A test I carried out to assert the applications and limitations of optical flows effectiveness on depth was to move slowly does a corridor with the Kinect pointing forward while recording. From examination of the results several interesting points were ascertained. When moving laterally in a fashion perpendicular to a flat surface with the same viewing angle and the same distance to the surface the relative depth values will not change and as such movement is not detected. This is better described by imagining a point in the view on the surface, as the view is moved parallel to the surface the same points depth value will change and it has moved but optical flow will not detect the movement as, in the views perspective, the same point depth hasn’t changed. Similarly, to how optical flow on luminosity cannot detect motion without visual texture, optical flow on depth cannot detect movement on a surface without depth texture.

In the following diagrams the following key is used:

* Blue triangle, the depth sensor
* Orange line, line drawn by a pixel from the sensor to a point
* Blue line, the line drawn when the central pixel is taken from the sensor
* Blue arrow, indicates a change of state

d

v

d = v

Θ1

Θ2

Θ1 = Θ2

No movement

P1

P2

P1

P2

When moving rotationally at a fixed location, on the other hand, the apparent position of the same point on the planar surface will move as before however this time the same depth value should be associated with it as the view has not traversed from its location, this should be picked up by optical flow as the point of the value has changed its apparent location.

d

v

d = v

Θ1

Θ2

Θ1 ≠ Θ2

Movement

P1

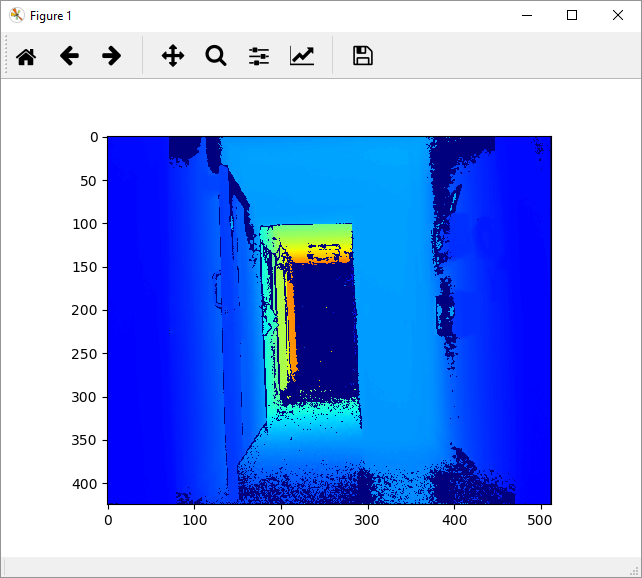
P1

We can see from the above that when rotating at a set point optical flow on depth should be able to extract movement from non-textured/featureless surfaces which optical flow on luminance may not regardless of light conditions. Instinctually we would say that combining the two solutions into a single combined program which used the depth in conjunction with luminance to provide a flow field would be optimal. There is no solution or library allowing for this to be done out there that I could find, and as open CV’s functions only allow a single image where each pixel is luminance as input without developing a standalone program I cannot test this.

We can compare the two different optical flow implementations provided by open cv to see how they compare when applied two separate test footage sequences recorded by the author:

* Down hallway
* Rotation hallway

To conduct these experiments, we first view the depth footage to ascertain its quality and any imperfections, below are two images, the first is from the Down hallway sequence and the second is from the rotation hallway sequence.



From this footage, we can see that the Kinect, as specified in the documentation, has a limited depth range of up to 8m, there are zero-value artifacts being introduced in various locations in the footage, this has been examined before and the sources of the values have already been identified.



Unfortunately, it is impossible to say whether the sources we have identified cover all elements of zero-value introduction as we can’t prove the absence of other sources. The sources identified for these zero-values are:

* Out of range, as indicated above the Kinect has both a maximum and minimum sensing range after which zero values are produced
* Occlusion, as has been previously stated in prior works, the location of the IR sensor and emitters on the Kinect v2 sensor are at different positions, this creates the effect that points on objects are hit by IR beams but the sensor is occluded from the reflections or the case where a point on an object is visible to the IR sensor but is occluded from the emitter meaning it will never reflect the TOF beam, note that this only happens horizontally and a more robust system could utilise a combination of two TOF sensors one being reliable in the horizontal plane and one being reliable in the vertical plane.

Emitter

Sensor

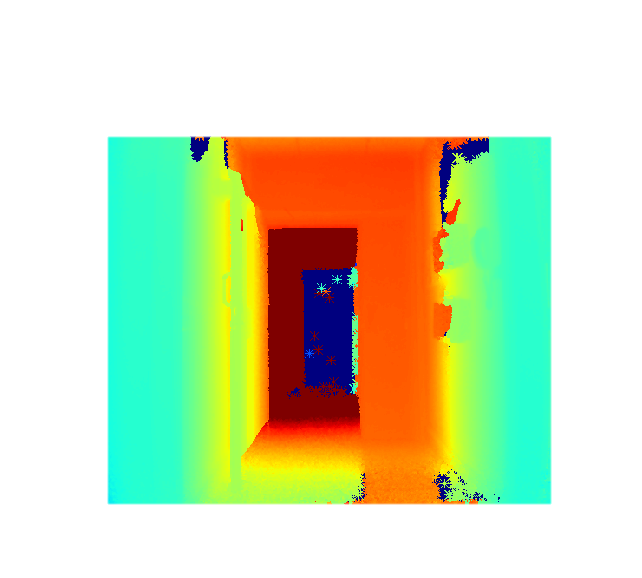
From this diagram, we can see that the only components, in this case, of the object which are correctly analysed is the frontal plane.

* Reflective surfaces, where a surface is reflective in the IR spectrum the IR beams projected onto the surface are reflected meaning that the detected object is not the reflective surface but whatever it is reflecting. As described above the component detected will be the object with zero-values at each horizontal edge.
* Interference, Interference from other IR sources is theorised to be able to distort the data although I have not done any testing which supports this due to a lack of having access to an IR emitter which is not internal to the Kinect.

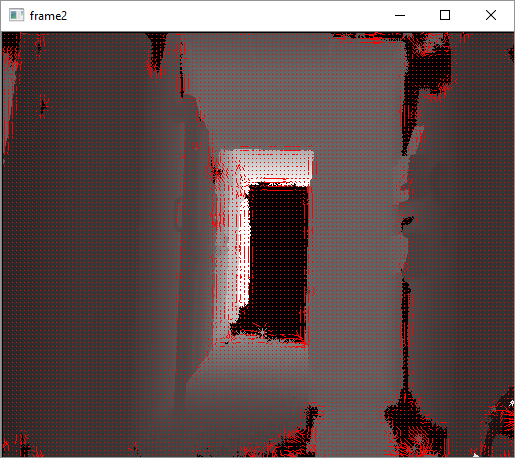
By applying our naïve noise remover (Naïve because it relies on the fact that the scene background is static, to port this into this context I’ve removed the confidence calculation), we get a slightly less noisy sequence, the following is two exerts in the same order as before.

By applying Farneback optical flow to these sequences we get some interesting results as detailed below, the parameters for the Farneback procedure are as follows.

* Flow: The previously calculated motion field
* Pyramid scale: 0.5
* Pyramid levels: 1
* Window size: 4 (Size of the motion field used when calculating a points value, blurs the whole image)
* Number of iterations: 2
* Polynomial expansion size: 10 (Size of the pixel neighbourhood used when finding the polynomial representation, larger values gets blurs the features used allowing optical flow to generalise better but too large means it will lose details)
* Polynomial sigma: 1.2 (standard deviation of Gaussian used in polynomial expansion)
* Flags: 0

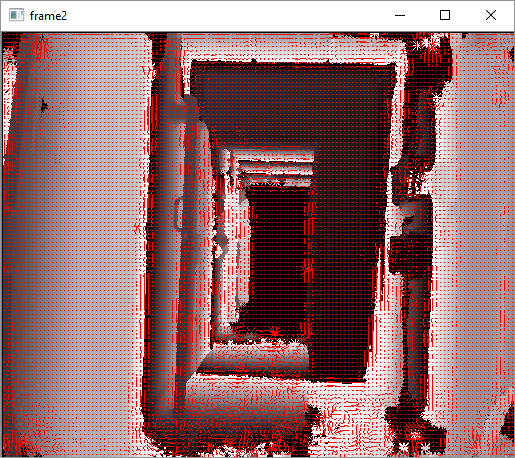
  
(There are some artifacts in this noise reduction which are due to noisy real world data, not much can be done about these.)

We can see that this is a much cleaner image as there are no longer any large spots containing values flickering between zero and their actual value although some spots are less accurate than others. We can apply Farneback optical flow to this and get the following motion field, this motion field has been overlaid on top of the depth footage, the depth footage is represented by a greyscale image and the flow field is the matrix of red lines representing magnitude and direction.



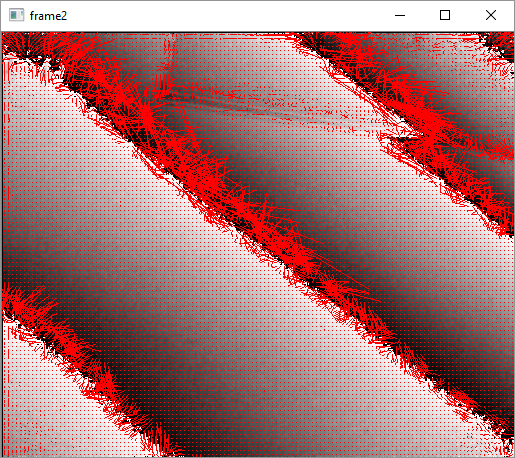
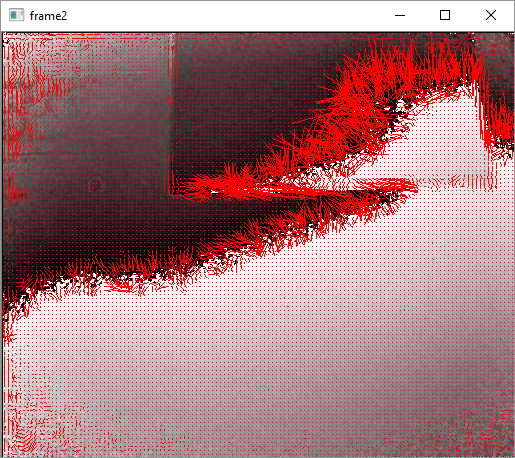
Aside from the flow introduced by noise we can see that, overall, the flow field is relatively static and only produces some small flows in the correct direction at points of interest in the sequence, i.e textured objects/edges.

In fixing bugs and increasing effectiveness of the implementation I accidentally changed the following expression. np.array(frame2 \* float(255/self.file.getFootageMax()),"uint8"). This normalises the values to be between 0-255 as required by the farneback function provided by open cv, this obviously loses a large amount of fidelity as a range of 8,000 is converted to 255. I accidentally changed the expression to be np.array(frame2 \* float(2000/self.file.getFootageMax(None)),"uint8"). Whereas I would have assumed that this would simply not work as the farneback function requires a matrix where each element is between 0 and 255, it had the following effect.

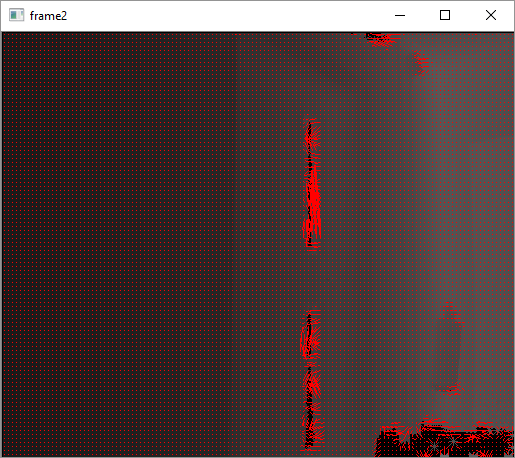


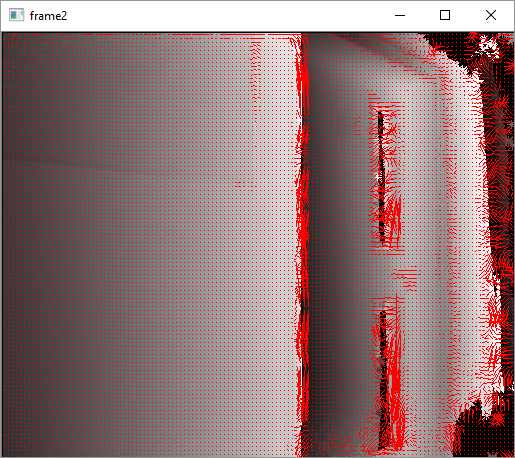
I think this happens because the Farneback function takes the modulus of the input with 255 meaning that, normally the values are between 0-255 for the entire range but now the values have been pushed into 7.8 ranges of 255 which has an effective range of 0-2000. This allows us to squeeze much greater fidelity into the optical flow algorithm. This will not work in all cases of course as this is not the purpose it was designed for. Instantly we can see that we cannot increase the fidelity indefinitely as it would approach a point where every point is almost identical in value to every other point due to the 255 range being equal to the distance between the two points. Optical flow noise would also be very prevalent as there would be an abundance of similarly valued points in the neighbourhood as shown below.



From this accident, it became apparent how farneback didn’t care about individual values but more identified hard boundaries and features. This is shown below by performing a similar range squash to footage of rotating the sensor in a stationary position focusing on a flat planar surface, a wall. 

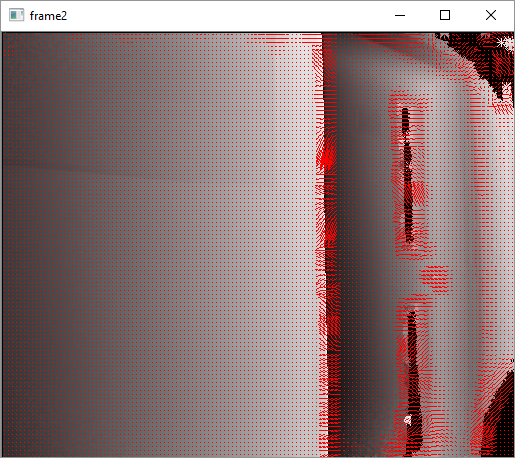
By applying the two forms of farneback optical flow we can get the following comparison.

0-255 range, there is little to no motion in the general motion field and it’s only at the points where there is a sharp change is there any flow detected.  


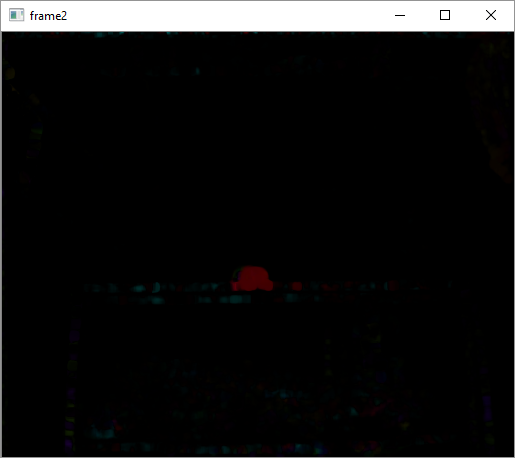
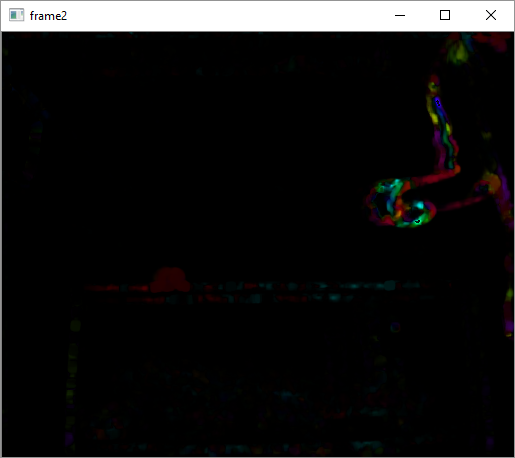
0-3060 range, as we can see, due to the higher granularity, the optical flow now detects motion around the door frame and around the edge of the notice board in the top left quadrant.  


By increasing the window size and polynomial expansion size we can get a more blurred motion field better approximating the general motion of the scene. This works very effectively with farneback arguments:

* Flow: The previously calculated motion field
* Pyramid scale: 0.5
* Pyramid levels: 1
* Window size: 15 (Size of the motion field used when calculating a points value, blurs the whole image)
* Number of iterations: 2
* Polynomial expansion size: 20 (Size of the pixel neighbourhood used when finding the polynomial representation, larger values gets blurs the features used allowing optical flow to generalise better but too large means it will lose details)
* Polynomial sigma: 1.2 (standard deviation of Gaussian used in polynomial expansion)
* Flags: 0



An addendum is that for some reason optical flow effectiveness seems to be better and worse in alternating frames, this is probably to do with some issue in my implementation but I have not had a chance to find it. This occurs above as well where the magnitudes of one frames flow field is smaller than the next.



Note how the ball changes colour indicating a change in magnitude even though the ball is moving at the same rate, this is not due in any way to the figure on the right being introduced as it happens for the entirety of the balls motion.