Stereo vision systems can be used to generate real time disparity maps and extract 3D information from scenes without the use of super expensive and ugly sensors like LiDAR's. All that's needed is an image pair with sufficient quality. A drawback of using this method to obtain 3D world points of a scene is that there should be sufficient lighting. If the 3D world points are obtained, point clouds for the scene can be manufactured and visualised with the help of software such as open3D. We also explore a monocular depth estimation (which is the goal of finding where the 3d points lie in a world)

Our KITTI data is "perfect" for this task as it contains rectified stereo images and the readings from the IMU are taken to be very accurate.

We explore by comparing and contrasting the approach taken by multiple algorithms proposed in papers.

StereoBM

This method divides the images into block sizes dictated by the input parameter blockSize. It looks to exploit the temporal redundancy among the blocks between the two frames. This method also has multiple tunable parameters, but the disparity map is noisy and not smooth despite trying out various permutations for the hyperparameters of which there are only two: numDisparities and blockSize. Examples for the generated disparity are shown later, it is clear that StereoSGBM produces a better disparity map and thus a better point cloud.

StereoSGBM

Initially, we use a semi global block matching algorithm to find correspondences in the two images. This algorithm works by performing a line optimisation (along the epipolar line) in multiple directions and computing an average cost function between the intensities of each greyscale image but is highly memory inefficient. Doing this results in a disparity map which is fed to the next part of the code. There are multiple hyperparameters that need tuning in the stereoSGBM matcher which can be understood by going through the official documentation of the classical Semi Global Block Matching technique. The disparity generated for each dataset is governed by these hyperparameters. The hyper parameters are tuned such that they produce a smooth disparity for our stereo pair. These parameters dictate the feature matching done along the epipolar line.

In order to reproject the image points into the 3D space, we need something called a baseline matrix. This turns points 180 degrees around the x-axis such that the y-axis looks up and uses the camera intrinsics to find the right depth of a pixel, given the focal length, disparity and the baseline, via the formula:

$$\implies z = \frac{fb}{d}, \quad d \triangleq u_L - u_R$$
$$where \;\; d \rightarrow disparity/parallax$$

We reproject points into 3D (using the method of triangulation) in the coordinate frame used by opencv. This alone is enough to generate the individual dense point-clouds for each stereo-pair.

We now have the coordinates of the points in three dimensions. We extract the colours from the image and bring the color space from BGR (which is how opencv reads images) back to RGB.

With the 3D points and their corresponding colours, we can generate a point cloud for a given stereo pair.

PSMNet

This method aims to extract global context information via pyramid pooling, as with most deep learning methods in stereo vision it formulates the problem of depth estimation from a stereo pair as a supervised learning task.

While CNNs outperform classical approaches in terms of speed and accuracy in finding correspondances, it is still difficult to find accurate corresponding points in inherently ill-posed regions such as occlusion areas, repeated patterns ,textureless regions, and reflective surfaces. Solely applying the intensity-consistency constraint between different view-points is generally insufficient for accurate correspondence estimation in such ill-posed regions. It is useless in textureless regions. Therefore, regional support from global context information must be incorporated into stereo matching and that's where spatial pyramidal networks as well as dilated convolution comes into the picture.

This is an end-to-end framework which requires no post processing (unlike other methods, e.g. SfMLearner).

The architecture is that of a stacked hourglass 3D CNN, this means that there are layers of down sampling followed by up sampling in order to be able to better use global features. The information from the top-down and bottom-up architecture is integrated via skip connections.

The left and right feature maps are concatenated into a cost volume. This is then fed into a 3D CNN for regularisation . The disparity is generated on applying regression to optimise over this feature map. The predicted disparity is calculated as the sum of each disparity weighted by its probability from each of the outputs of the three stacked hourglass modules.

The spatial pooling module learns dependencies between features (car: tires, bonnet, windows. Usually, texture less regions for which it is hard to calculate the disparity). The final depth is calculated using the above formula.

SfMLearner

This is an unsupervised learning framework for the task of monocular depth and camera motion estimation from an unstructured video sequence. The method uses single-view depth and multi-view pose networks, with a loss based on warping nearby views to the target using the computed depth and pose. The networks are thus coupled by the loss during training, but can be applied independently at test time

This paper explains a novel concept of "view synthesis" that is given a view and the transformation to the the next camera pose, a target view is synthesized given a per-pixel depth in that image. This pose is predicted as a part of the learning framework. When applied to monocular videos the above view synthesis formulation implicitly assumes 1) the scene is static without moving objects; 2) there is no

occlusion/disocclusion be-tween the target view and the source views; 3) the surface is Lambertian so that the photo-consistency error is meaningful.

For single-view depth prediction, the DispNet architecture proposed is adopted, it is mainly based on an encoder-decoder design with skip connections and multi-scale side predictions. The input to the pose estimation network is the target view concatenate with all the source views (along the color channels), and the outputs are the relative poses between the target view and each of the source views.

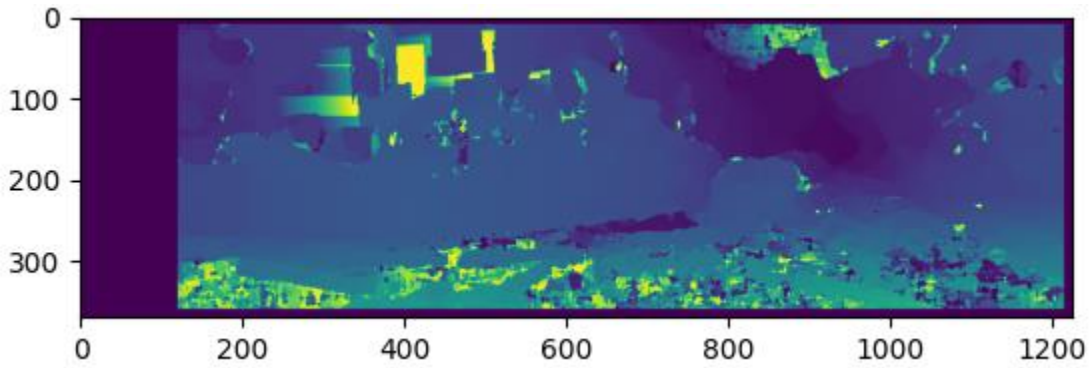
the depth CNN has low confidence in predicting thin structures well, and tends to mask them as unexplainable, this is visible in our point cloud where the pole is. The issue has been opened in the pytorch implementation https://github.com/ClementPinard/SfmLearner-Pytorch/issues/102.

Concatenating multiple point clouds:

What we aim to do is to register all the point clouds according to the given odometry data obtained from the KITTI IMU. To do this, we must take care that the points of the point cloud are in the same frame as the IMU. Then using matrix multiplication, we get out points into the world frame. A sanity check can be done here to check if the registration is happening properly or not. Instead of passing the points from the image, one could pass a sparse NumPy array which has a small block of ones and every other entry to be zero. This array can then be registered using the given odometry. If a curve appears in the final output then the registration is proceeding properly.

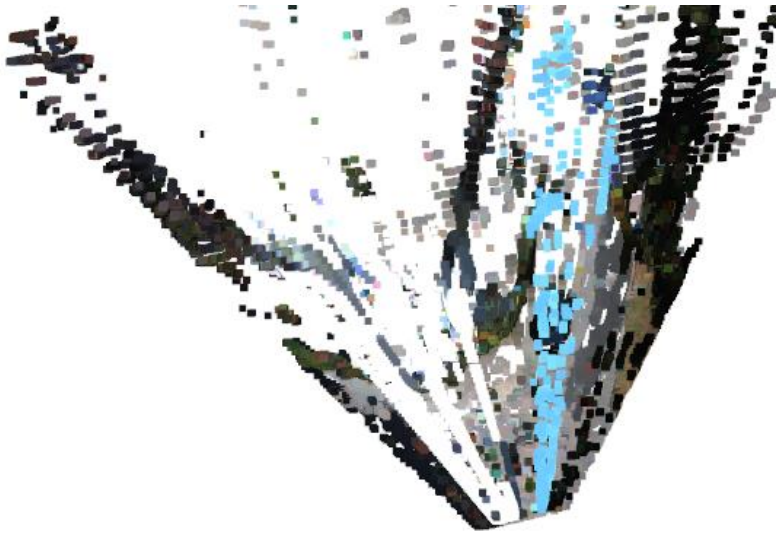
Comparing results:

The disparity map for an optimal set of hyper parameters for this data set (numDisparities = 112, blockSize = 21) for the StereoBM_create() method of opencv.

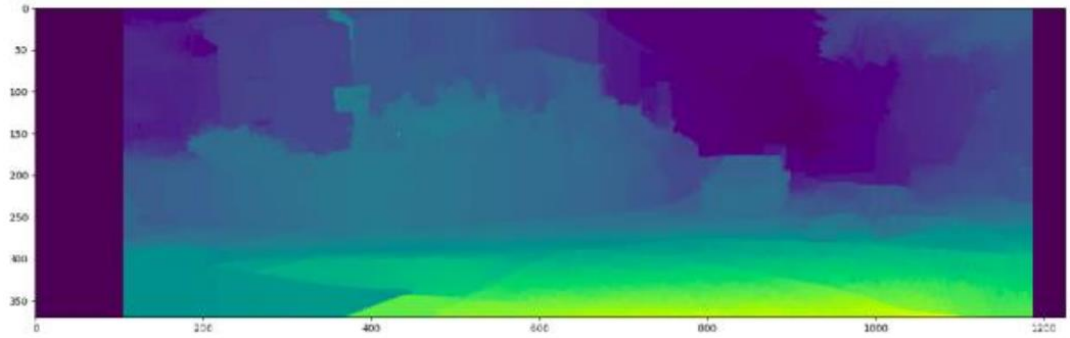


Clearly, this is very noisy and is not smooth, the same is reflected in the below point cloud.

While the point cloud may seem rather clean, a top view presents to us the real picture and reveals the noise in the generated disparity thus leading to a not-so-smooth image despite applying a Weighted-Least-Squares filter.



We use only StereoSGBM in our further comparisons as it generates a much cleaner and smoother disparity map:



We now compare StereoSGBM, PSMNet and SfMLearner on image numbers: 460, 465 and 480.

The classical StereoSGBM gives us a single dense point cloud which most closely resembles the ground truth.

The images are in the order mentioned above.

The registered point cloud looks like

We visually inspect and compare point clouds below. From this view, it is clear that registration is happening properly and that the colours are being accurately reproduced. There is some noise in the image and that noise is attributed to noise in disparity generation. There are no observed ill-effects like warping / swirling or elongation of edges.

Above is the top view for the SGBM matcher. We can see that it doesn't get smeared points as conspicuously, we're able to observe a cluster of blue points at the bottom left of the point cloud. On inspection and analysis of each individual frame as well as a concatenation of a subset of frames, we see that it is the result of a few erroneous disparities accumulated over the course of all 21 frames. As mentioned in the report, this is because of the implementation of stereoSGBM which doesn't explicitly handle such "textureless" surfaces which the sky in the frames is. Hence there are a few points for which the disparity isn't correct (we had looked at multiple parameters for the SGBM matcher and found this set to be ideal in that it minimised the number of "wrong" disparities). There is also an evident "bending" of the car in the middle, this is because of the stereo setup turning around the corner, the views presented to the camera first have the car being at a distance (where the view is more side-on) to a view where the car is close but the view is more from the rear. When viewed from the right angle though, this "bending" (it isn't the smearing that is observed in the stereoBM method) isn't noticeable at all.

We now look at the resulting point cloud from the disparity map generated by PSMNet

Looking at the birds-eye view of the generated point cloud, we can see that for certain sharp edges, the matching hasn't been done properly thus resulting in a stretched object, this is effect is visible when observing the white car on the left of the point cloud, it's side has been "stretched". The image on the right presents a clearer view as it does not include all the points in the point cloud and we can see the elongation of the sharp edges of the car clearly. We can also note the seemingly straight-line structure of the point cloud, in that the scene seems to be rendered a plane at a time, unlike the case for the classical StereoSGBM. The blue car in the first image below though doesn't show its side profile to be distorted as we have multiple views from that angle, b

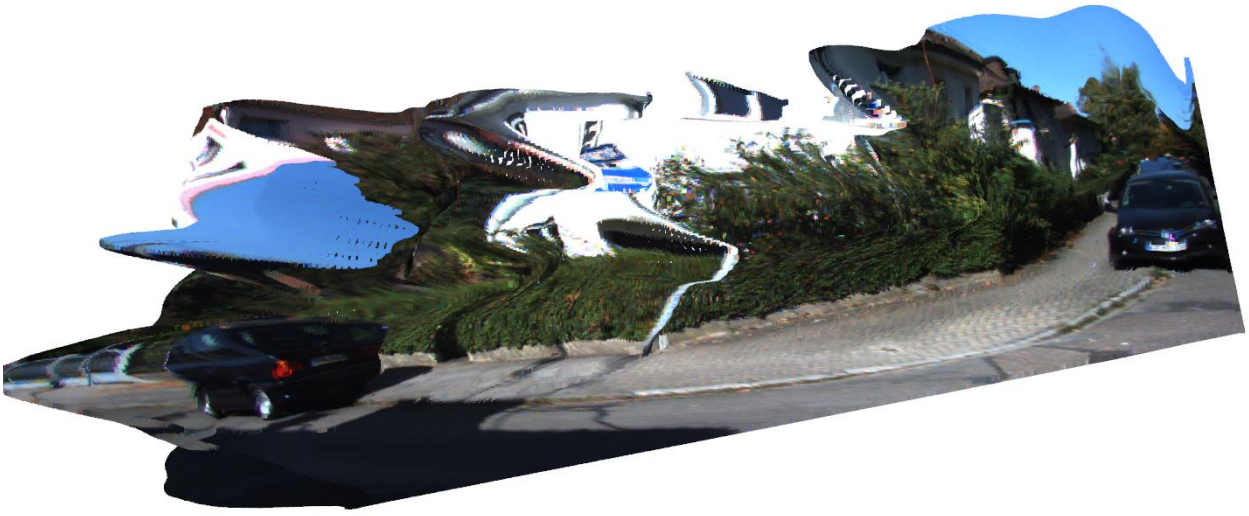
We now observe the registered point cloud from the PSMNet disparities.

In the first of the above images, we see that sharp edges as well as large textureless regions like the sky and the wall of the house aren't registered properly, this is surprising as the PSMNets claim-to-fame is that by using global information it is able to get an accurate disparity for these textureless regions. A possibility for the above observation is that the error because of the textureless regions had accumulated over the frames (as elongation of such a degree wasn't observed in the single disparity

point cloud). In the second image, we note that the blue car on the right hasn't been registered properly, again, this is because the right side of the car (its right side) is occluded from the stereo set up and thus we cannot know for sure where it lies in the 3D world.

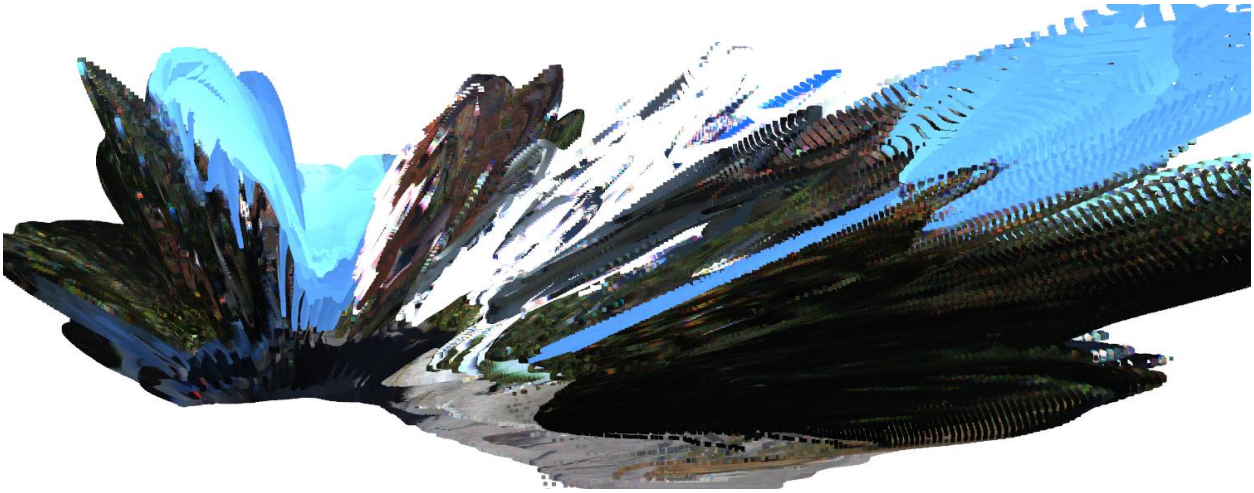
We now take the three views from SfMLearner.

We also present another view of the second scene where it is apparent that the pole hasn't been registered properly.



The swirling here is caused by the fact that our network has only seen this view once, it doesn't know the preceding and succeeding view to this frame, thus it doesn't have a ground truth to optimise over. This scene was rendered using the pretrained network with its pretrained weights.

We do not present the registered point cloud for the SfMLearner as it is rather rubbish in terms of a registered point cloud but looking at it from another angle, with all of its curves it does look like a piece of art (which monocular depth estimation is!)



Conclusion:

We see that depth estimation using a stereo setup solely lies on how well the disparity of a point can be estimated. Problems faced in this field include but aren't limited to: Occluded regions, low light (low visibility of features, edges), smooth / textureless surfaces and noise. Modern Deep-learning based approaches use CNNs in order to better estimate the disparity between stereo images and then regress on the formulated cost function. These methods are proven to be faster than classical block matching algorithms, this speed is important in real life applications such as autonomous cars where depth has to be constantly estimated without the help of external LiDARs. In the modern day, the rise of monocular SLAM (which involves depth estimation) and monocular depth estimation has led to research into such areas. The generated point clouds help us visually understand the process followed by each of the papers.