Findings:

SuperPoint + SuperGlue:

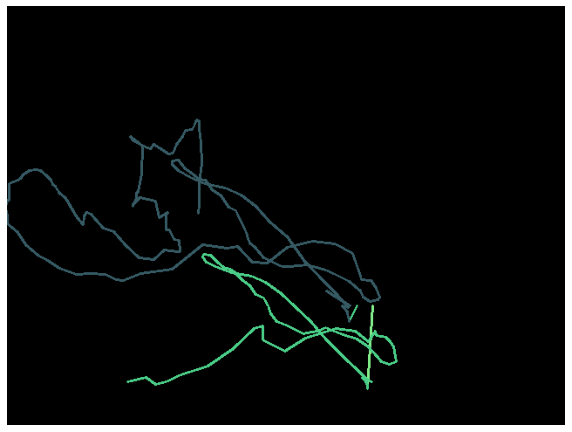
* Requires a very low threshold for both kp and matches else nothing will be found
* I used 0 for kp threshold and 0.01 for matches threshold
* Much more effective than sift
  + Sift results on testing image: kp=355:405, matches=355
  + Super results on same image: kp=7783:7353, matches=1681
*  (Left Image)
* A picture containing pencil

  Description automatically generated(Super)
* Sift is only comparable when this Infrared light is used, doesn’t matter for Super using Infrared or not, still finds the same amount which is great for larger objects the infrared doesn’t reach.
* Use parameter nms\_radius for much more keypoints found, as well as keypoints found more accurately on very similar looking images such as the stereo images from d435 camera.
  + Radius of 1 means every 1 pixel a new kp can be found, if 5 only every 5 pixels a kp can be found
    - Radius of 1 requires a lot of memory however on an image, hence a higher one is usually preferred, 2 is about the minimum
  + If radius is 5 the distances between kp are min 5, so finding depth from a kp in left and right will be hard to do as the kp on each image are at the same point with radius of 5 where they may differ by 1 or 2 pixels

KLT (Lucas-Kanade Tracker)

* Method:
  + Detect Harris corners in first frame
  + For each Harris corner compute motion between consecutive frames
  + Link motion vectors in successive frames to get a track for each Harris point
  + Introduce new Harris point by redoing Harris every m frame (10-15)
* <https://www.youtube.com/watch?v=tzO245uWQxA>
* Opencv version of klt:
  + <https://docs.opencv.org/3.4/d4/dee/tutorial_optical_flow.html>
  + <https://docs.opencv.org/3.4/dc/d6b/group__video__track.html#ga473e4b886d0bcc6b65831eb88ed93323>
* Works well with superpoint as the points to track rather than Harris corners
* Display video of it in progress: Depth Estimation/Videos/Video1-Tracked
* Suffers when a kp is occluded by another object, will begin to follow its new path of where contact was made. E.g. a kp is on my bed and my hand swipes over it the KLT will start to track where the kp hit my hand.
  + To solve this problem I can either:
    - Ignore points that are not uniform to the majority of movement I am looking at, not part of the building so ignore
    - A person wearing headphones and holding a phone

      Description automatically generated with low confidenceDetect kp every few frames and check to see if the kp is still in the frame (kp matching can be used) -> if kp not matched to next frame under a threshold the point has become occluded and is now useless and we can loose it
* How does it perform with moving camera?
  + Performed very well given that my holding was very very shakey and it still kept its tracking really well, lost 1 of its 2 points but kept 1 which I was very surprised about
  + A picture containing game, playing, controller, indoor

    Description automatically generatedThe mapping across the 2 points was very similar up till the second point was lost
* How does it perform with moving camera and moving object?
  + Works just as good as the stationary camera as seen below the tracking on 2 points is very similar
  + The pink trail is lost as the kp went of the camera, is it possible to regain the kp later on and keep tracking it?
  + Some slight misses near the top left of the mask (below image) due to lower quality and again not great stability from my hand



* How does it perform with vibration?
  + Worked well, computing the changes in x and y direction they ranged from 0.044 pixels to 22-pixel changes allowing for small and larger vibrations
* Next step:
  + Compute KLT on stereo pair and find depth at each frame to get a 3D path tracker

KLT tracker on stereo pair:

* Algo suffers from quick movement, looses the object, also suffers from turns that are quick, the slower the movement or the more frames (I used 6 fps at higher res) per second used will increase the accuracy of the algo.
* Parameters to tune:
  + Winsize:
    - Low, looses the object when moving to quick
    - High, looses the object as searchers much more space and can jump to different points
    - Medium about best at 15
    - A green line on a black background

      Description automatically generated with low confidenceA picture containing night sky

      Description automatically generatedChart, line chart

      Description automatically generatedbelow example: 9-15-20 on stereo-video-1
  + Maxlevel:
    - Low, bad at corners and looses object
    - High, pretty good, takes longer
    - Medium, also pretty good
    - Chart

      Description automatically generatedA green line on a black background

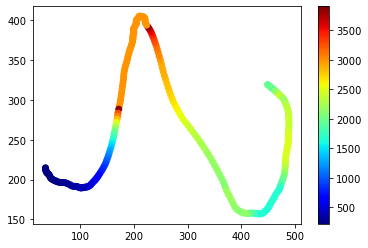
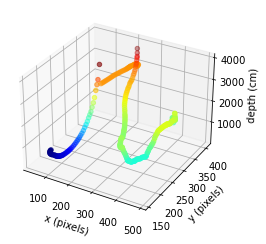
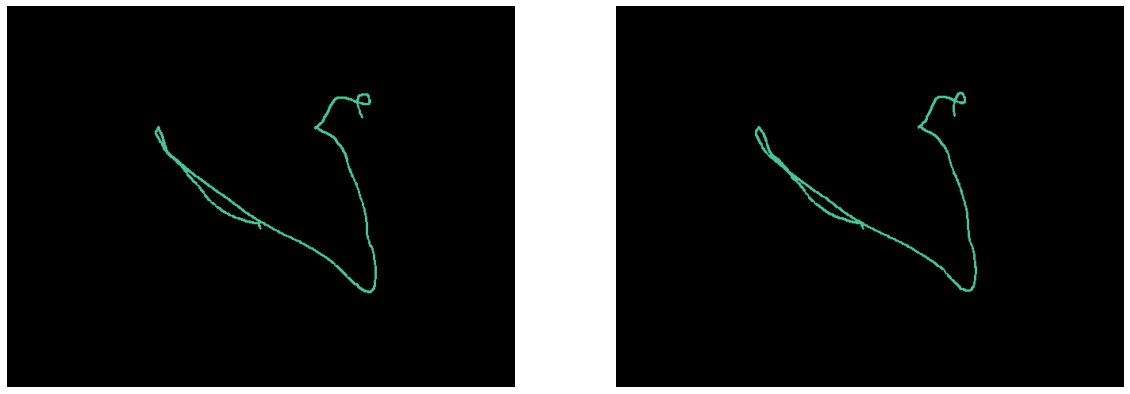
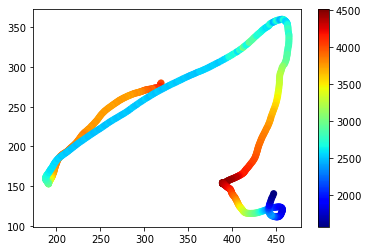
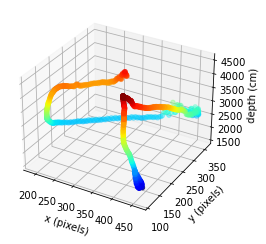
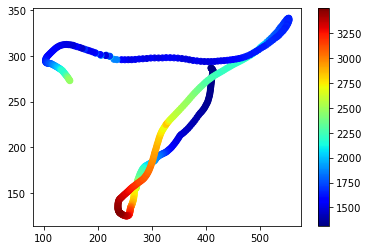
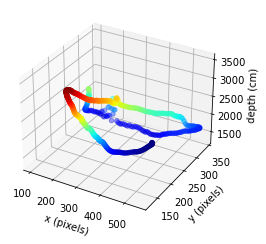
      Description automatically generated with low confidenceA green line on a black background

      Description automatically generated with low confidenceBelow example: 1-5-10 on stereo-video-1
  + Findings:
    - Doesn’t change a whole lot, it’s the footage quality and fps speed that makes the big difference, given the object doesn’t move to fast and you have a higher fps than 6
* Using fps of 30 with a lower reso (640x480 rather than 1280x720):
  + Performance is much better, tracking is more accurate
  + Tested on stereo-video-3 which is a slower paced moving object along with the higher fps its kp is kept the whole time and the tracking seems perfect
* Tracking across both left and right images and see the difference:
  + A picture containing dark

    Description automatically generated
  + Left to right mask of stereo-video-3
  + Some more points computed:
    - A picture containing text, laser

      Description automatically generated
  + Perfect matching between the 2 images, 0 difference between the 2 lines (point tracked)
  + Can’t compute depth from this however as each tracked location is exactly the same?
    - Might need to compute some rectification on the images and then find depth from there? Not sure how to do that
    - The kp starting location is the same in each image and it follows the same path to the pixel? This shouldn’t happen right as left and right image pair should see the same point at different locations?
      * Silly me, the left and right videos where both from the left camera…
      * No wonder it all was the same xd
      * Now to properly assess it…
  + A picture containing text

    Description automatically generated
  + Left -> right of new stereo-video-3, can clearly see the difference of x points but same y scale
  + From these saved points we can find the disparity between left and right images and hence the depth of the point.
  + As we know the framerate and movement in the x, y and z direction we can find the displacement and direction of the keypoint.
    - 3D output from the 2 L and R views:
    - Chart

      Description automatically generated
    - As we see the depth when we get to close to 0 it spikes (cannot find depth there)
      * Because disparity is really small the depth=b\*f/disp therefore making it spike really high depth
    - The 0 depth dots are where the disparity was too close (pixel range of 1) so I set the depth there to 0 (the range I found is at 0.5m is too close to the camera -> starts to spike at that distance)
* Trying out 3D point mapping on a few trials:
  + Closer to the camera: (stereo-videos-close with point 648)
    - 
    - Looses tracking but still computes a great depth? Not sure maybe something with drawing the mask went wrong?
    - 
    - 
  + Far from the camera: (stereo-videos-far with point 229)
    - 
    - 
    - 
  + Everywhere: (stereo-videos-far with point)
    - 
    - 
    - 
    - This one has an output video named everywhere\_l-tracked (same for right view)