COMP3421

VR and AR 2, Exam, Revision

Robert Clifton-Everest

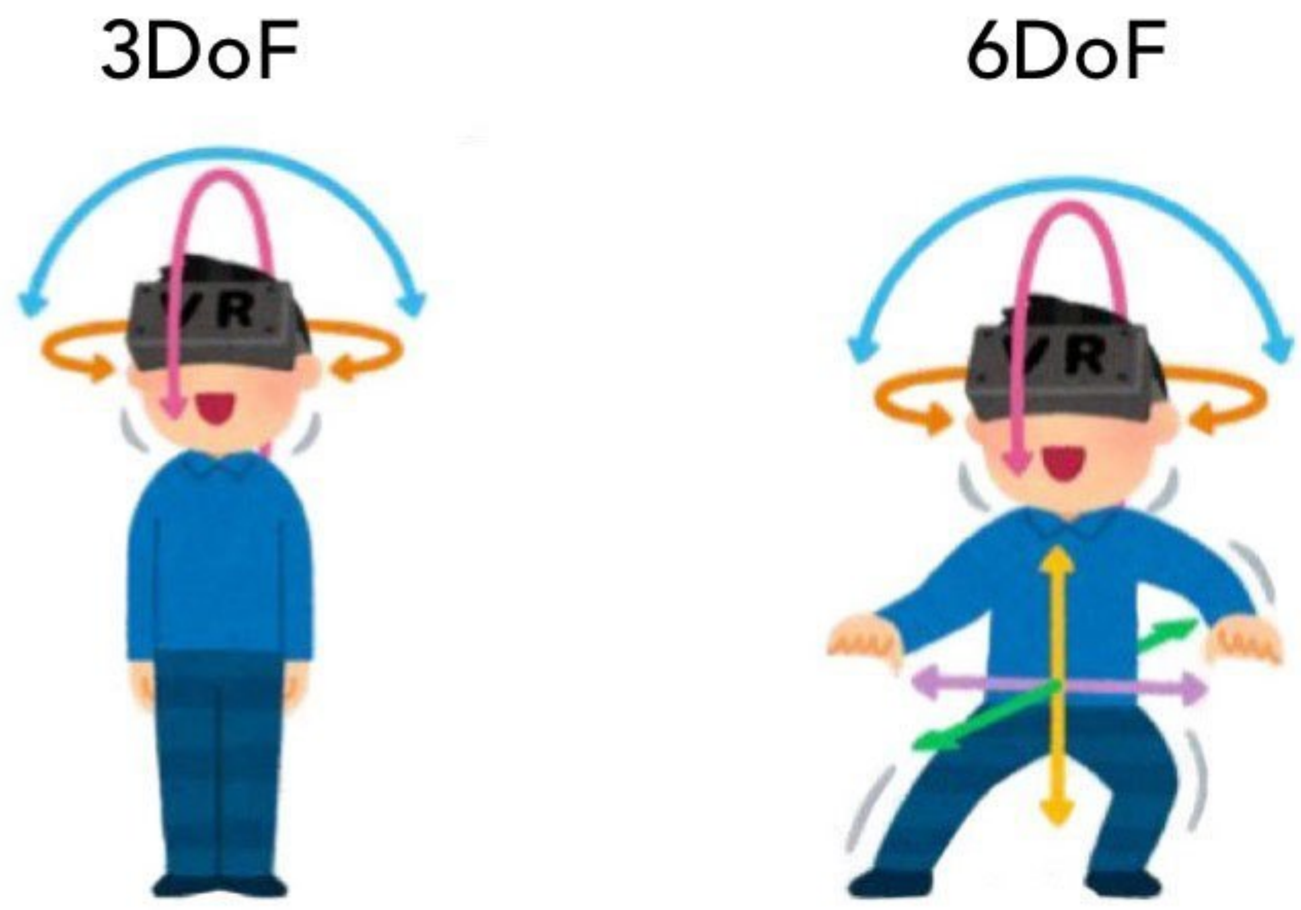
Email: robertce@cse.unsw.edu.au

Tracking

- For VR to be immersive, the camera in the virtual world needs to track perfectly with the user’s head.

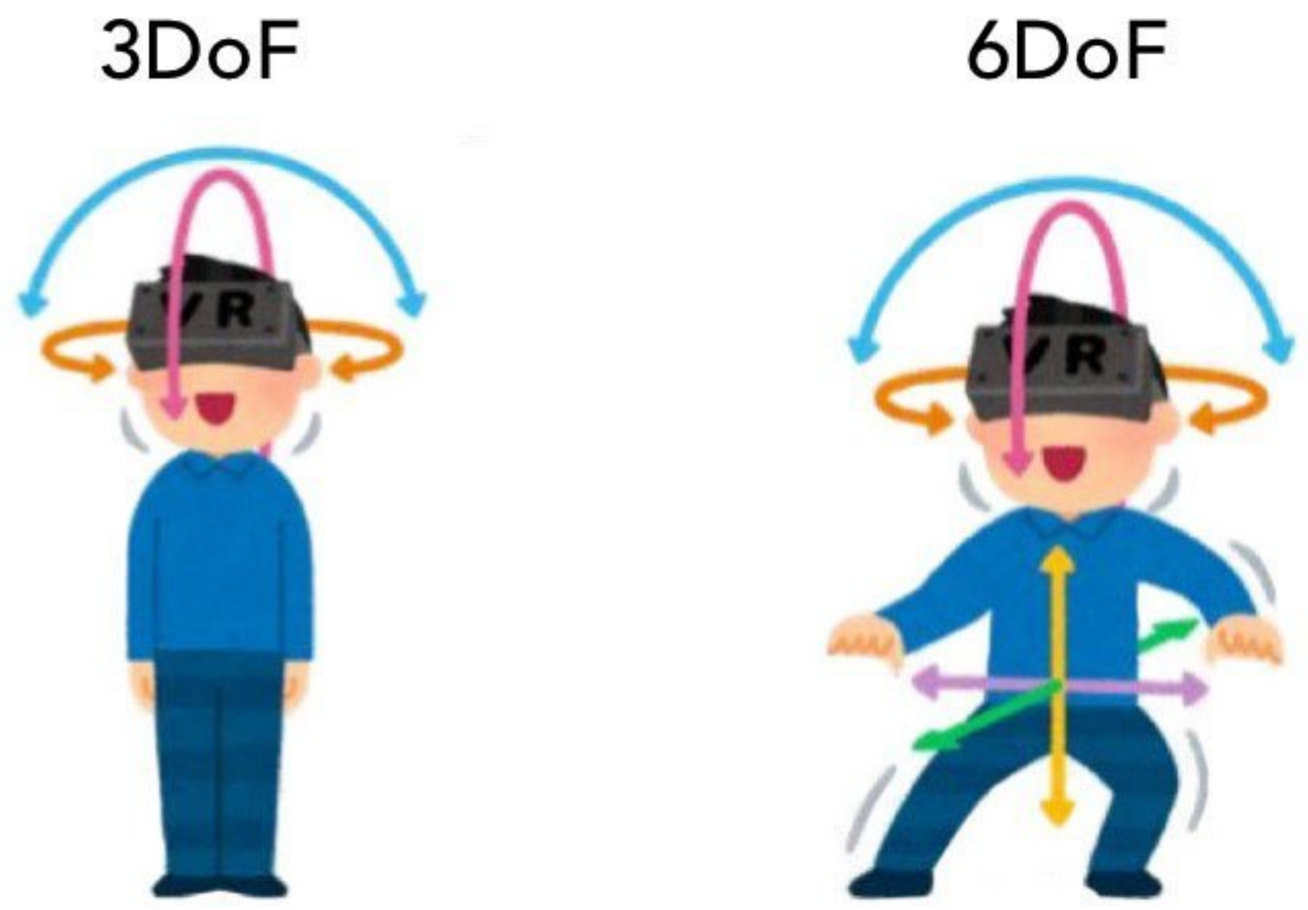
Tracking

- Rotational tracking just tracks the rotation of the head for all axes (3 degrees of freedom).



Tracking

- Combing rotational with positional tracking yields a complete definition of a camera (6 degrees of freedom)



Rotation Tracking

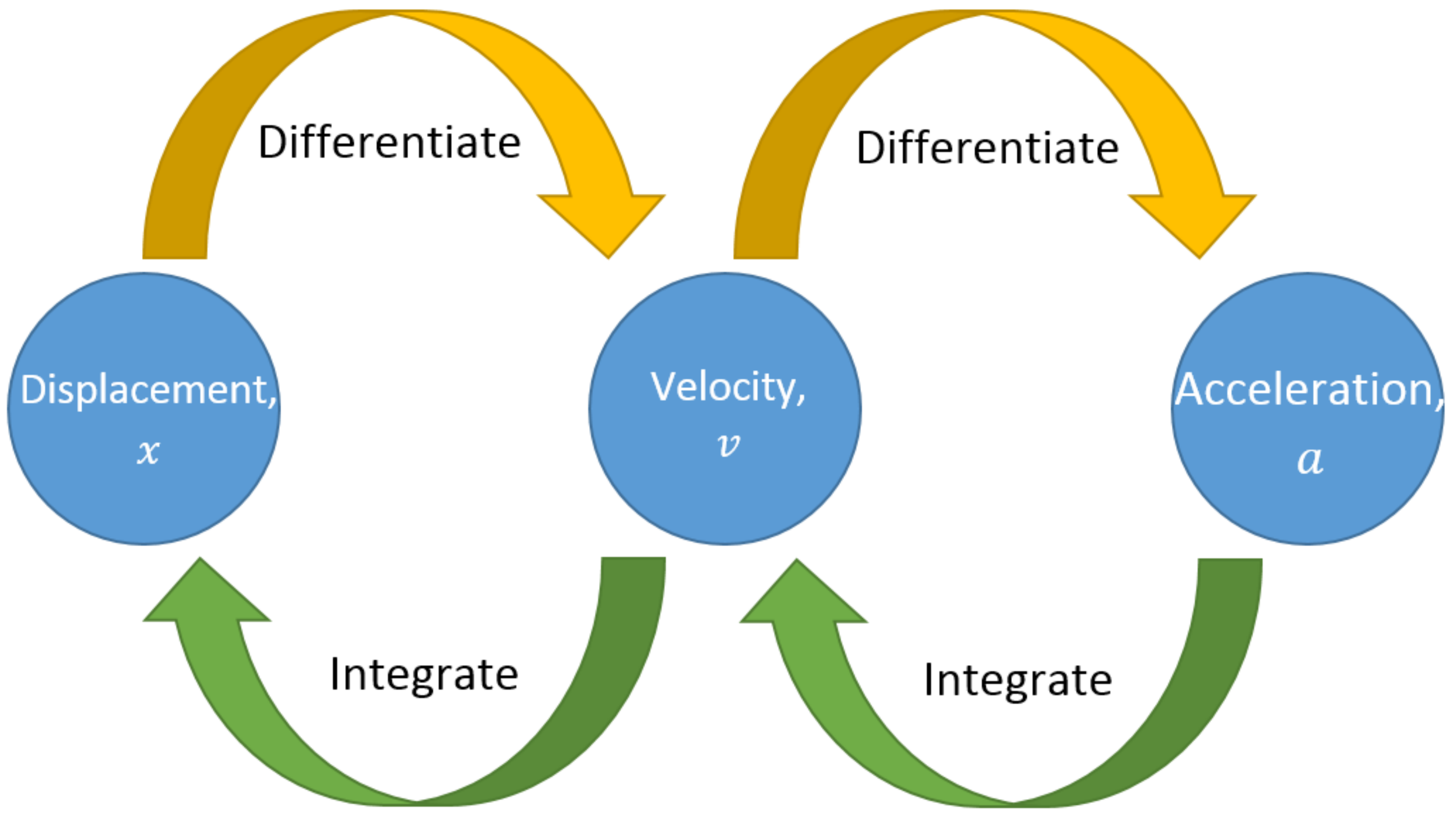
- Hardware gyroscopes and accelerometers give a fairly accurate reading of how the head is rotated.
- The same technology is in mobile phones and relatively cheap.

Positional Tracking

- Positional tracking is significantly harder than rotational tracking.
- Physics tells us that without a frame of reference we can’t tell where an object is or how fast it is moving. We can only detect changes in velocity (acceleration)

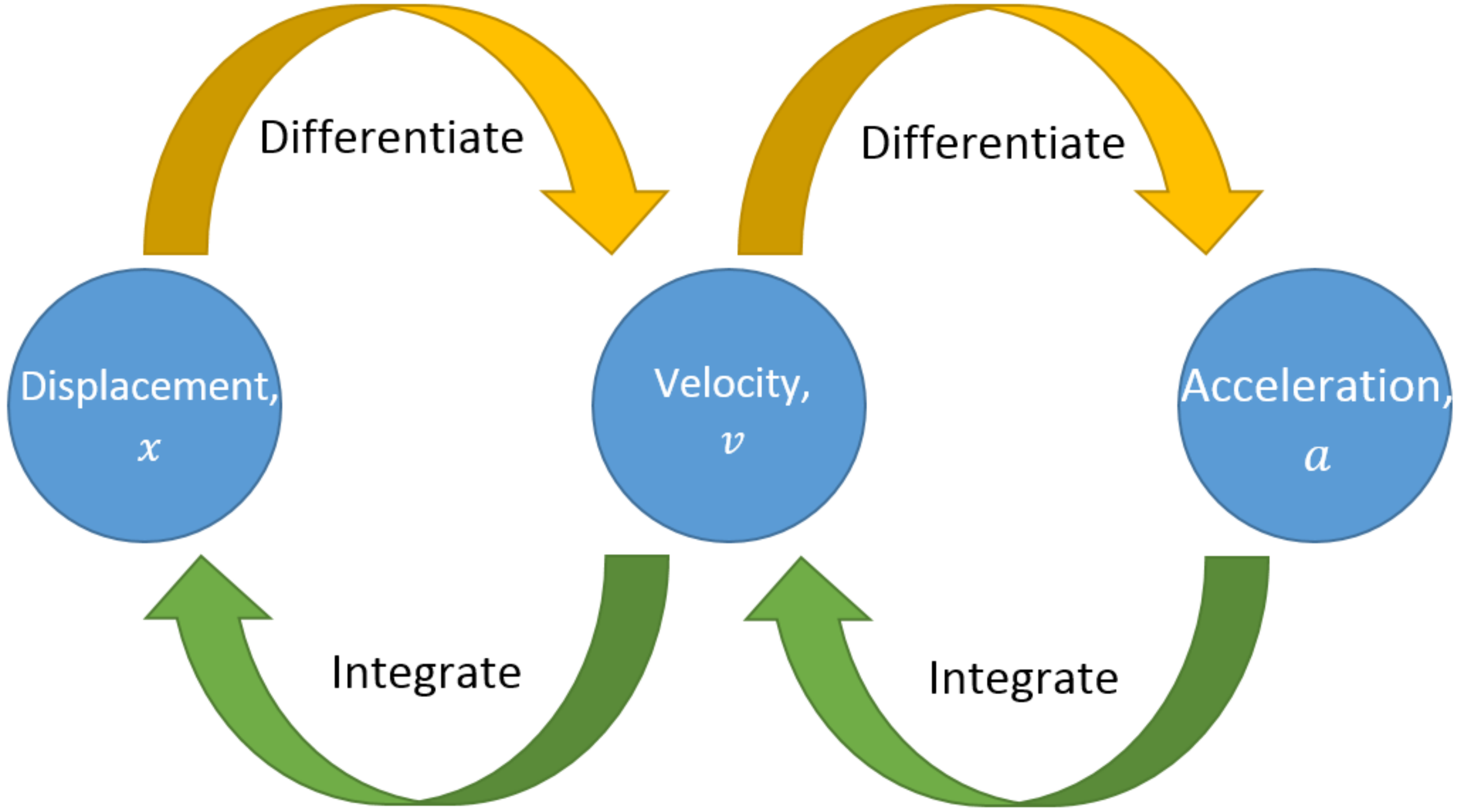
Dead reckoning

- The simplest approach is dead reckoning: using readings from accelerometers to estimate velocity and consequently position.



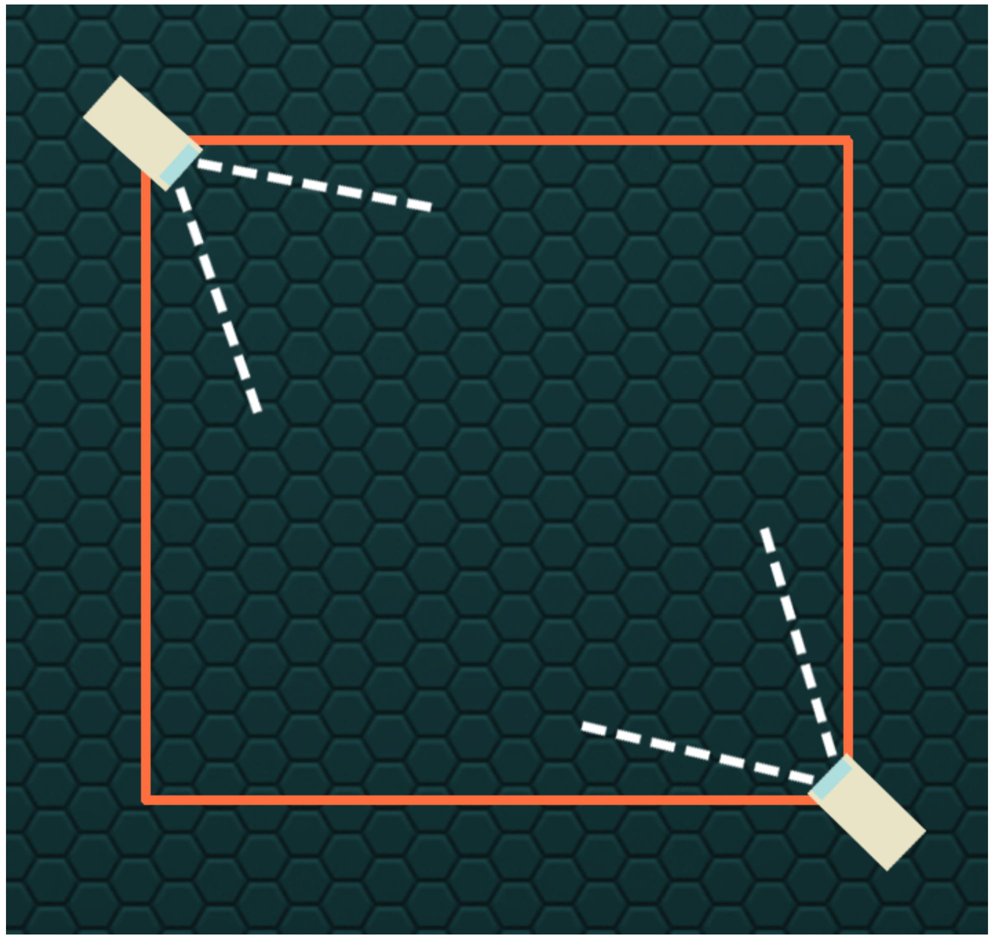
Dead reckoning

- This approach is very inaccurate. The estimated position quickly drifts from the actual position and the user becomes disoriented.



Outside-in tracking

- For accurate tracking, we need to include external information in the estimations. With outside-in tracking, external sensors (cameras) estimate location in a fixed space.



Outside-in tracking

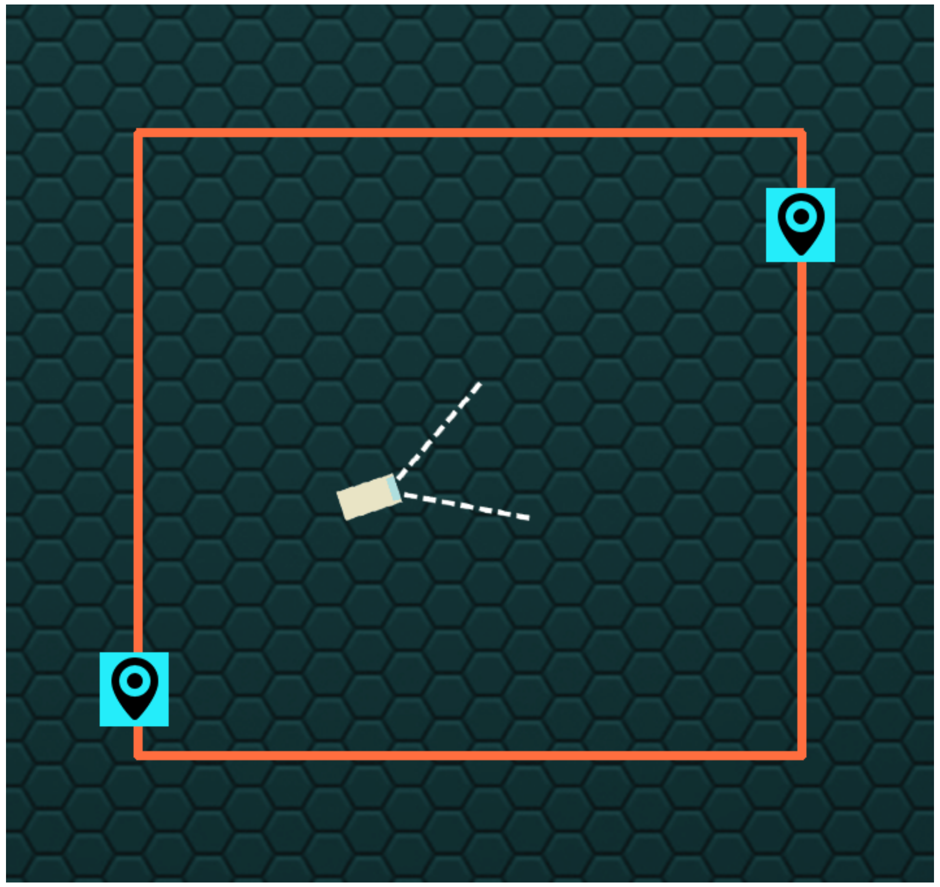
- Clear indicators are placed onto the headset for the sensors to detect. A popular choice is infra-red LEDs as they are invisible to human eyes.

Inside-out tracking

- Inside-out tracking avoids mounting sensors by putting them in the headset itself.
- This is typically cheaper as larger areas don’t require more sensors.
- The disadvantage is that headsets require extra power for the sensors.

Inside-out tracking

- Marker based tracking requires special markers to be mounted in the environment.



Inside-out tracking

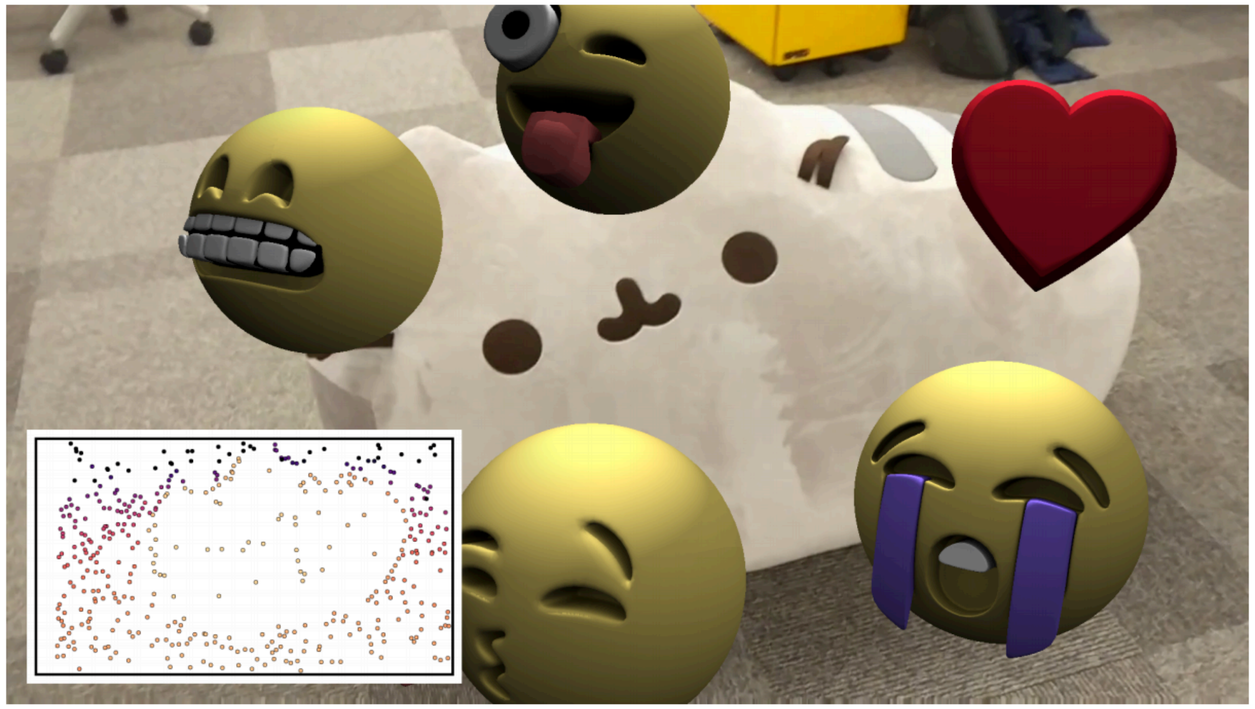
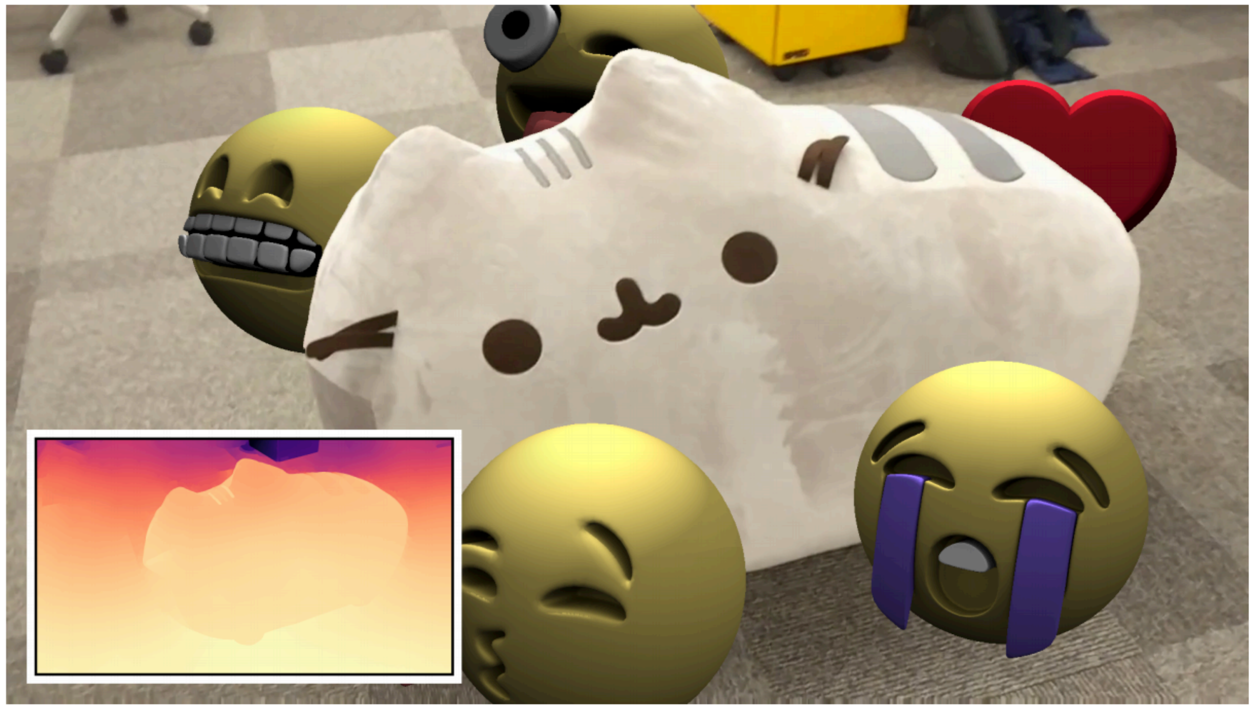
- Markerless tracking uses clever computer vision algorithms to detect features in the surrounding environment and track them.

Rendering in AR

- Rendering for AR is much the same as for VR. Either you’re using an image of the real world as a background (e.g. phone based AR), or you’re rendering onto a transparent display (e.g. Microsoft HoloLens).
- Tracking in AR needs to be even more accurate to create an immersive experience.
- One of the biggest problems is occlusion-awareness

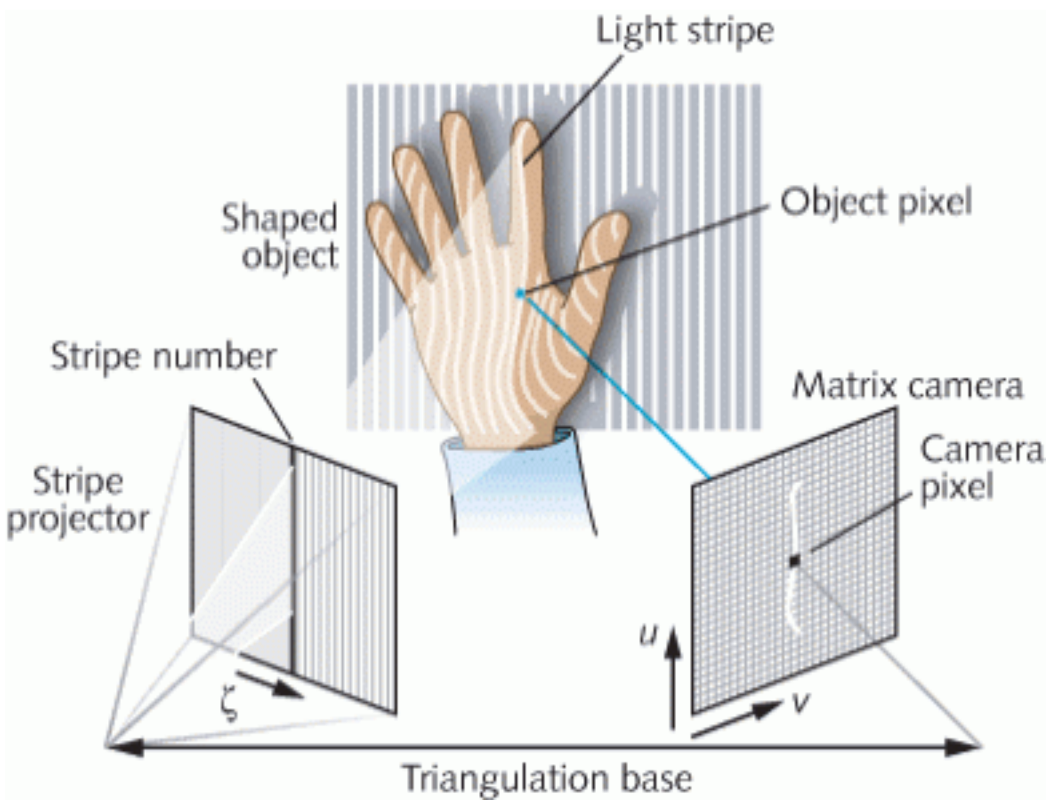
Occlusion-Aware AR

- In order to ensure that virtual objects appear behind real world objects, it's necessary to have depth information for the surrounding environment.

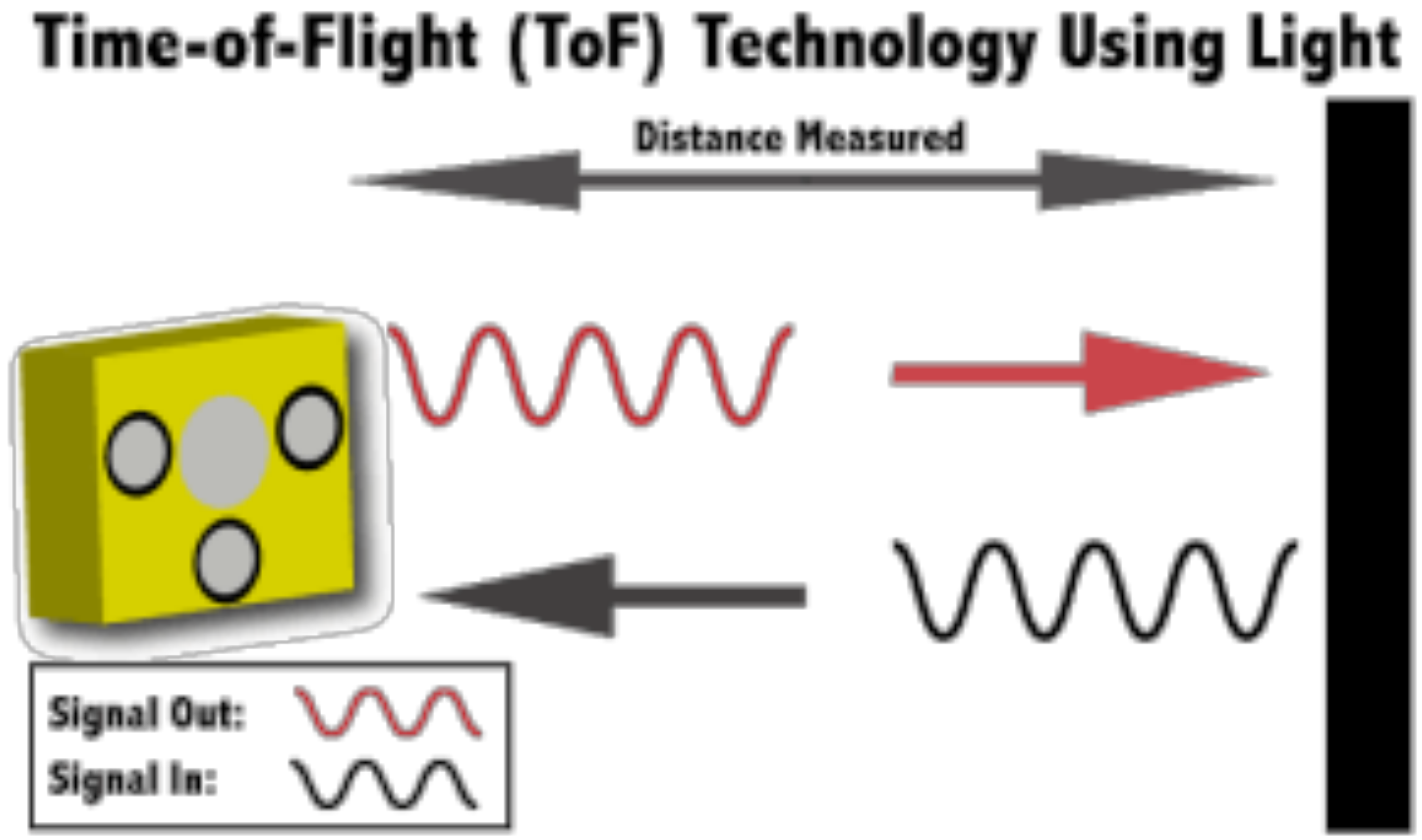
Detecting Depth

- Structured Light sensors sense IR light that has been projected as lines or a grid and analyse how they have been distorted to construct 3D information.



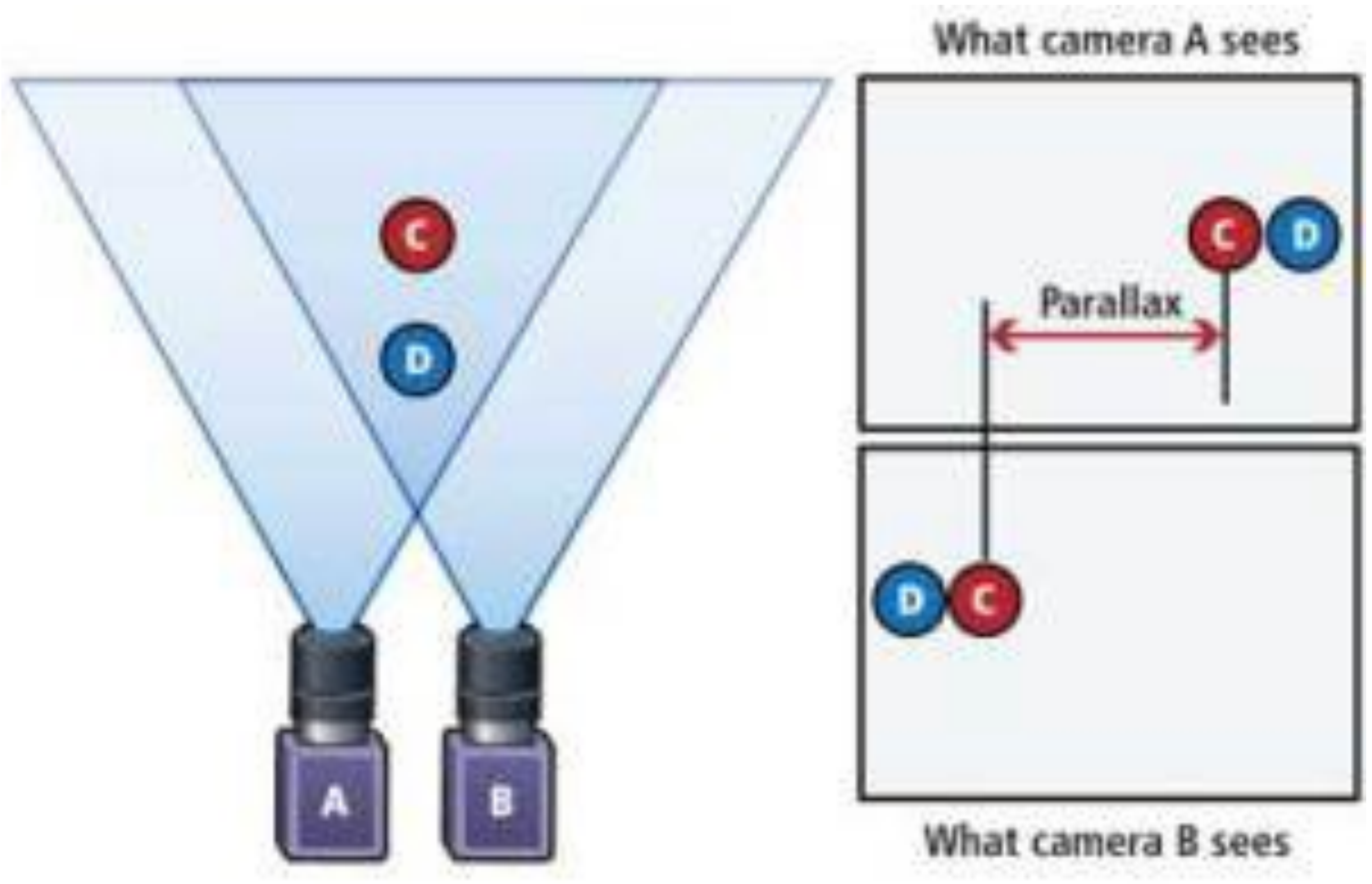
Detecting Depth

- Time-of-Flight sensors project IR signals and measure how long they take to bounce back in order to measure depth.



Detecting Depth

- Stereo cameras in conjunction with computer vision algorithms can detect the distance to distinct features of the environment.



Detecting Depth

- None of these techniques work for more than about 4M with current implementations in headsets.

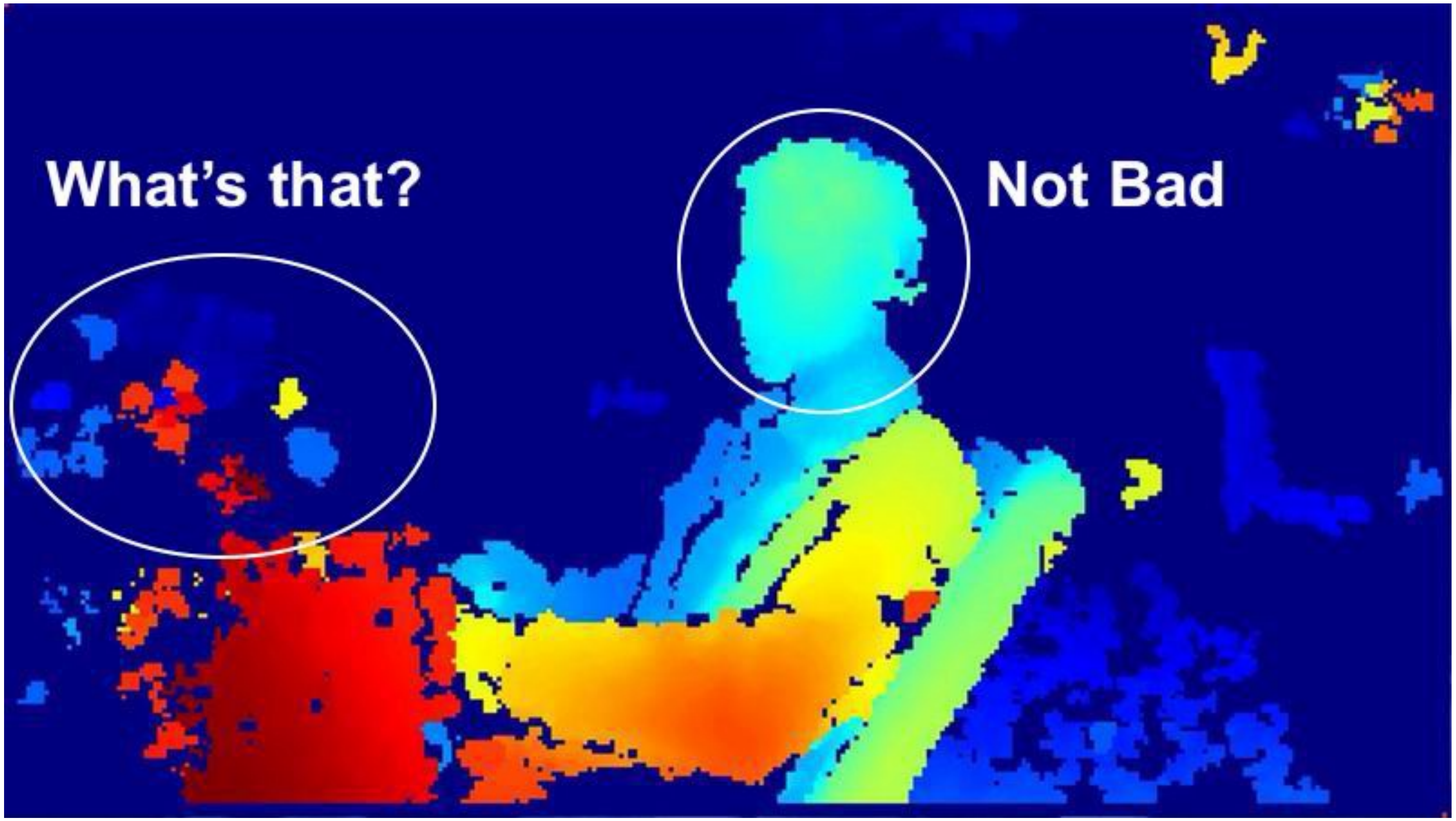
- IR based solutions don’t work well outdoors.

- Stereo camera solutions need environments with lots of distinct features.

- Results are very low-resolution

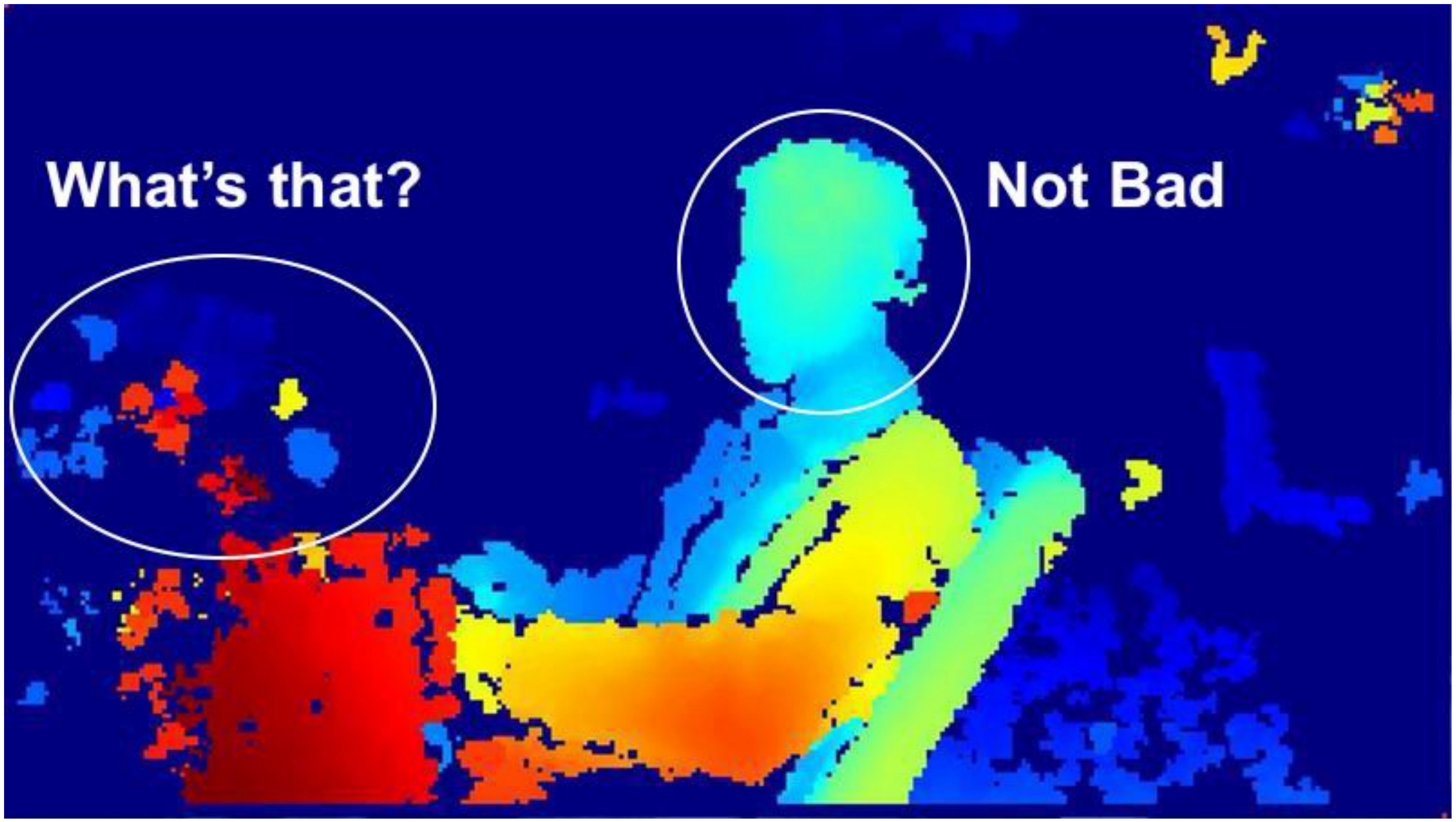
Detecting Depth

- A lot of noise is generated and needs to be accounted for.



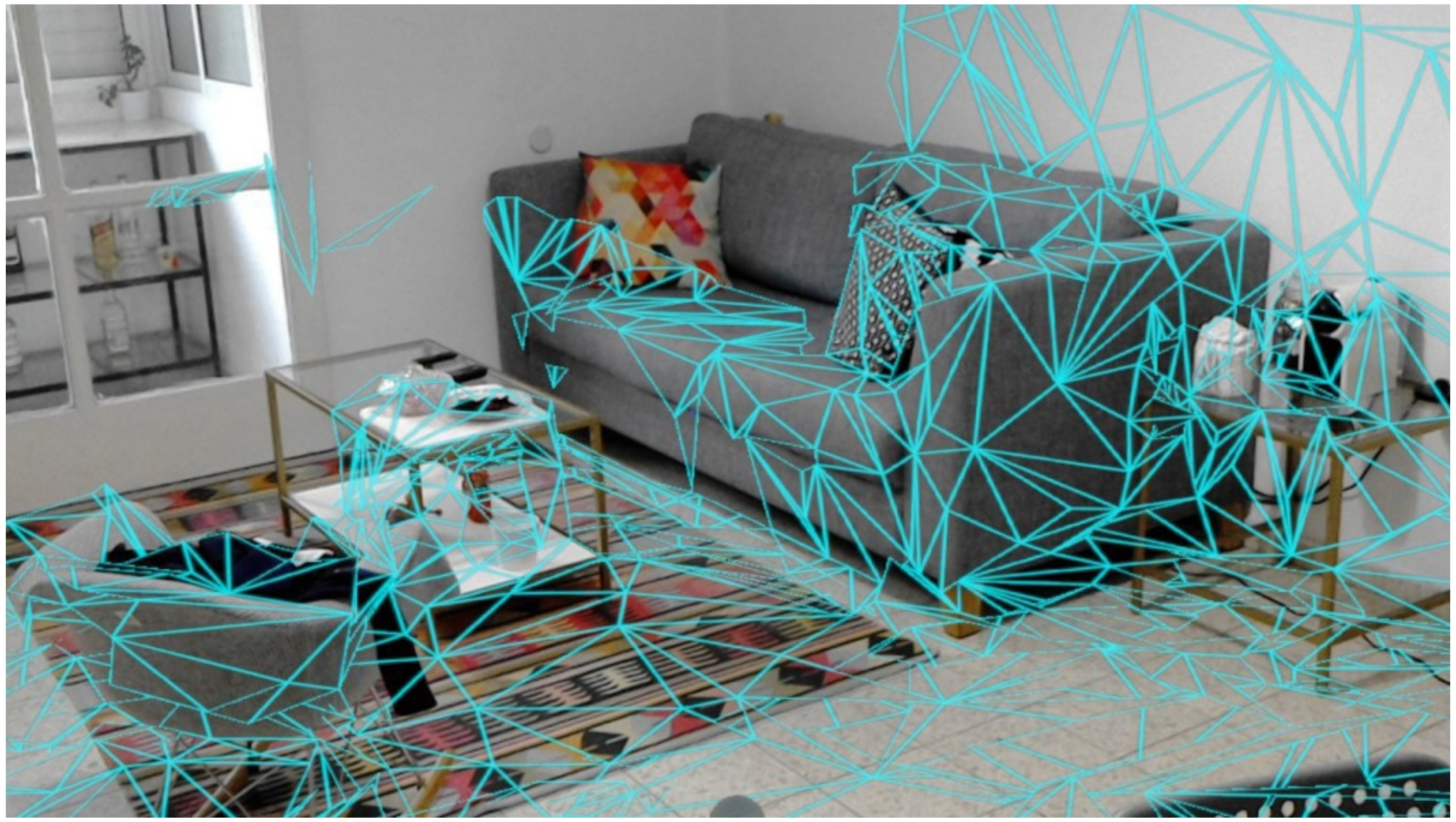
Detecting Depth

- Edges matter a lot.



Geometry reconstruction

- Instead of writing the depth information directly into the depth buffer, it often works better to reconstruct the geometry and render that into the depth buffer.



Time-Based depth information

- Using adjacent frames in a video can allow for depth information to be generated without any depth sensor.
- See: https://homes.cs.washington.edu/~holynski/publications/occlusion_sa2018.pdf
- Video teaser: https://www.youtube.com/watch?v=VjR11gkmm3E

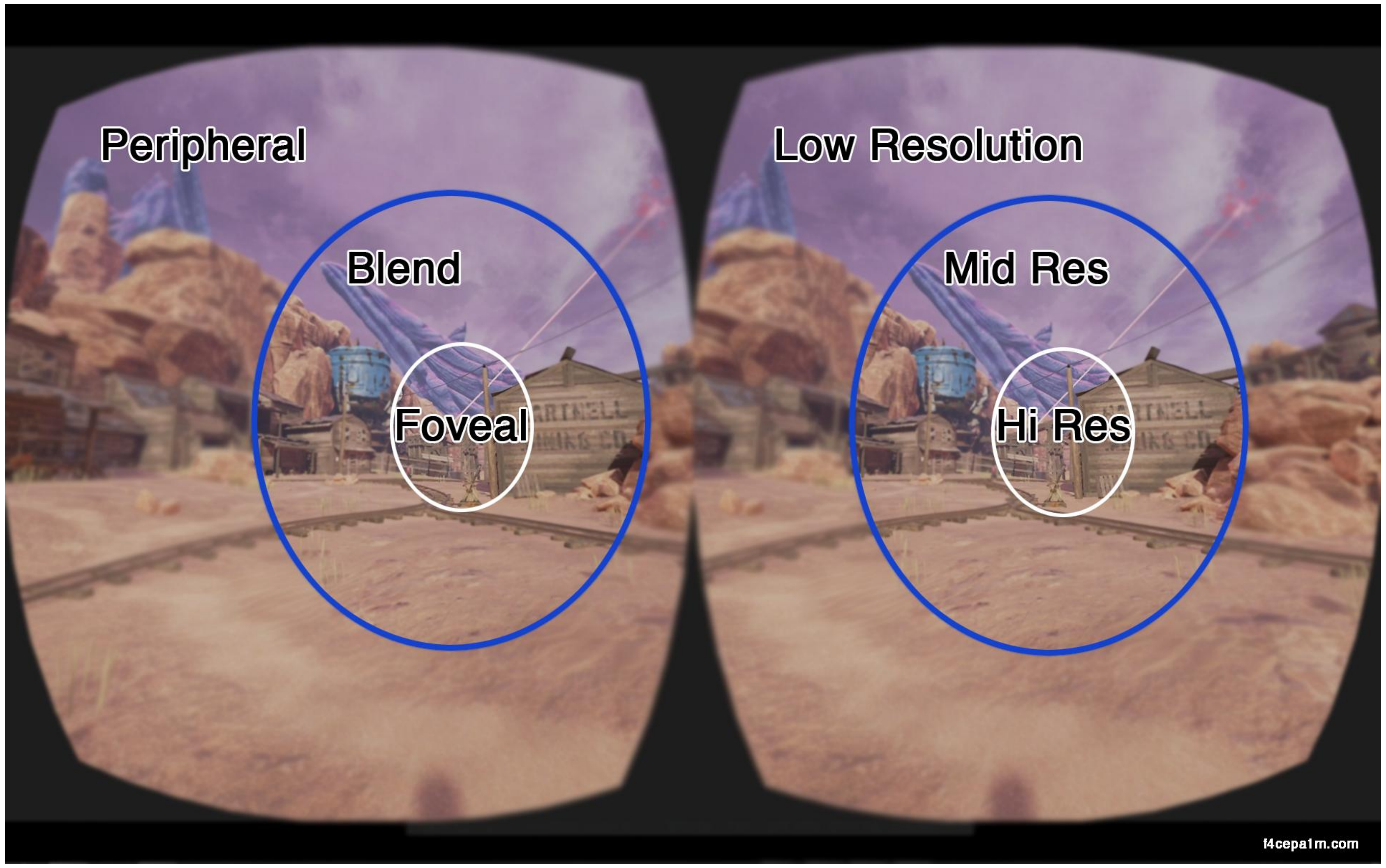
Eye-tracking

- The newest generation of VR headsets support eye tracking

- In addition to being an additional form of user input, eye tracking allows for foveated rendering

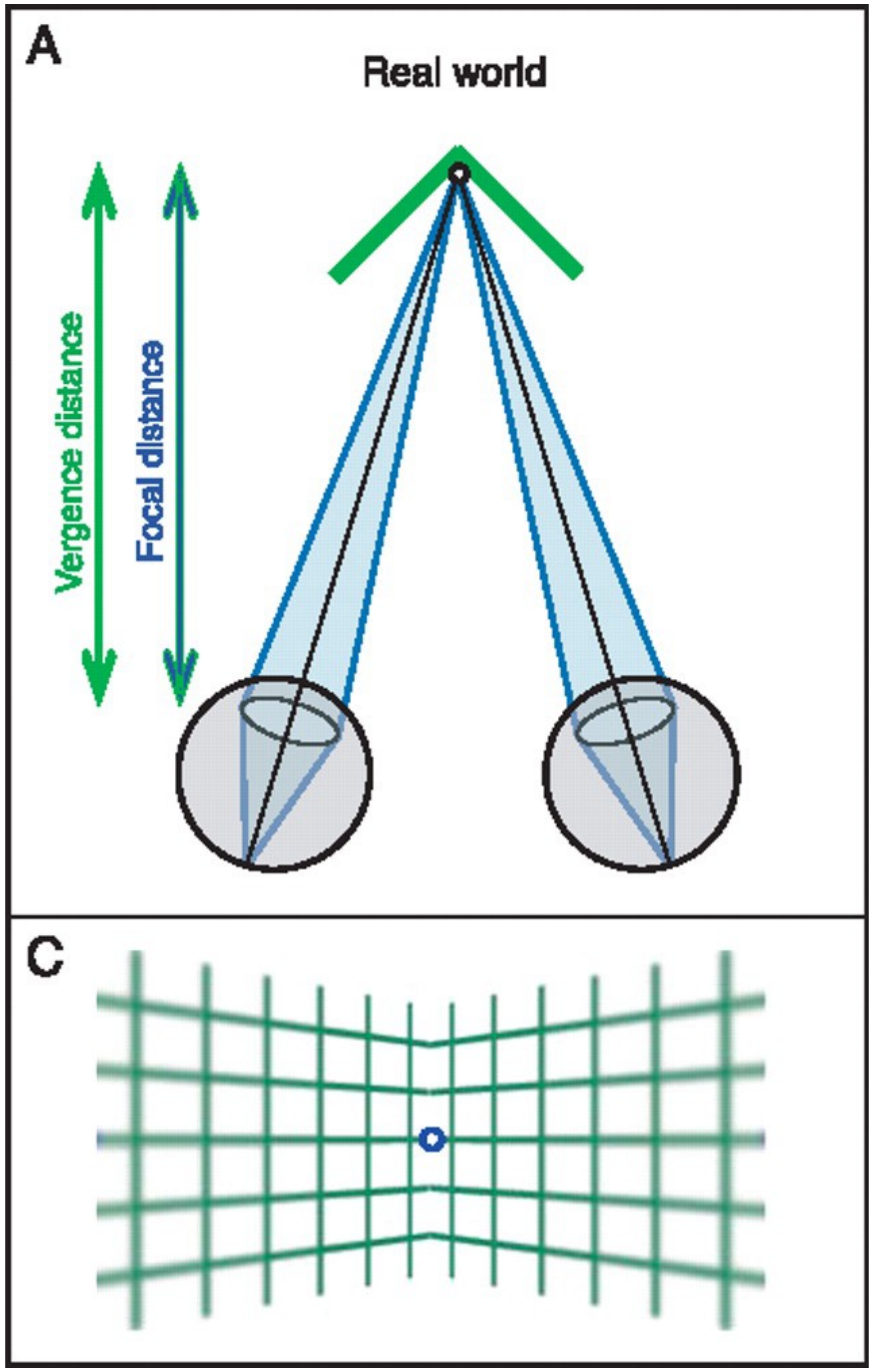
Foveated Rendering

- Foveated Rendering: render in full resolution only the section of the display the user is focused on.



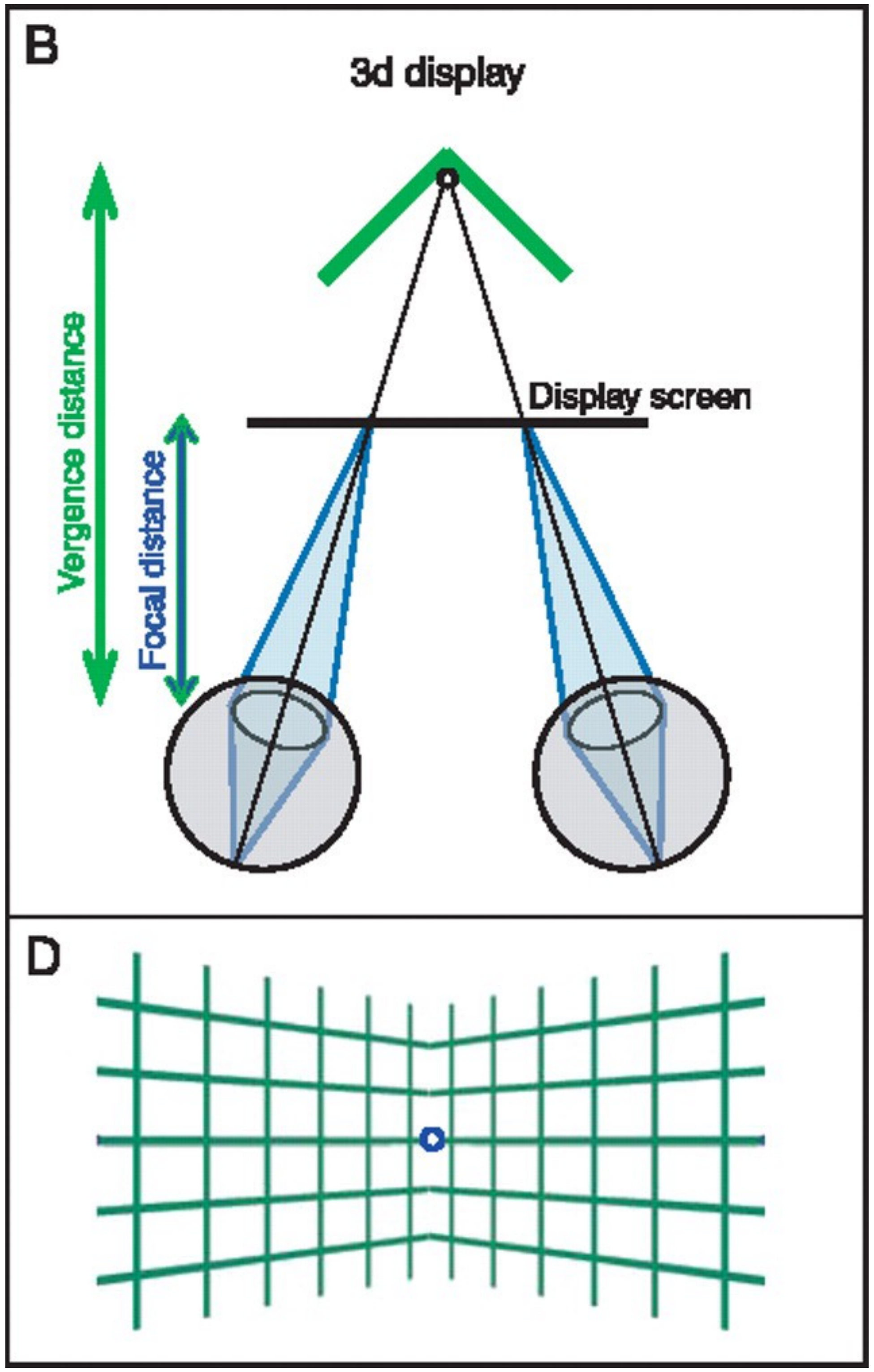
Focus

- Human eyes are able to focus on objects at different depths by adjusting both vergence (relative rotation of the eyes) and accomodation (the shape of the lens in each eye).



Focus

- With 3D displays, there is a mismatch between vergence and accomodation. This is referred to as vergence-accomodation conflict and can be disorienting, especially when looking at things up close.



Varifocal displays

- By quickly adjusting the distance between the lenses and the display, vergence-accomodation conflict can be somewhat mitigated.
- Either the lenses or the display can move.
- The movement can be based off eye-movement or what “should” be focused on in the scene.

Varifocal displays

- Varifocal demo: https://www.youtube.com/watch?v=PhQLIMIprBY

Exam!!!

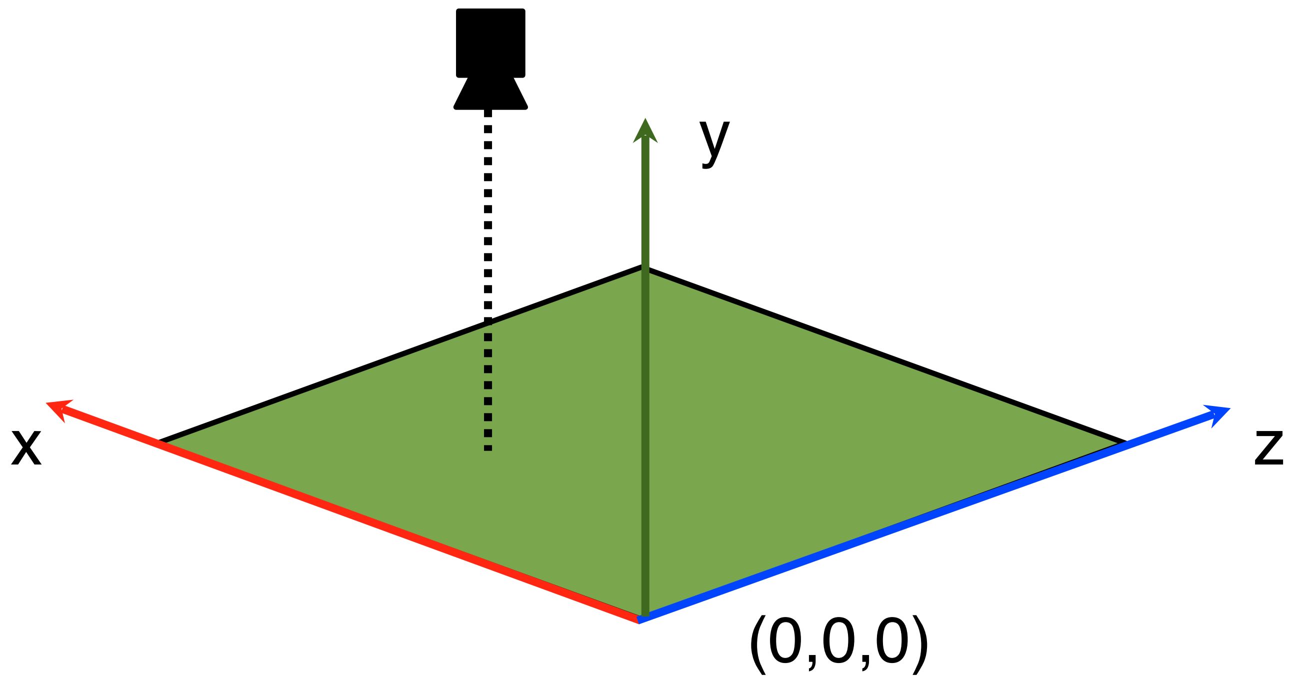
Exam

- https://www.cse.unsw.edu.au/~cs3421/19T2/exam/ExamInfo.html

Part A – Algorithms + code

Demonstrate use of an algorithm. Similar to many tutorial questions.

"In the scene shown, the camera is at (2,3,1) in world coordinates, and is rotated to point straight down. Calculate the view matrix for this situation. "



Part B- Definitions

- Definitions and advantages/disadvantages of different approaches.
- “What is a depth buffer? How is it used for hidden surface removal? What advantages or disadvantages does it have over the painter’s algorithm.”

Sample Solution

The depth buffer is a block of memory that holds the depth information for every pixel. For each fragment we draw, we calculate its pseudodepth and compare it to the value in the depth buffer. If it is closer to the camera, then we update the colour buffer and the depth buffer to the new fragment's colour and depth respectively.

The advantage of this over the painter’s algorithm is that polygons can be drawn in any order. However, it requires more memory and does not support true transparency.

Part C- Applications

- Questions that present a particular scenario and ask what methods you would use and why.

- "For a 2D game you want to generate a variety of realistic images of apple trees in winter, like the one shown.

- What kind of algorithm would you use to do this? What considerations affect your choice?"

Sample Solution

- L-Systems are useful for producing realistic botanical models. I would use a stochastic L-System to incorporate randomness to provide a variety of trees.

- Because it is a 2d game we would be able to render them fairly quickly. (If we were in 3d, we would have to be careful about run-time which can get high with complicated realistic 3d models)