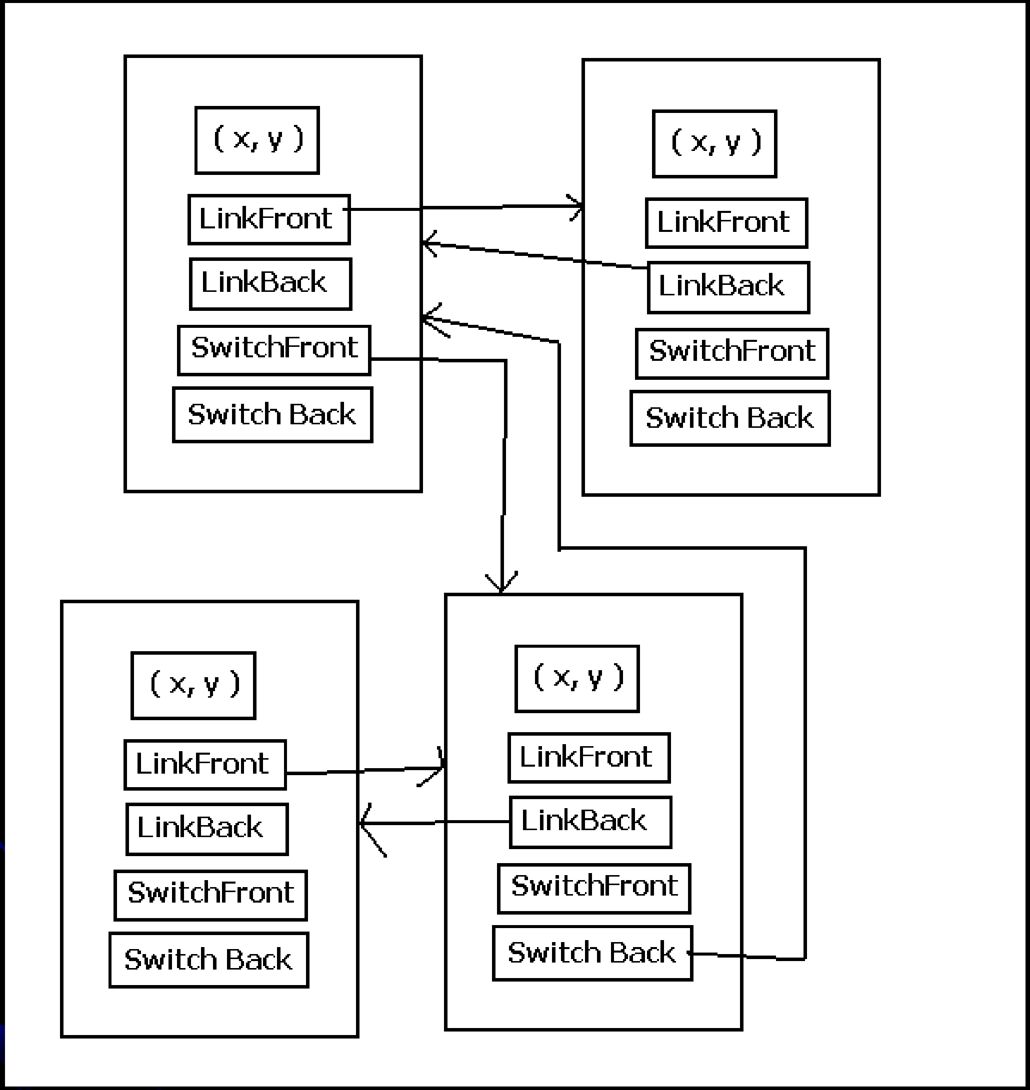
The Virtual Track – A Node



The Virtual Track – Linked Nodes



The Virtual Track – A Switch



Motion Detection

- Save two images (in greyscale) as close to each other in time as possible
- Determine the absolute value of the difference between their greyscale values
- Compare this difference to a threshold
- If it exceeds the threshold then it is motion
- Threshold should be set to a level that balances noise and functionality

Area Motion Detection

- Apply the motion detection algorithm to a small area of an image
- The area used is the size of a track point

Finding Trains

- Start at array entry 0 (the first entered track point) and parse the links calling the area motion detection on that specific track point
- If motion is detected then call another function to find where the motion ends
- The parse function then starts after this end and continues to search for motion
- Trains must be of a minimum length (user-defined)
- Cars are considered to be part of a train

Train Reports

- The virtual track is written out to a file
- Analytical Output
  - Beginning and Ending Track points of a train
- Graphical Output
  - Image of the virtual track motion detected at the point of searching for trains
  - Animation of the trains moving in the virtual track

Streamlining Algorithms

- Because of the need for immediate results the algorithms must run as fast as possible, thus making streamlining of central importance to this project
- Minimize I/O
  - PSet (pixel set) versus JFB's ShowImage
    - PSet writes one pixel to a picture at a time
      - Thus in a 160x120 image you are making 19200 I/O calls
    - ShowImage "blasts" all of the image data in an array to the picture at once
- Remove Recursion
  - Loops execute faster

Demonstration

Strategies

- Modularize Code
  - Its reusable, makes debugging much easier, and makes combining stubs easier
- Stub Programs
  - Test concepts in stub programs
  - Stubs give you a controlled environment
- Trial and Error
  - If you think it could work, code it; you might be right

Strategies

- "Plan to throw one away, you will anyhow" – Fred Brooks *The Mythical Man-Month*
  - I have discarded so many algorithms, ideas, etc. that I have lost count
  - I had much of this project done in another system that wasn't desirable so I went back to the drawing board
  - Expect that some ideas will just not prove to be useful

Useful CS Courses

- CS205/CS220
  - Data Structures
- CS321 Analysis of Algorithms
  - Asymptotic order
  - General concepts
- CS330 Database
  - Good models make for easy code
- CS225 Machine Organization
  - How I/O works and how much it costs
- CS370 Intro to Operating Systems
  - Additional I/O concepts
- All courses
  - Software Design principles

Extensions

- Use more than one camera to be able to see the entire track
- Modify the track parsing to handle switches (already available in virtual track)
- Develop a signaling or messaging system whereby another program (that has the virtual track file) can ask for train location and be sent the trains and their start/end points
- Possibly store additional data in the structure should it prove useful
- Remove the GUI and have the software run as a daemon
- Allow for web reporting

Advice

- As Dr. Pankratz says KEEP IT SIMPLE, do not over complicate things
- Talk to the CS Professors, they can help you solve a problem, get a new idea, or just help you get your bearings
- Keep Dr. Pankratz updated

Any Questions?