The setup

The current setup consists of an AXIS camera (AXIS Q1615 Mk II Network Camera) and a Velodyne lidar (HDL32E), connected via a D-Link switch (DGS-1008-P) with POE(power over Ethernet) capabilities to a computer running windows.
The Axis camera is powered over the switch POE while the Velodyne lidar is powered from a 12v power brick. (it is advised not to connect the lidar box ethernet jack to a POE ethernet connector as the Velodyne box does not specify if it supports it.)

Camera

The axis camera is an outdoor camera that has a 120 degree camera lense

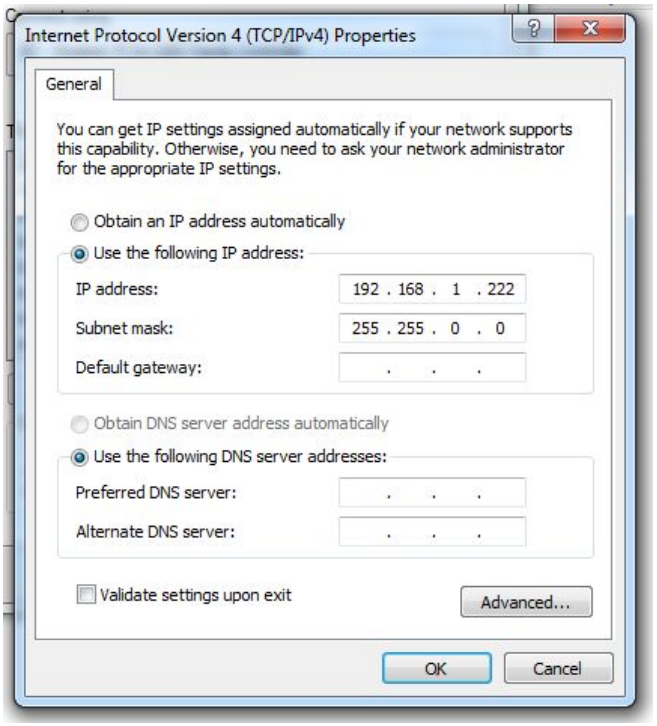
Lidar

The lidar scans the environment with 32 laser beams 8 of them points up from straight in front in a 10-degree range, and 24 of them point down in a 30-degree range. Aka its vertical FOV is 40 degrees and the Horizontal FOV is 360.
To improve the collected setup, the lidar is tilted back 10 degrees, aligning the centre direction vector of the laser beams and the centre direction vector of the camera.
This allows for 8 of the laserbeams to be more in line with the camera and therefore provide better overlap with the camera view.

The lidar is a class 1 laser product meaning that it is safe to stare at with the naked eye (but don't stare at it for hours).

The computer is connected to an arbitrary non-POE port in the switch. The lidar is sending its data in a UDP format, while the Camera acts as an HTTP server that can be requested using TCP.

To gain access to both the lidar and the camera at the same time, the PC's IPv4 setting needs to be configured.
It needs to have a static IP that the lidar can send data to, this is specified in the lidar settings(but in this case we use one of the default ones 192.168.1.222).
The right subnet mask level also has to be specified to have access to the camera as well as the lidar(in this case we use 255.255.0.0 to both get the lidar at 255.255.255.0 and the camera at 255.255.0.0) as seen in the image below.



The software

The current software exposes the functionality to record, save and playback recordings using a Qt GUI, the overview is that there is a window with buttons that you use to press record stop save etc. and then windows will open showing camera feed and pointcloud feed.

When operating the software uses a multithreaded approach to simultaneous communicate with the Lidar and Camera and collect/save the data in realtime. The output data format is just an arbitrary binary file with (x coord)(y coord)(z coord)(reflectance) and for the camera a jpeg file is used to minimize the size of the collected data.

Lidar

In software
The horizontal FOV can be configured in its settings to only send back an arbitrary range of its view, in this case only the 120 degrees of point cloud data in front is wanted as this is the only data that overlaps with the wide angle 120 degree camera FOV.

Camera

The camera is using a 120-degree wide FOV lens to widen its view to be able to correlate with more lidar data. This does also lead to a larger warp in the image data which should be corrected in the post processing.

The AXIS camera functions through the AXIS HTTP API that both allows for HTTP requests but also web-based configuration. There are two ways that data can be delivered from the AXIS camera, there is the video streaming option, which streams a video with a chosen encoding, and an option to take a snapshot.

Under testing in a test program using HTTP requests, it was determined that there is a random startup delay that takes around 250-500 milliseconds for the camera to start its (probably custom h264 encoder) and get a video stream running. This delay is probably dependant on the temperature of that camera and other factors. The delay makes the video streaming option unsuitable for the consistent acquisition of video. As this leads to unsynchronized data.

The Axis camera does support live streaming(H.264), this however does have a random startup / request delay around 300ms+-100ms from pc to camera through the router.
It is believed(although hard to fully prove) that this delay is caused by the internal custom H.264 streaming protocol on the limited camera hardware in the camera, a further complication of the hardware is that as the camera warmes up(at least indoors, they are designed to be outdoors cameras) the framerate of the H.264 stream varies complicating synchronization(most likely due to heat). And although

The other option that the camera has is a request for a single image in a variety of resolutions and compressions, this does not suffer from the time delay as there is no streaming server that needs starting, it is, however, slower depending on resolution and compression amount wanted. This is not a problem as the lidar scanners max frames per second is 20, therefore requesting an image from the camera at 1080p resolution with 50% compression 20 times a second was measured to be feasible.

Under testing the time it takes to receive the data from the camera varies
under
the time it takes to get data very much depend on the hardware used.
It should be noted that the manufacturer's choice of using the UDP protocol is fast but not safe, so packet loss can happen(but rarely do) generally it happens in bursts where the lidar will lose a few packets leading to a loss of 30-50 degrees of the 120-degree FOV. In the system, these are not recorded.

The current software

The software is constructed in C++ using PCL(Point) (Which is not necessarily the most optimal choice for development as getting the CUDA dependencies for PCL to compile is difficult)

The code repositories

The repository for Stefan's lidar code and lidar documentation
https://bitbucket.org/Crotaphytus112/lidar_processing/src/master/

The repository for Julius' camera, lidar code and documentation
https://bitbucket.org/aauvap/lidar_software/src/master/

A test dataset was recorded on traffic, it can be found on Morten bornø server, or on the hard drive Western digital LIDAR 101.