How to use the CMUcam4 properly

By: Kwabena W. Agyeman

Overview

- Background
  - CMUcam Project
  - Color Tracking Explanation
  - Lighting Explanation
- Board Connections
- Graphical User Interface
  - How to track colors
- Arduino Interface Library
  - Packet types
- Class Examples
  - For line detection and object tracking

CMUcam Project

- Started in 2000 to produce a low-cost embedded computer vision system

CMUcam1

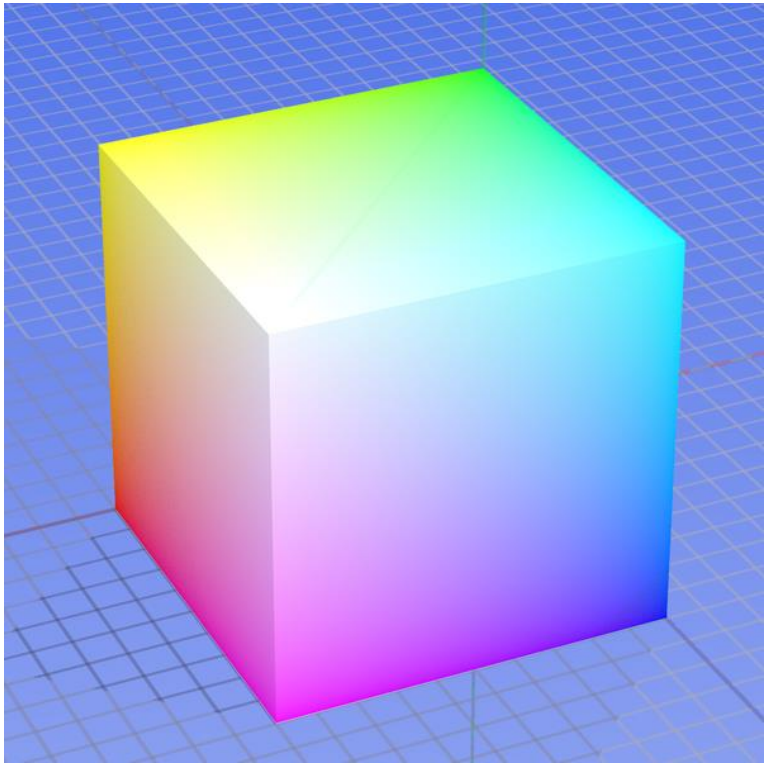
CMUcam2

CMUcam3

CMUcam4

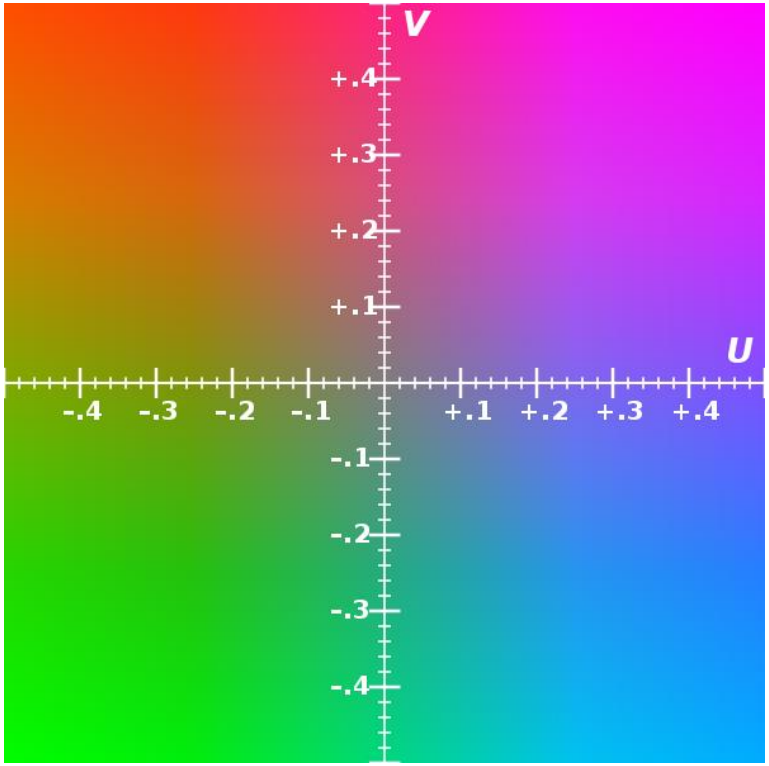
Color Tracking Explanation

RGB Color Space



Source: Wikipedia RGB Color Model

YUV Color Space

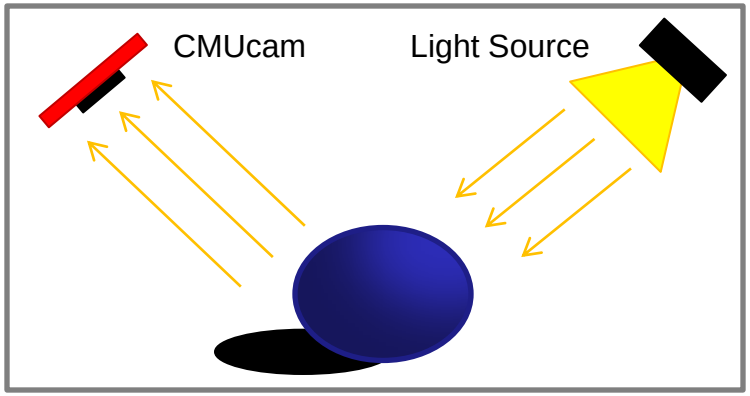


Source: Wikipedia YUV Color Model

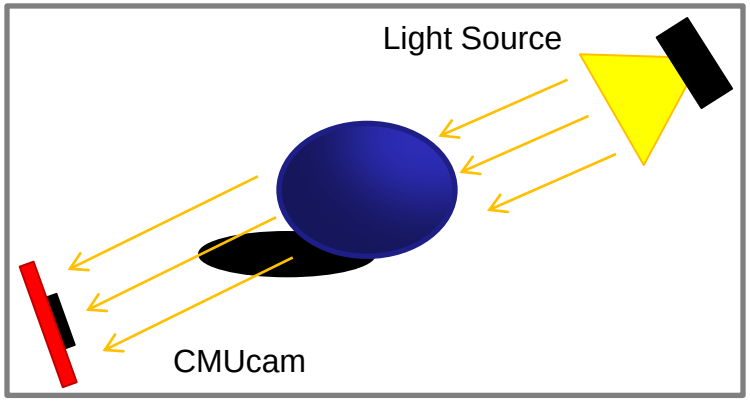
Lighting Explanation

Bad lightning examples

CMUcam sees light and dark parts of object – tracking will be bad

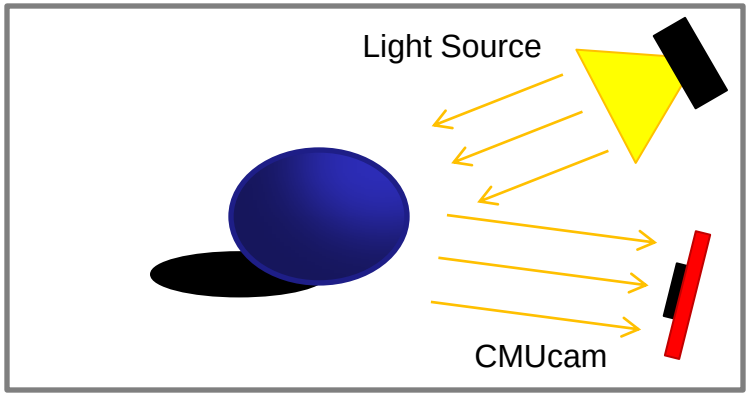


CMUcam only sees dark parts of the object tracking will be very bad

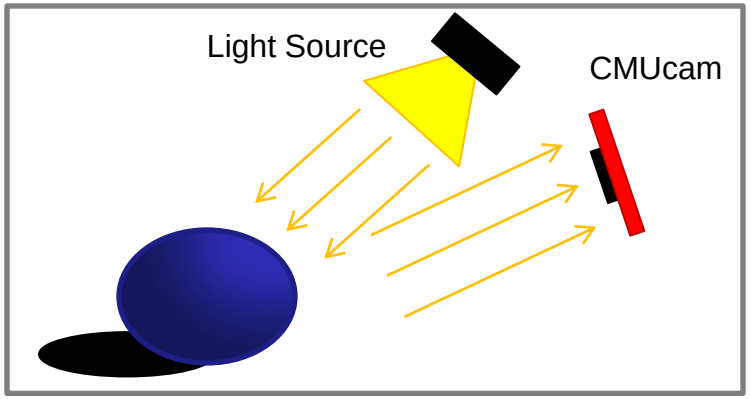


Good lightning examples

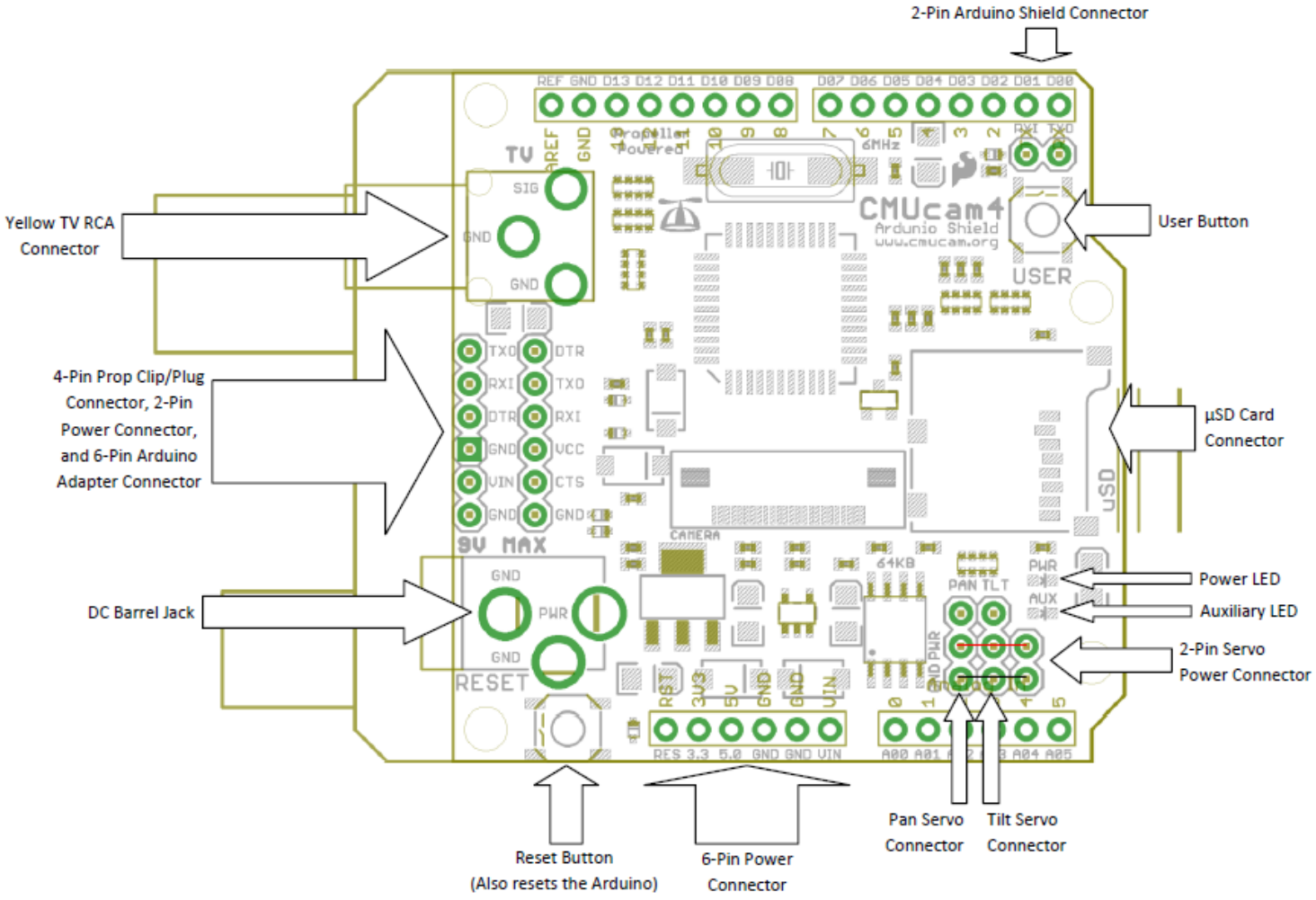
CMUcam sees light and darks parts of the object – tracking will be good



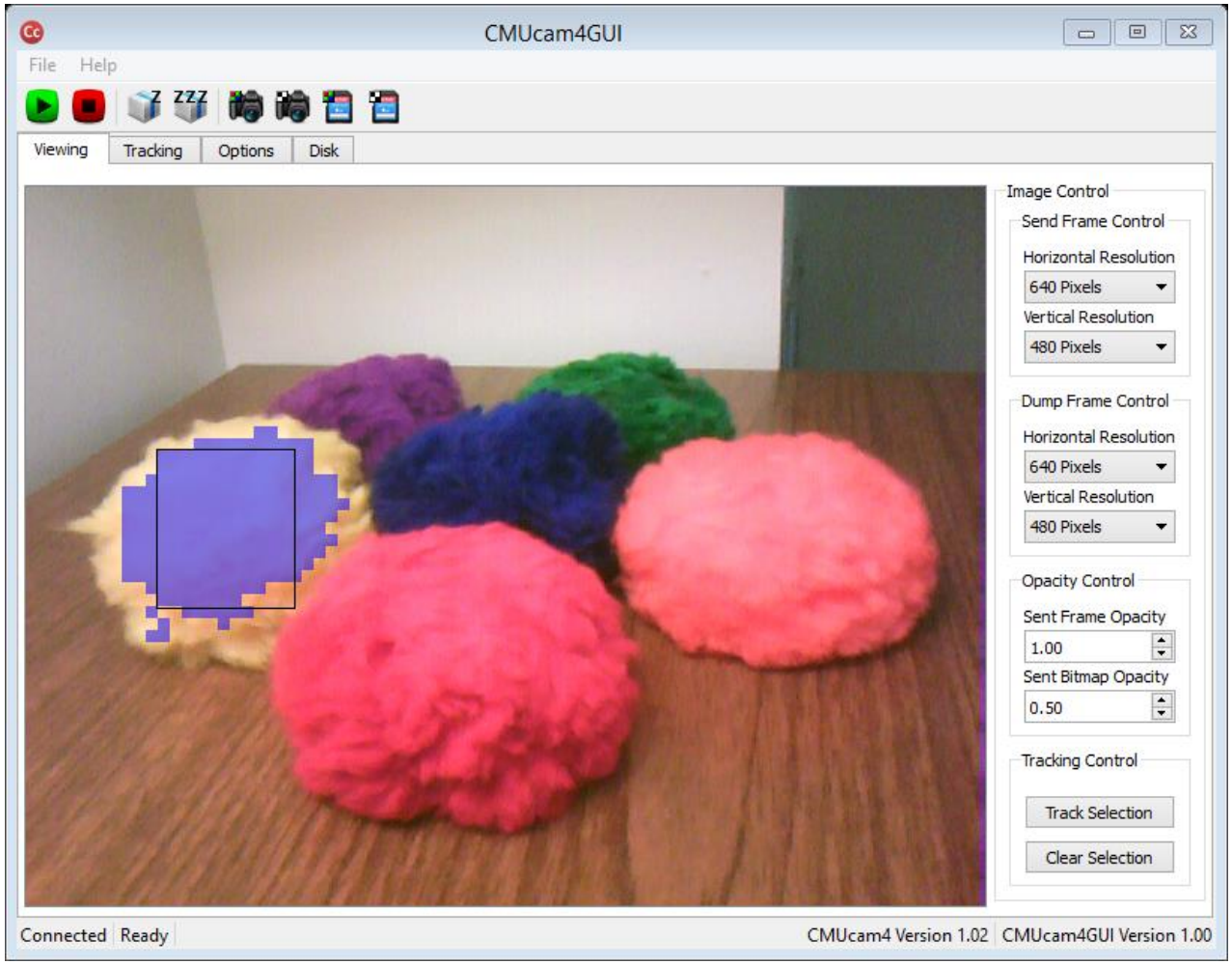
CMUcam only sees light parts of the object – tracking will be very good



Board Connections



How to track colors with the CMUcam4GUI



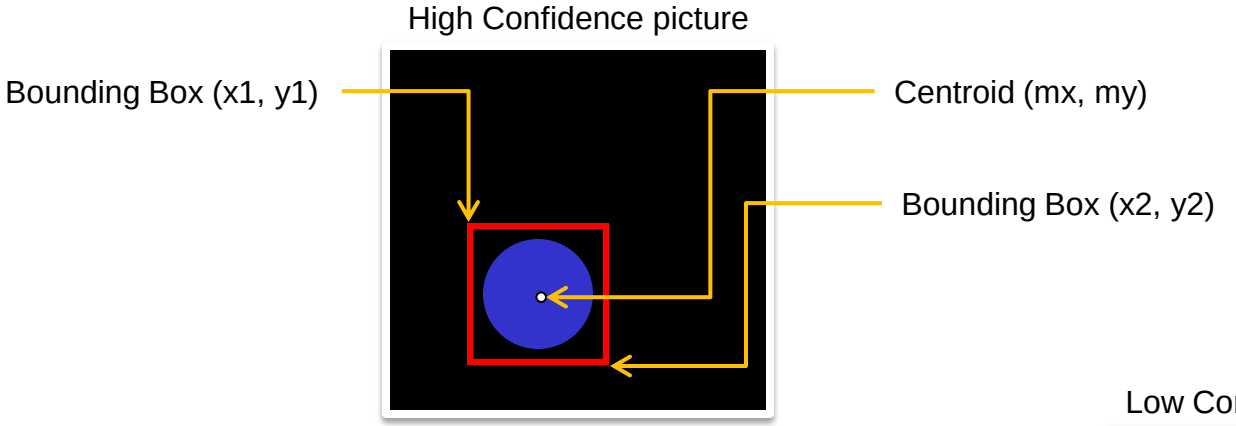
Arduino Interface Library

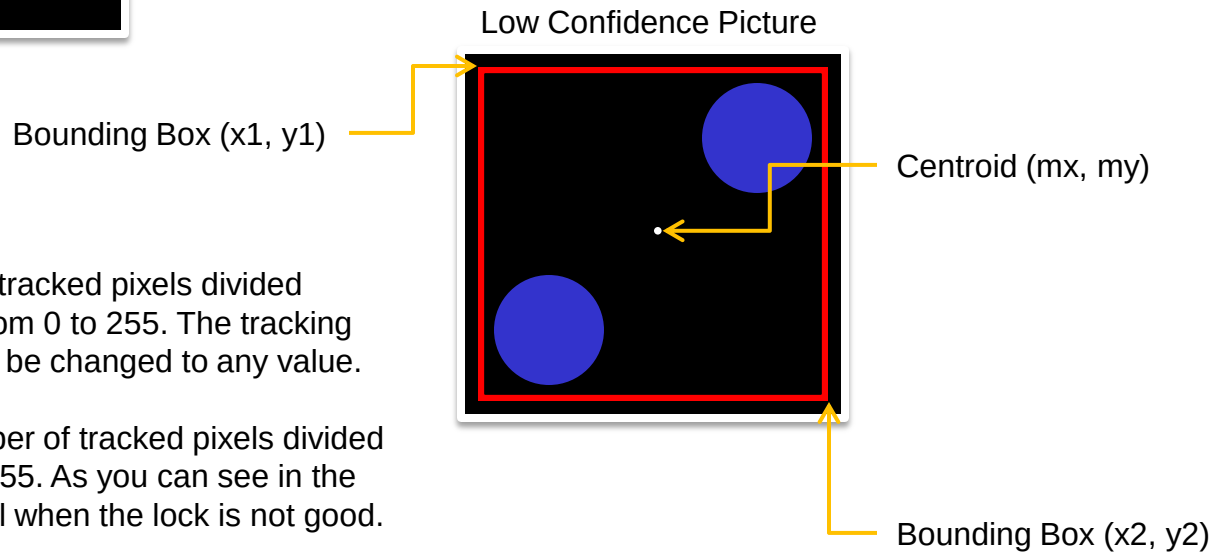
- Read the Arduino Interface Library Main Page
  - I wrote it for a good reason!

- Four color tracking information packet types:
  - Type T (Tracking Data)
  - Type F (Frame Data)
  - Type S (Statistics Data)
  - Type H (Histogram Data)

- Finally… Read The Manual about commands!

How to do object tracking





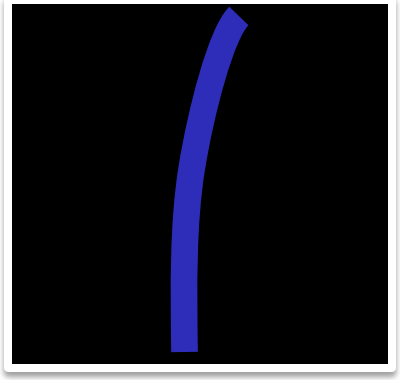
NOTE:

The “pixels” value is equal to the number of tracked pixels divided by the area of the tracking window scaled from 0 to 255. The tracking window is 160x120 pixels by default but can be changed to any value.

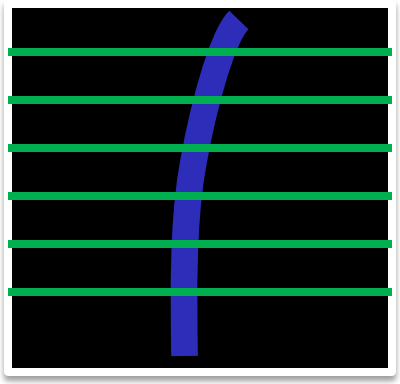
The “confidence” value is equal to the number of tracked pixels divided by the bounding box area scaled from 0 to 255. As you can see in the example above, the confidence value will fall when the lock is not good.

How to do line tracking (1)

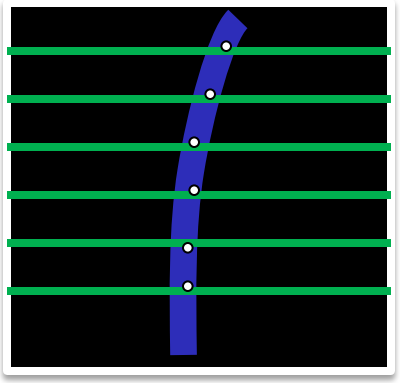
Step 0: Example Line



Step 1: Take horizontal stripes from binary image



Step 2: Compute centroids of stripes



NOTE:

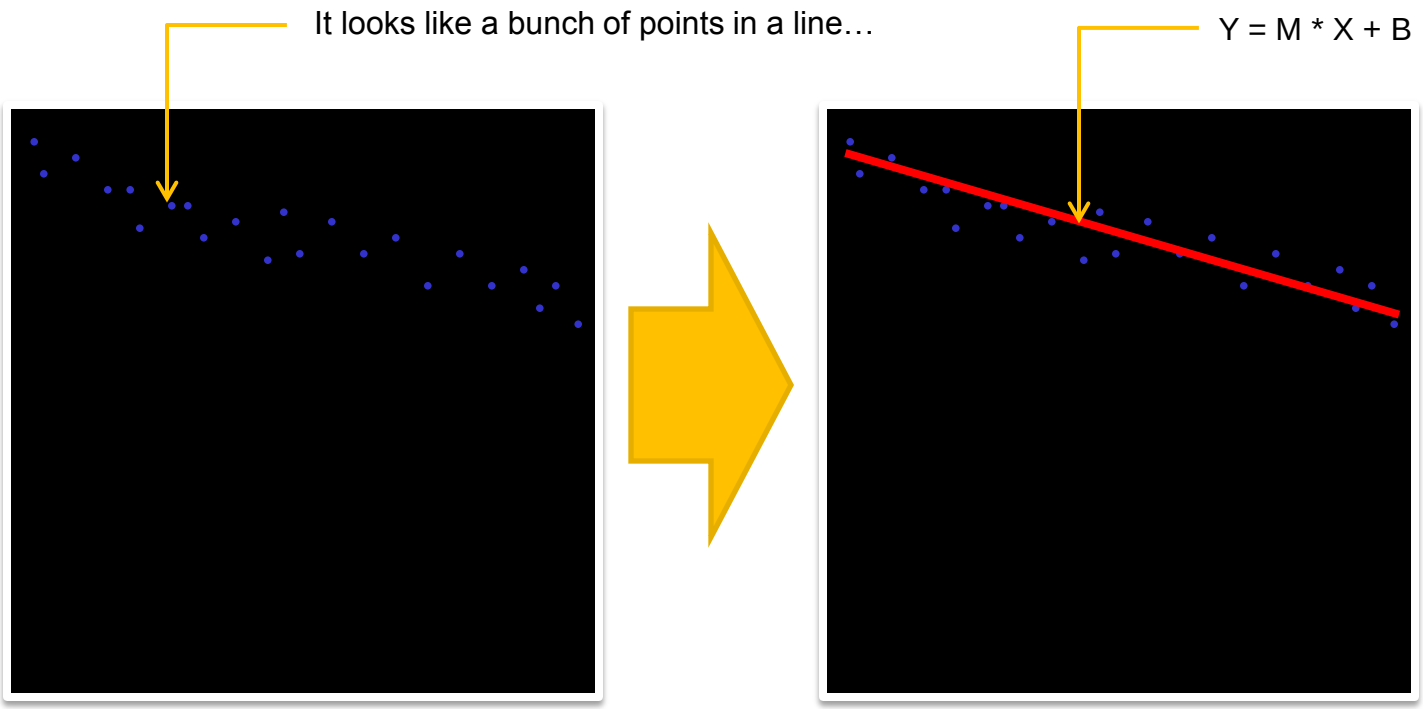
The above approach works for any number of points sampled from the line and for any rotation of the line.

Depending on your particular application you may wish to take the stripes from different directions and use a different number of stripes depending on how many points you need.

Step 3: Now you have a bunch of points representing a line. You can now use them In your control loop anyway you like…

How to do line tracking (2)

Linear Regression Calculation Code



Questions