NUBOTS

Computer Vision is a 3D Problem

Dr Trent Houliston
NUbots

What do we need to find in RoboCup

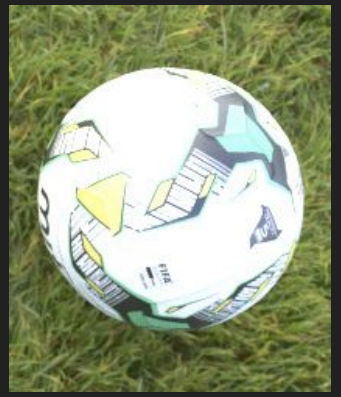
Soccer balls

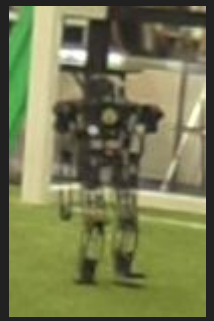
Goal Posts

Field Lines

Other Robots

Obstacles



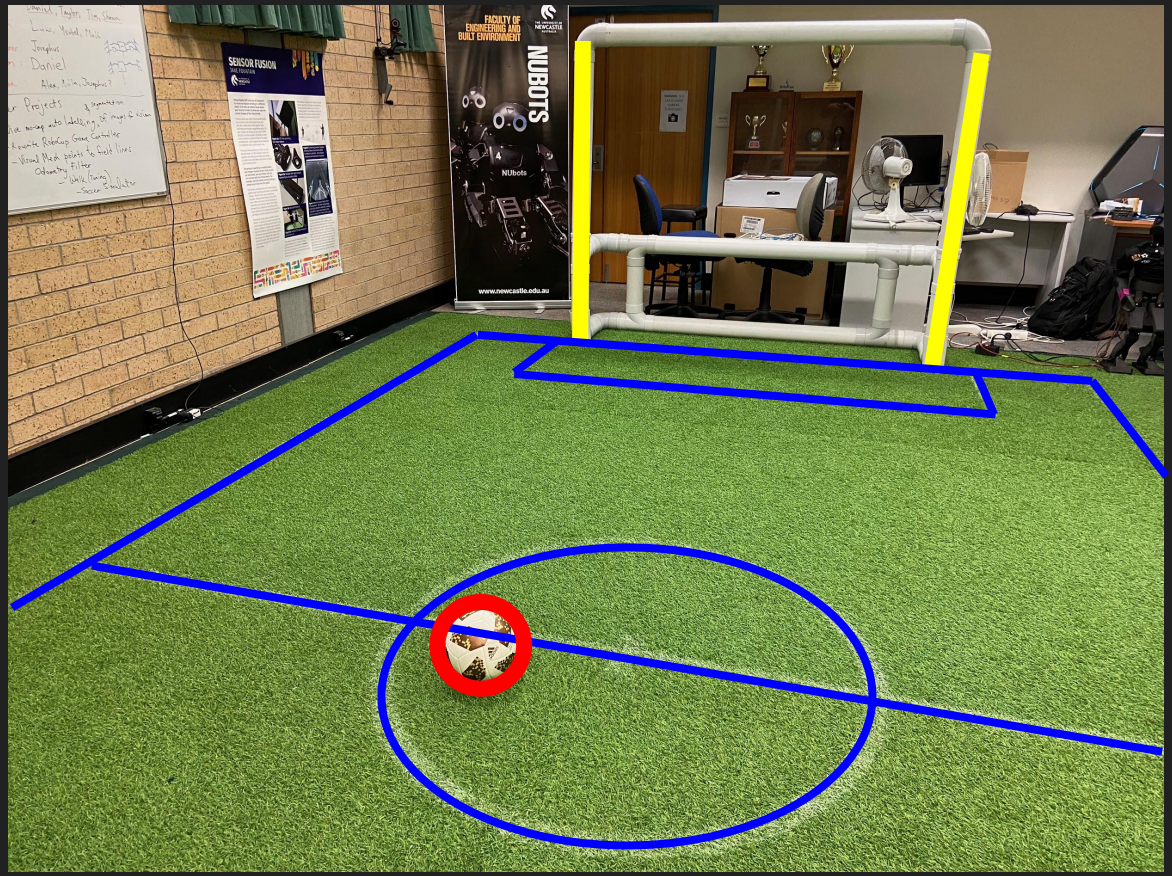


How do we find objects

How do we find objects

How do we find objects

How do we find objects

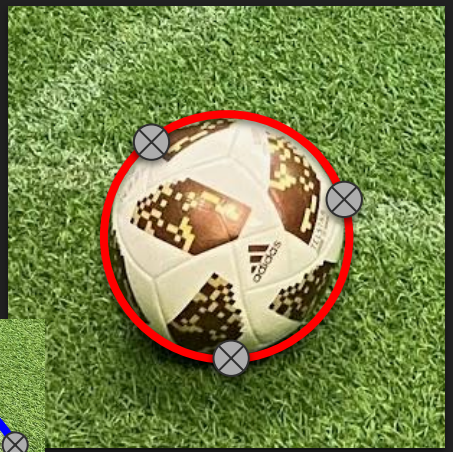


This seems fine

2 points on a line can be used to define goalposts, field lines and other straight objects

3 points on a circle we can use for a ball

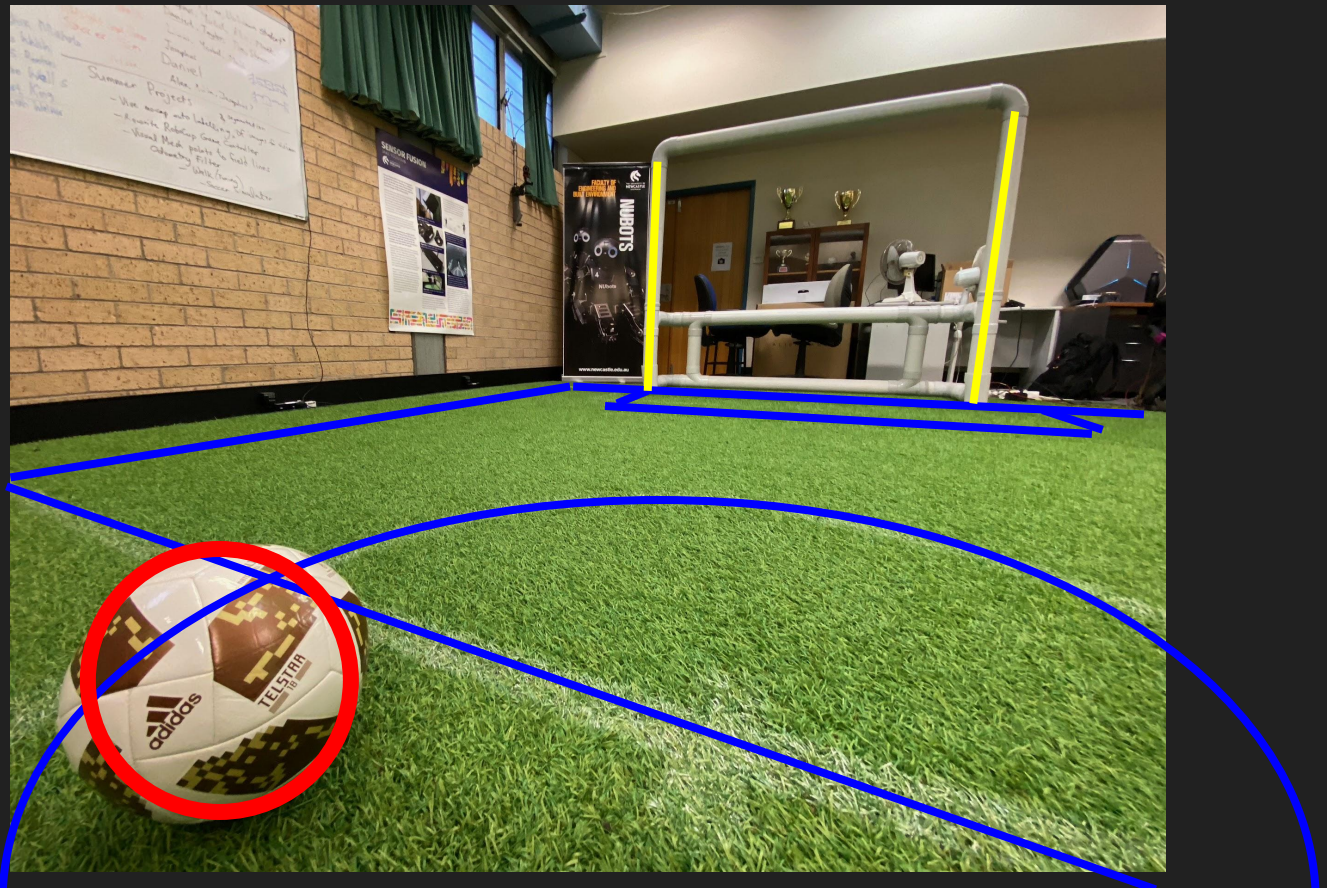
5 points can be used to define an ellipse (conic section) for the centre circle



Another example

Another example

Another example



This still seems fine

We can just fit an ellipse to find the ball

Or we can slacken the circularity requirements on the circle detector

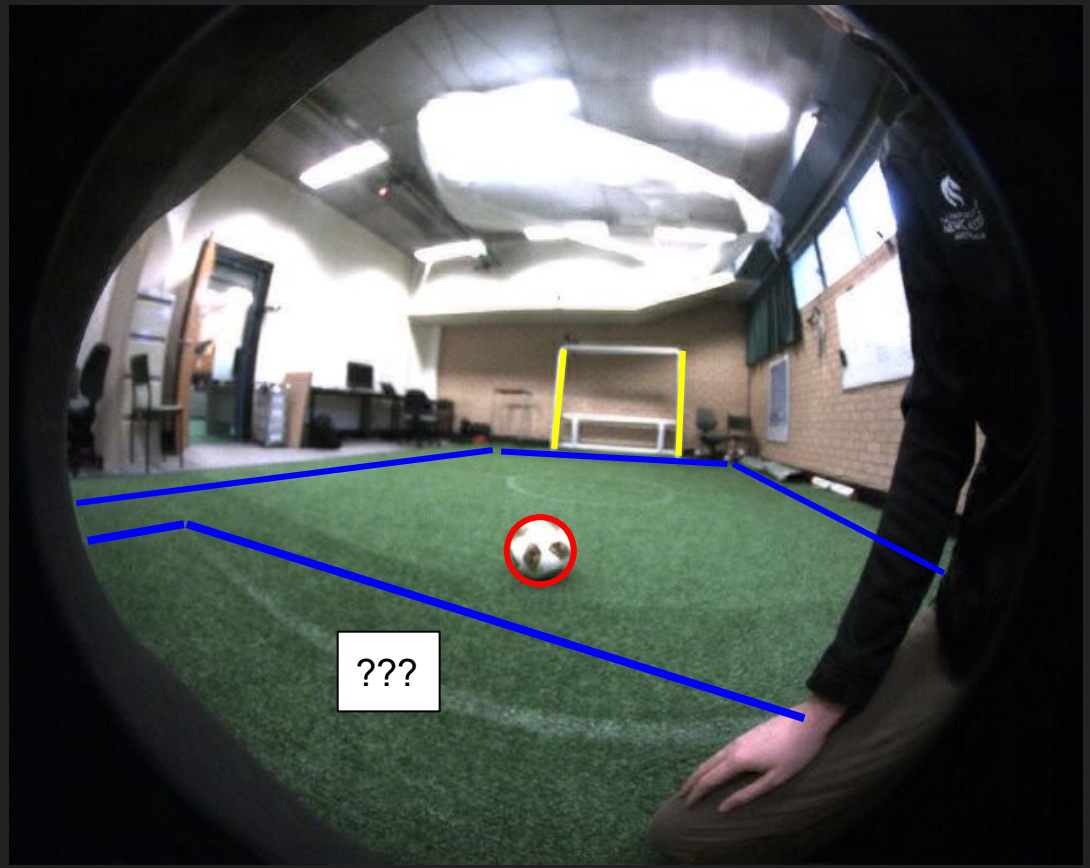
Goal posts and field lines still seem straight enough to use lines

The centre circle looks a little worse though

Fisheye Lens

Fisheye Lens

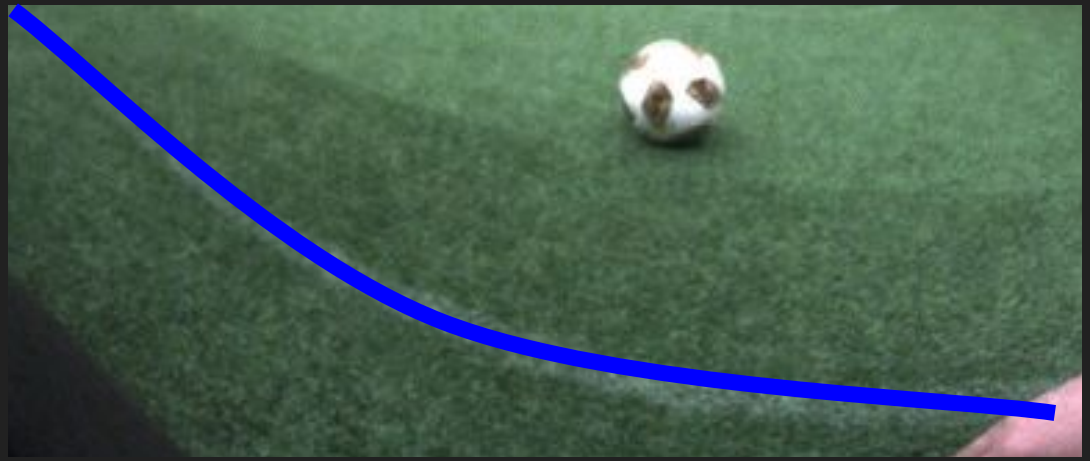
Fisheye Lens



Is this fine?

The circles still seem to work here

Maybe we can fit curves to the goal posts and field lines?

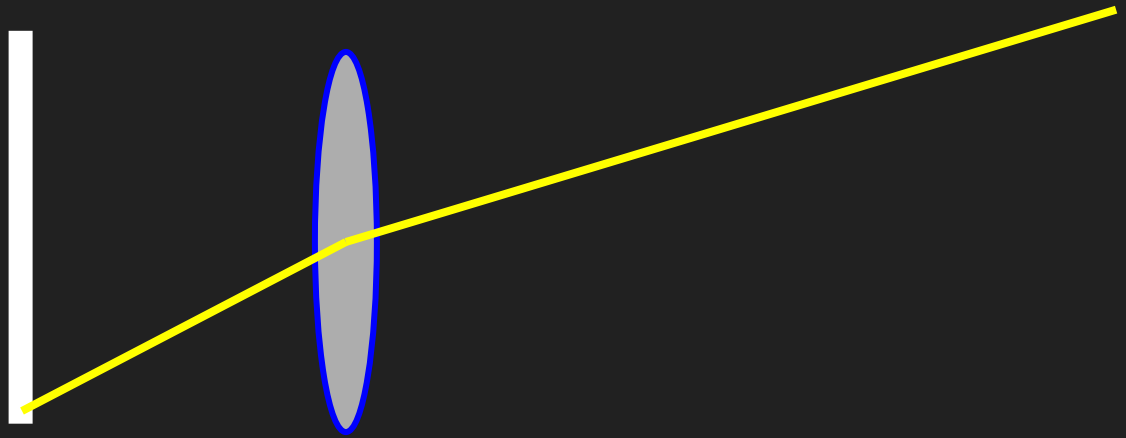


Oh dear

What is the problem here?!

The problem is we are taking a 3D problem, and trying to solve it in a 2D projection

To understand what is happening we need to look at the optics and what is happening to the light that is hitting our camera sensor



Lens Projections

Lens Projections

A lens projection is how your camera takes the 3D world and squashes it onto a 2D surface

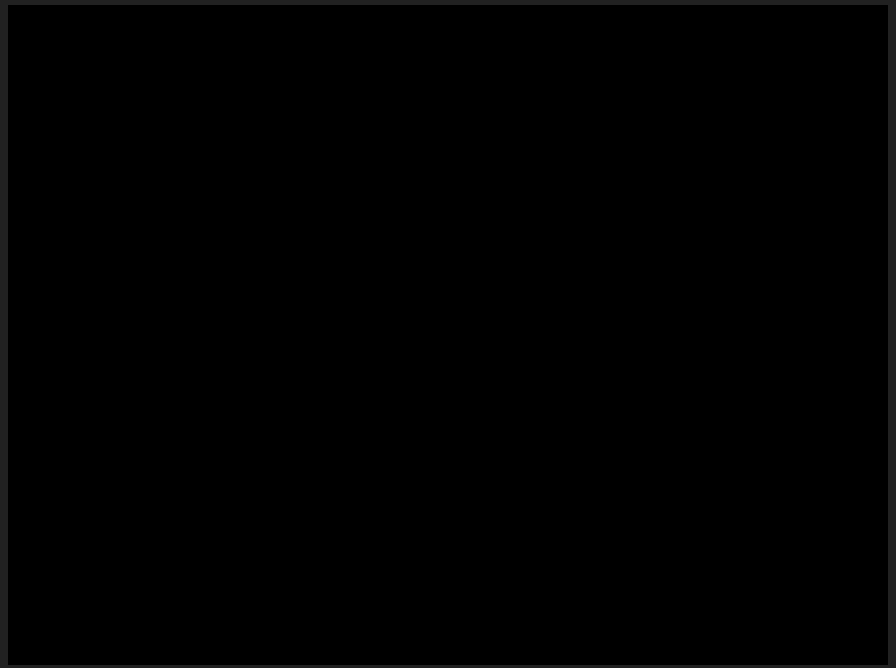
Just like how you can’t take the earth and make a perfect 2D map you can’t take a 3D world and make a perfect 2D image

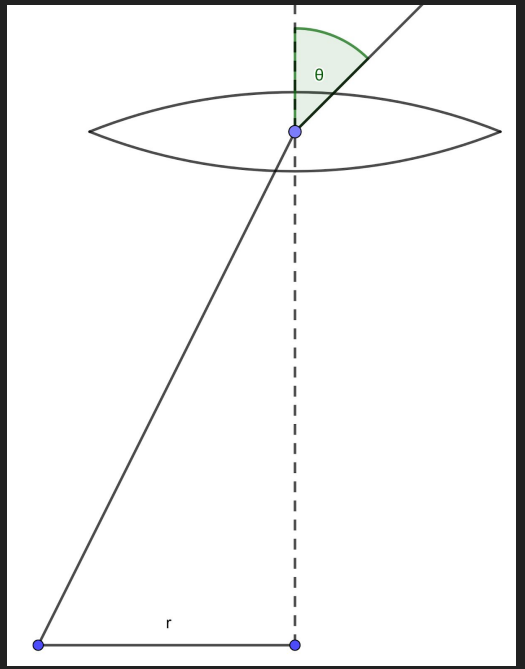
The most important thing to know is that a lens projection is a function that converts an angle to a distance

$$r = f(\theta)$$

Lens Projections

A lens projection takes the angle of incoming light and projects it to a distance on the camera sensor





Projection Zoo

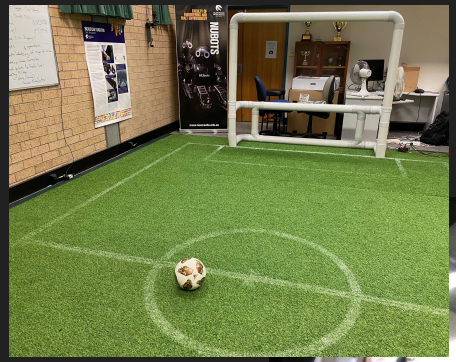
There are many different projections out there each good for different tasks but the two most common are equidistant (fisheye) and rectilinear (“ordinary lens”)

Rectilinear $r = f \tan(\theta)$

- Projects objects onto a plane
- Great for humans as we make our monitors and print photos onto planes

Equidistant $r = f\theta$

- Great for wide angle lenses, distances in pixels linearly maps to angle



But wait, isn’t this just distortion?

One very common problem when discussing projections is the confusion between a projection and a distortion

Non “ordinary” (rectilinear) lenses such as equidistant (fisheye) are often considered “distorted” and must be “undistorted”

Different lenses produce different projections, different mathematical models that convert angles to distances

Then what is distortion?

Distortion is any **deviation** from your lens model

So if you took a rectilinear projection, but the lens was not perfect you might be projecting into a not quite flat plane that would be distortion

Most commonly it is factored as a Brown-Conrady model which is basically a polynomial equation

Like how a Taylor series can model any function if you have infinite terms, lens distortion can model **almost** any distortion if you have enough parameters

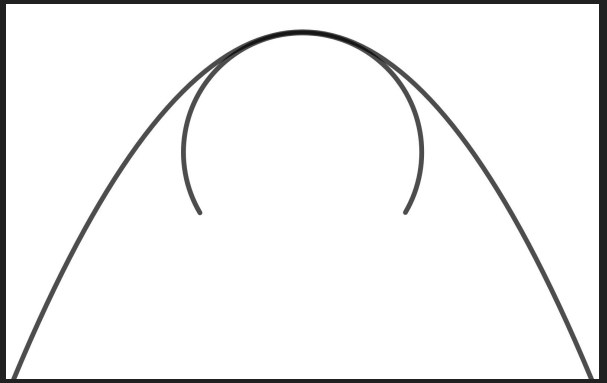
Fisheye is not distortion

However you could never model a fisheye lens with more than 180 degrees using distortion regardless of the number of terms

You would be trying to fit a polynomial to a circle

No matter the number of terms you have
you can never make it bend back on itself

This is why most “undistortion” functions you find online for fisheye say they won’t work for lenses with over 150º degree field of view



Use the correct model and then add distortion

When you are modelling a lens, first identify what kind of lens it is

Then you can use fewer distortion terms to define the lens

I have yet to find a lens that required more than 2 polynomial terms to achieve subpixel accuracy for the model distortion

3D Vision

So what do we do

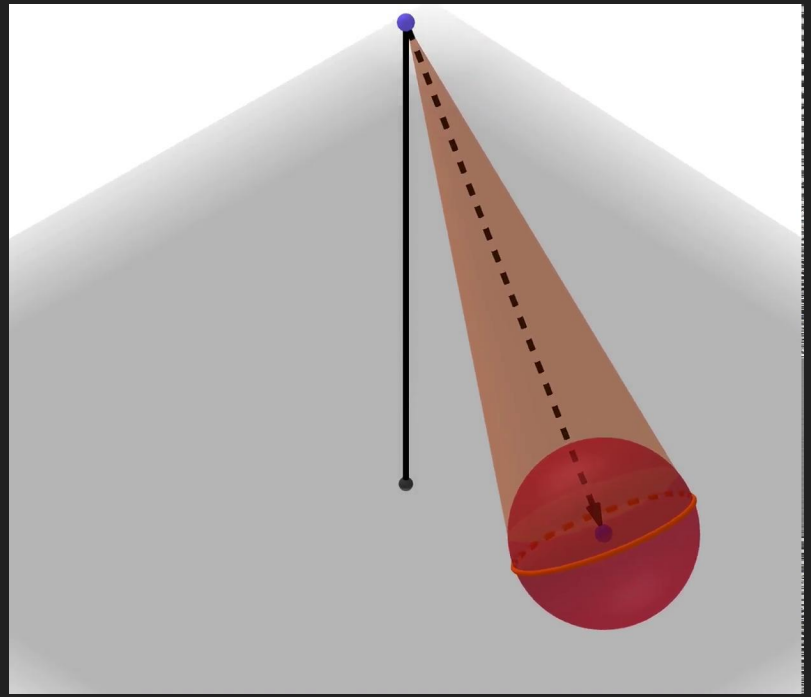
Stop trying to solve a 3D problem in a 2D projected space

Most objects that you are looking at in 2D become very simple in 3D

To do that, you convert your pixel coordinates into 3D unit vectors using your lens projection

With these unit vectors we can start to define the geometry we are after

Balls



In 3D, no matter what surface we are projecting the ball onto, it first makes a cone

There is an axis that goes through the centre of the ball, and the dot product of that central unit vector to any point on the edge will always be equal to a constant

No matter what lens projection we use,
a ball will always be a cone in 3D that we can define with 3 unit vectors

This also explains why an ellipse fits so well in a rectilinear lens. The intersection of a cone with a plane is a conic section.

Goals, Field Lines

Anything that is a straight line in the world can be defined using a plane (or plane segment)

Take any two unit vectors on a line and take the cross product

If you take the dot product of this Vector with any other and it is zero, then that vector is also on this same line.

Using this we can now identify straight lines regardless of the lens

Centre Circle

The centre circle is a little more complicated but if we think about it in 3D we can find a model for it too

This one is easiest to do with the combination of the robots IMU

First we need to project our vectors to the ground (vector * (height / vector.z))

Once we do this, if we ignore the z component of the vector our centre circle becomes a circle!

We can then use three points on this circle to define its model

Advantages of 3D

When thinking about these problems in 3D we get many advantages

Models are simpler,

- An ellipse model for a ball is far more complicated than a cone
- Trying to fit a curve to a fisheye line would be tricky

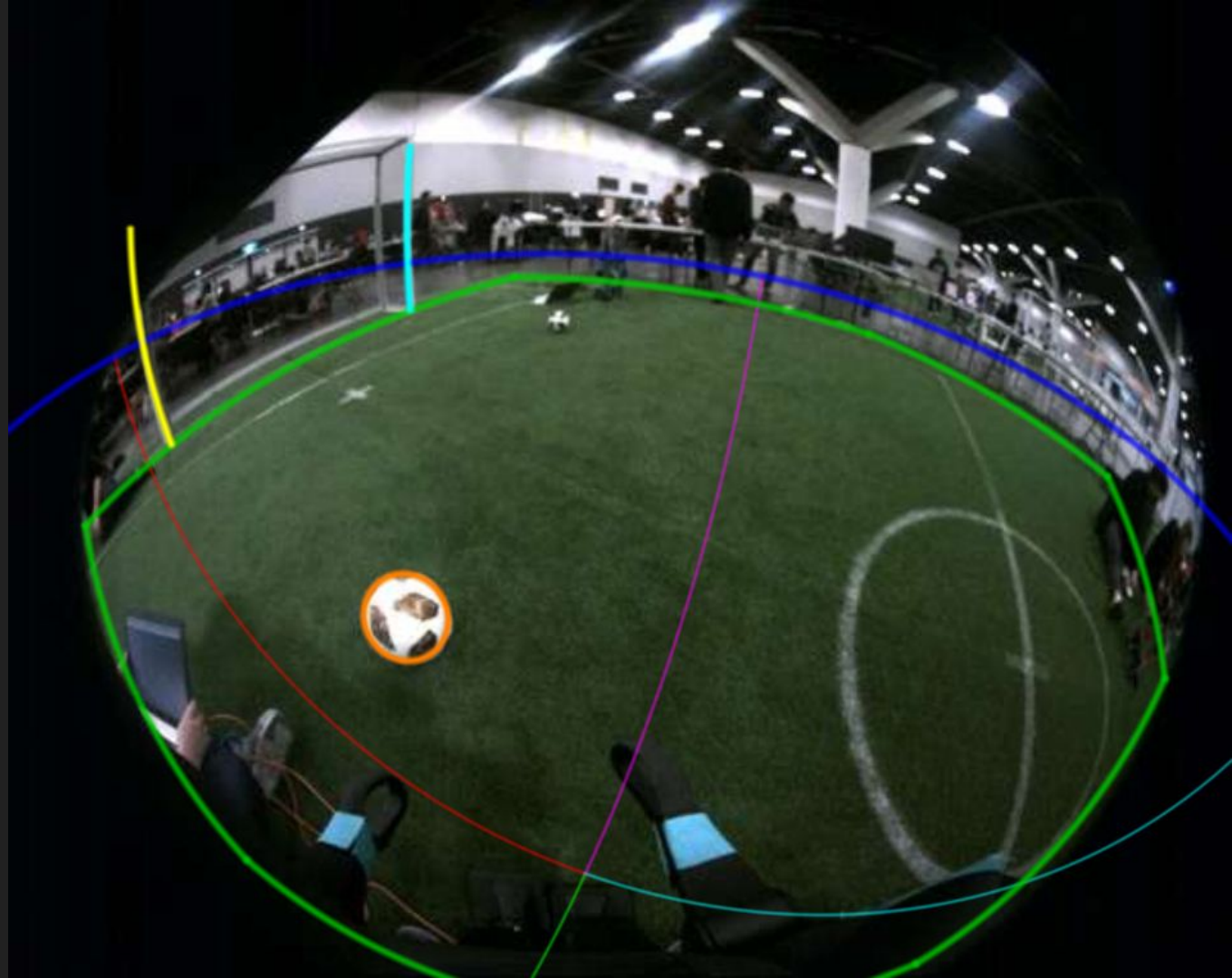
Models are more accurate

- We don’t slacken our detectors for the projection
- We can include our IMU in 3D (goal posts are always vertical!)

NUbots Vision

Objects here are drawn in 3D and projected back

Planes become curved lines once they are projected through our lens



Machine Learning

Machine Learning

A lot of the time now, we don’t use traditional vision methods for vision

We use machine learning to solve some of these problems for us

Do these same problems exist in machine learning?

How do we handle these projection problems there?

Training Variations

Often when training networks we will present them with many examples to help

If those examples include the object in different projections the network can learn these different forms

If we don’t have them in our training data, we can augment our data to create them

Training Variations

If we wanted the network to work reliably across different lenses we would need even more training data

We also need larger networks so they can learn these distinctions

A strong downside is the network cannot tell the difference between an object that is warped in reality and one that is warped by the projection

For example, imagine a goal post in a fisheye lens, depending on where it is it looks different. What if you were looking at a bent goal post? How would you tell the difference?

The Visual Mesh

The method the NUbots uses to solve this problem is the Visual Mesh

This technique takes the 3D approach all the way through the neural network

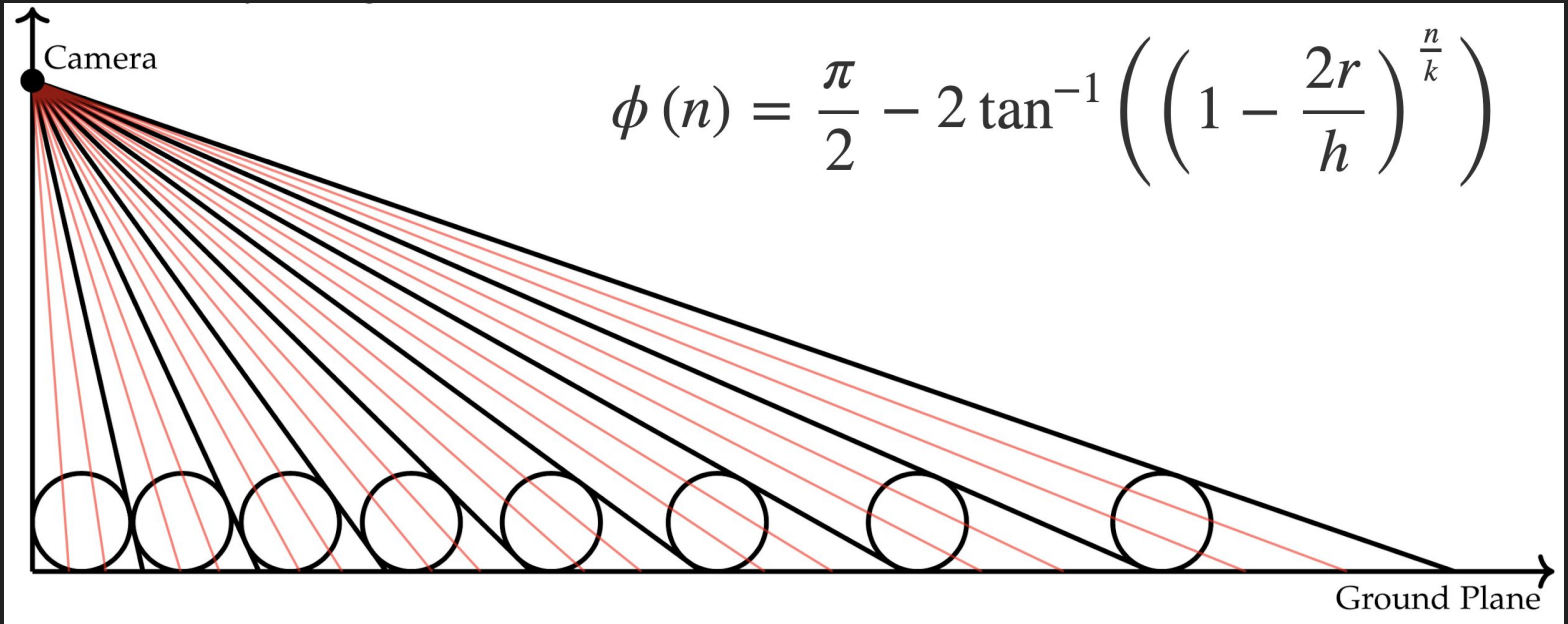
Perform your logic in 3D, and project back to 2D only when you need a pixel colour of a specific direction

This means that we no longer suffer the problems associated with projection and gain some other advantages too!

Visual Stacking

Stack objects as they would appear along a ground plane so they touch visually

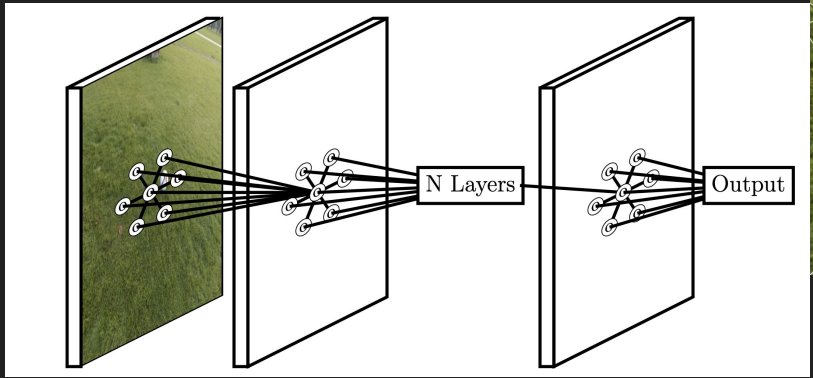
Now we're always certain that we have the same number of points on an object

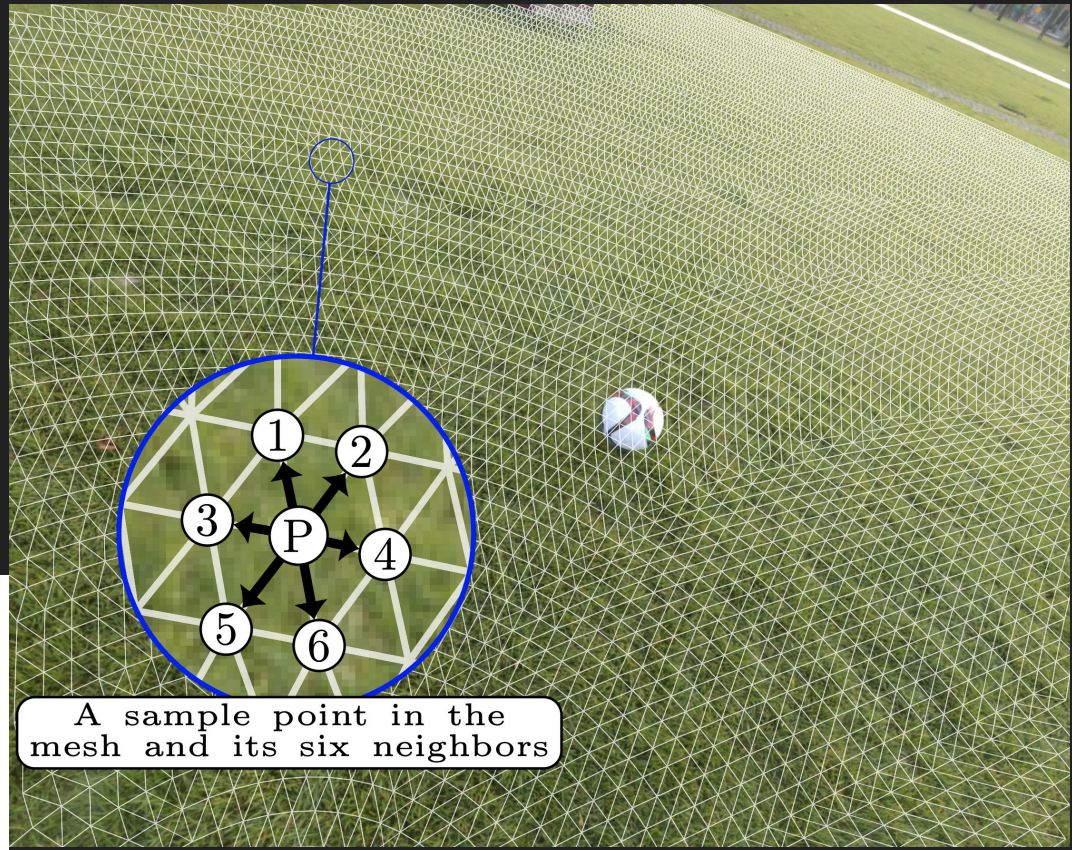


The Visual Mesh

This then makes a graph

Which we combine into convolutions

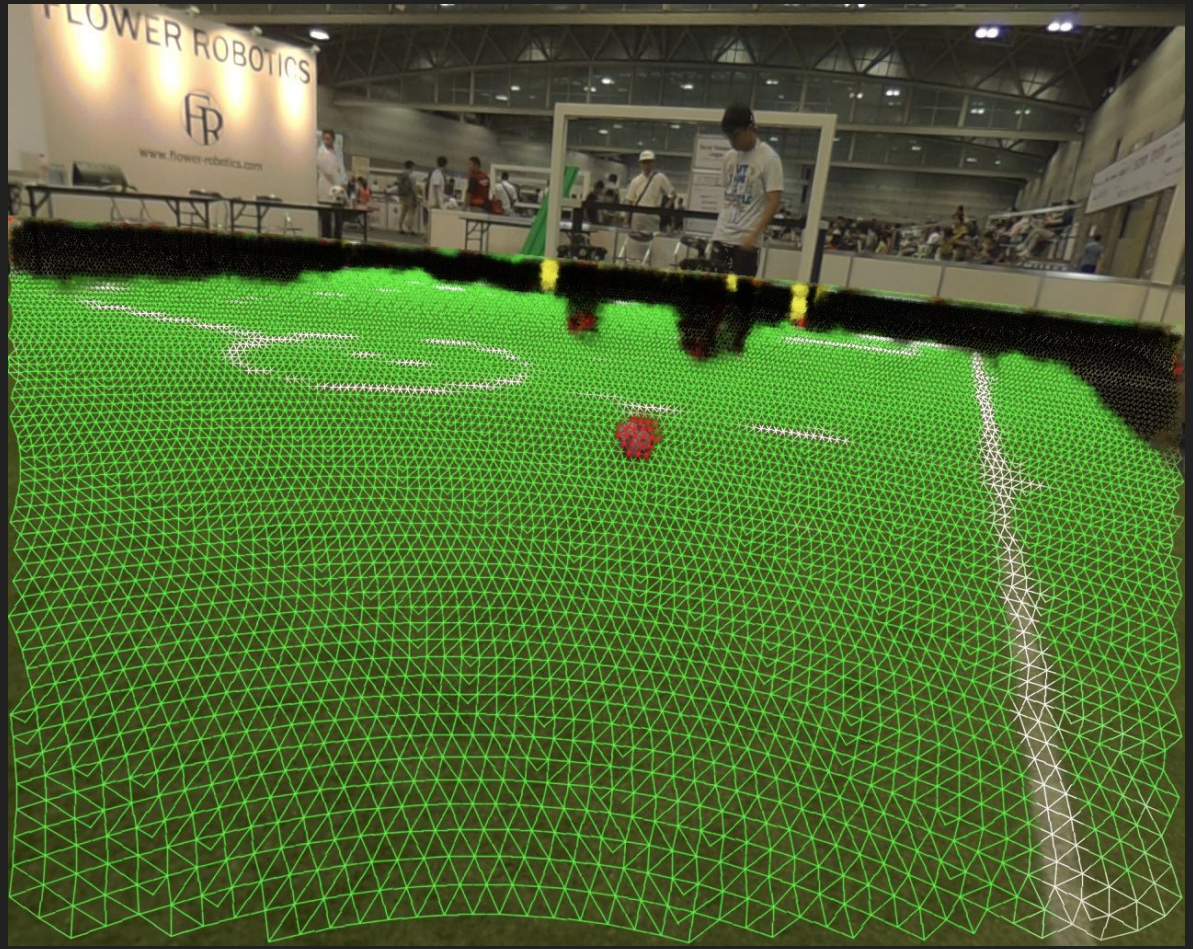




The Visual Mesh

This then makes a graph

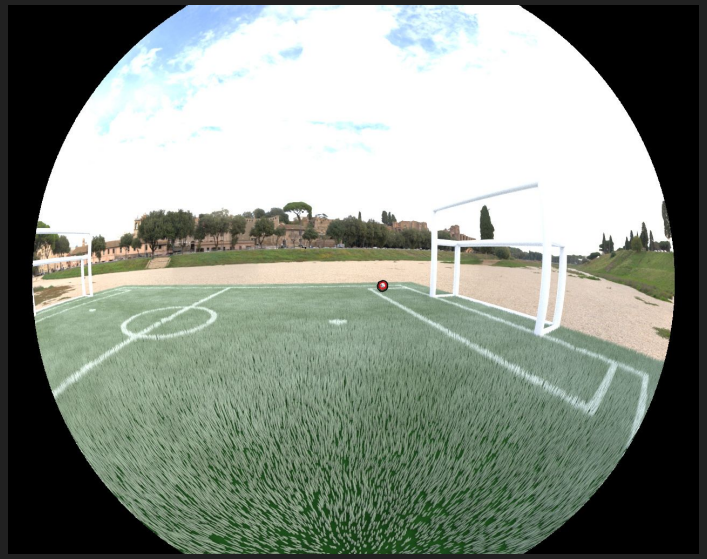
Which we combine into convolutions



The Visual Mesh

The NUbots run the visual mesh on the integrated Intel HD Graphics 650

Using this we are able to run a neural network at hundreds of frames per second on both of our fisheye lenses at 1280x1024 resolution!



Conclusion

Use the correct lens model for your camera

Use 3D as much and as early in your vision pipeline as possible

Think about what shape an object is in 3D not 2D

Try to let your machine learning operate in 3D as well

Visual Mesh: https://github.com/Fastcode/VisualMesh

NUbots Code: https://github.com/NUbots/NUbots

Thanks!