3 Steps of Robotics

1. Sense/ Percieve the world/data (Computer Vision is very important for perception)
2. Decide what to do based on what is percieved
3. Perform an action based on the Desisicon

Self-Driving car: lane markings, pedestrians, other vehicles, and more need to be percieved

Why bother doing perception with only a camera when we can also do it with radar and LIDAR data?

That is due to the cost of cameras when compared to those technologies.

Also Radar and LIDAR see the world in 3D while a Camera can only see in 2D. But the Spatial Resolution of a Camera is much higher.

In fact that Spatial Resolution is so high that we can actually infer 3D Data from a camera.

**Overview:**

1. Upgrade our Lane Lines Project so that it can deal with much more complex scenarios. Like Curving lines, shadows, change in pavement color, how much vehicle is curving, and how far it is from center
2. Implement vehicle detection and tracking, so that we can make decisions like when to change lanes.
3. Combine both to simultaneously measure where the car is on the road, where the road is going, and the location of other vehicles in our field of view.

Finding Lane Lines is the first step in measuring some of the quantities that need to be known in order to control the car.

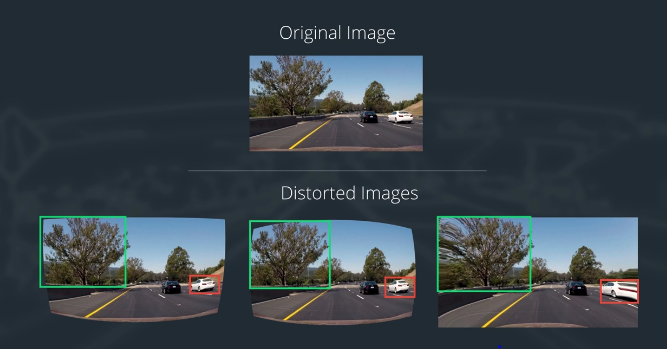
Ex. Steering a car requires info on how much our lane is curving. To do this we need to map out the lanes after transforming the perspective of the image received by the camera. Like for example transforming the perspective of the image to a top down view of the car. In order to get this perspective correct, we need to correct for the effect of Image Distortion.

Cameras don’t create perfect Images, in fact some of the objects in those images, especially near the edges can get stretched or skewed in various ways. These can be corrected with Distortion Correction.

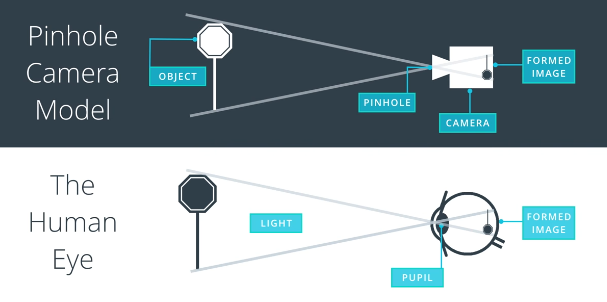
**Distortion:**

Image Distortion occurs when a camera looks at 3D objects in the real world and transforms them into a 2D image. The Distortion actually changes the shape and size of these 3D objects.

We need to correct these Distortions in order to get correct and useful information out of our cameras.

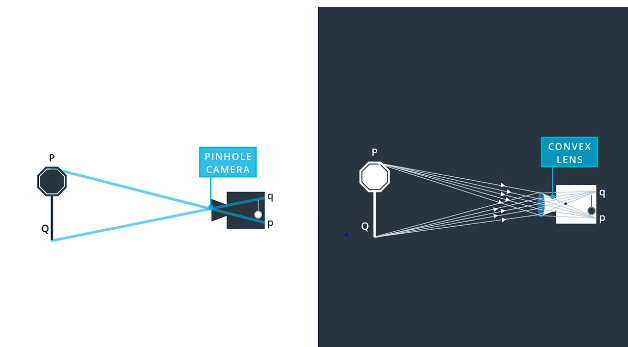


**Pinhole Camera Model:**



Camera model that emulates the human eye is called the pinhole camera model. Through the small pinhole in the camera, the camera focuses the light from a 3D object to a 2D Image at the back of the camera and is read using Film or a Sensor. This Image is also upside down and reversed due to the light rays from the top of the object being at the bottom of the sensor.

The transformation of a 3D object point P(x,y,z) to a 2D image p(x,y) is done by a transform matrix called the camera matrix. This Camera matrix is required to calibrate the camera later on.

Real cameras don’t use pinholes to focus light onto the sensor, but they use lenses. Lenses focus multiple light rays onto the sensor at the same time, but the bending of a light ray in the edges of the lens cause distortion.

The Pinhole model of a camera is free from distortions while the real life use of lenses introduces distortions.

**Types of Distortion:**

**Radial Distortion:**

Distortion at the edges of Images caused by the use of curved lenses in real cameras. These curves cause a bend in light rays that often bend a little too much or too little at the edges of the lenses. This distortion effect makes lines appear more or less curved than they actually are.

It is the most common type of distortion.

**Tangential Distortion:**

Occurs when a camera’s lens is not aligned perfectly parallel to the imaging plane, where the camera film or sensor is. It makes Images look tilted at some objects appear farther away or closer than they actually are.



**Distortion Coefficients and Correction**

These Distortions can be represented by Distortion coefficients that reflect how distorted our images is.

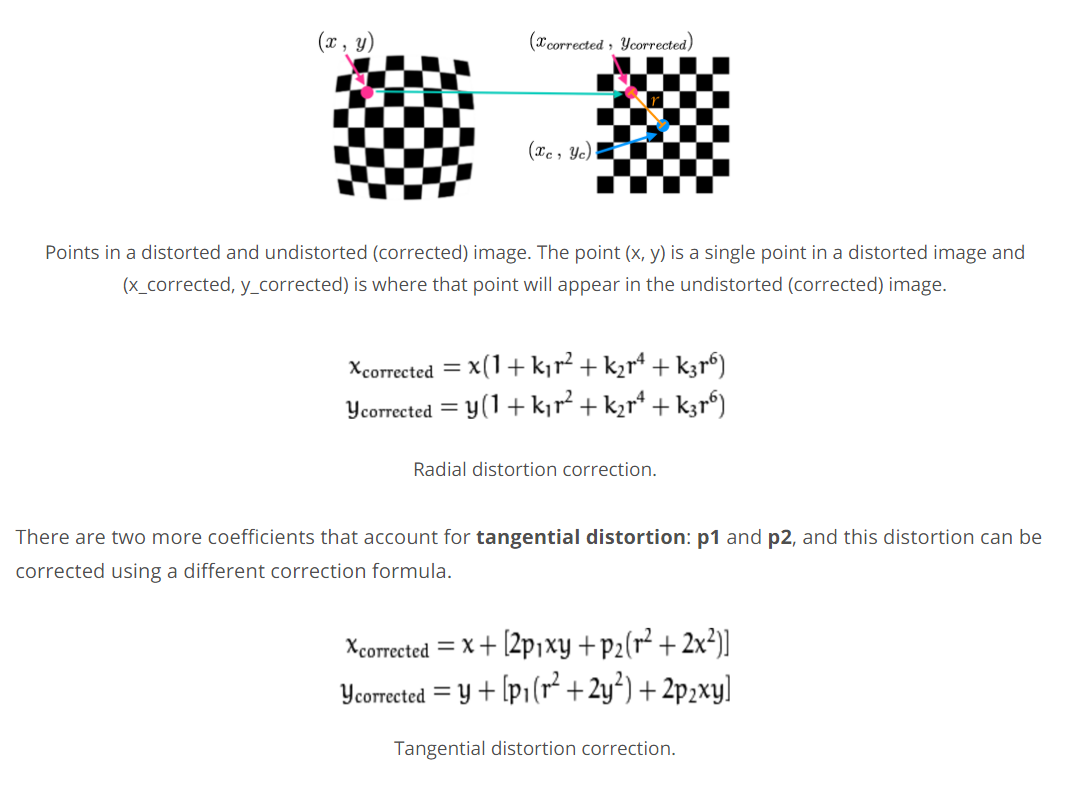
Radial Distortion can be represented by 3 coefficients, k1, k2, and k3.

Note: k3 is required for major radial distortion on things like wide angle lenses, but is generally zero on most regular camera lenses.

Tangential Distortion can be represented by 2 coefficients p1, p2.

Distortion values are typically put in an array (k1 k2 p1 p2 k3)

(x,y) is a point in a distorted image, while r is the known distance between a point in an undistorted image (Xcorrected,Ycorrected) and the center of the image distortion, often the center of the image (Xc,Yc).



To calibrate for distortions, we take an image of a known shape and then calculate the distortion coefficients, which helps us calculate or transform. The known image that is most used is the chess board. Its high contrast pattern helps detect distortions easily.

We can take multiple pictures of a chess board across a flat surface and then we can look at the apparent difference in size and shape of the distorted image to the values they actually are. Then the transform is calculated to undistorted your images.

**Look at the Finding Corners Script for an example.**

Print out the chessboard pattern and stick it to a flat surface. Take 20 to 30 pictures of the chess board from different angles and distances while ensuring they cover the enter field of view.

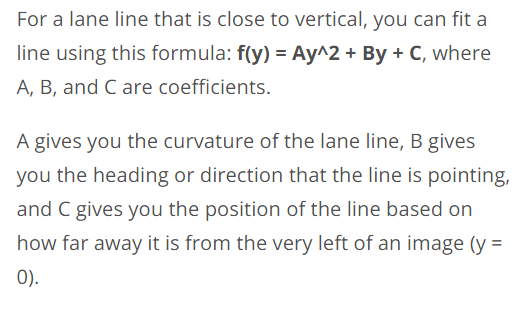
Use these images to calibrate your image

**Calculating Lane Curvature**

Now that our calibration is complete, here are the steps.

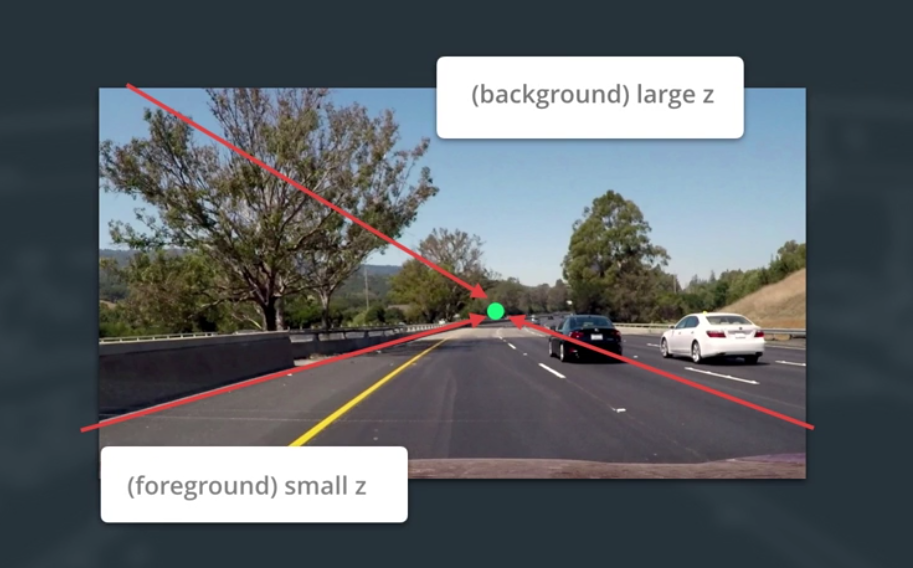
1. Detect lane lines using masking and thresholding
2. Perform a perspective transform to get a Birdseye view of the lane (Lets us fit a polynomial to the lane which is not simple without the perspective transform )
3. We extract the curvature of the lane with the polynomial we find using the points

We fit the 2nd degree polynomial to the lane line in order to gain some information about the curvature.



**Perspective Transform**

Perspective is the phenomina where an object appears smaller the farther it is from a viewpoint and parallel lines seem to converge to a point.



We can characterize perspective by saying in real world co-ordinates (x,y,z) the greater the z value from the camera, the smaller it will appear in the 2D image.

This info is used by the perspective transform. It transforms the apparent z co-ordinate of the object points which changes that objects 2D image. It effectively drags and pushes points from the camera to change the apparent perspective.

It maps the points in a given image to different, desired, image points with a new perspective.

The birds-eye view is the perspective we want to transform to calculate the lane curvature. This is because the curve in the lane lines is much more apparent with a top down view. That view shows that both lanes curve in the same manner.

The top down view also helps us match the road information with a map.

The perspective transform involves a similar process as calibration. Instead of mapping object points to image points, we map the points in a given image to different desired image points with a new perspective.

In openCV we have a getPerspectiveTransform method that gives us the transform with the input of the source and destination points. 4 Points that defined a rectangle on a plane are enough to transform from one perspective to another (Look at Perspective Code)

Now we want to detect the edges of the lane lines in our Image.

In the intro lesson, we used Canny Edge Detection to find our edges. However it gave us a lot of edges that we discarded.

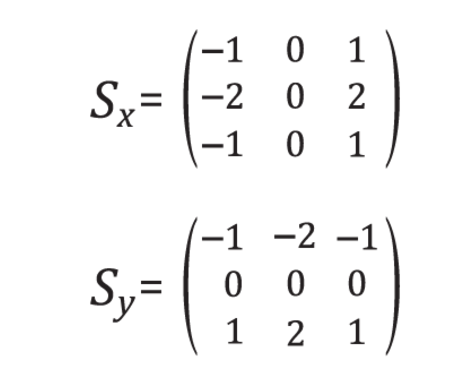
To make a better edge detector for lane lines, we can take advantage of the fact that lane lines are close vertical. We can use the calculations of the gradient of the image that is a part of canny detection in a way to detect steeper edges that are more likely to be a lane line in the first place.

Taking the derivative of an image involves using the Sobel Operator.

**Sobel Operator**

The Sobel Operator takes the derivative in the x-direction, or the y-direction.

Here are the operators for x and y Sobel. These are the 3x3 operator (kernel is the size of 3, the minimum size). The kernel can be any odd number, and the larger the kernel, the gradient is taken over a larger region, meaning a smoother gradient.



If you had an image with color values rising from left to right, and applied x-direction Sobel, you would get a positive output.

When using the Sobel operator on an image of the road, the gradient in the x-direction emphasizes edges closer to vertical, while the gradient in the y-direction emphasizes edges closer to horizontal.