***Abstract -* Accurate localization of a vehicle is a fundamental challenge and one of the most important tasks of mobile robots. This paper describes a stereo visual odometry algorithm for estimating frame-to-frame camera motion from successive stereo image pairs. In this work, we implemented stereo visual odometry using images obtained from KITTI Vision Benchmark Suite and presented obtained results.**

1. **Introduction**

For autonomous navigation, motion tracking, and obstacle detection and avoidance, a robot must maintain knowledge of its position over time. Vision-based odometry is a robust technique utilized for this purpose. It allows a vehicle to localize itself robustly by using only a stream of images captured by using only a stream of images captured by a camera attached to the vehicle. In [robotics](https://en.wikipedia.org/wiki/Robotics) and [computervision](https://en.wikipedia.org/wiki/Computer_vision), Visual Odometry is the process of determining the position and orientation of a robot by analyzing the associated camera images. The idea was first introduced for planetary rovers operating on Mars – Moravec in the early 1980s. Visual Odometry  is a technique used to localize a robot by using only a stream of images acquired from a single or multiple cameras attached to the robot. The images contain a sufficient amount of meaningful information (color, texture, shape, etc.) to estimate the movement of a camera in a static environment. Stereo visual odometry estimates the camera's egomotion using a pair of calibrated cameras. Stereo camera systems are inherently more stable than monocular ones because the stereo pair provides good triangulation of image features and resolves the scale ambiguity.

**Related Work**

In recent years many algorithms for visual odometry have been developed, which can roughly be devised into two categories, namely methods using monoscopic cameras or methods using stereo rigs. These approaches can be further separated into methods which either use feature matching between consecutive images or feature tracking over a sequence of images. If a calibrated multi-ocular camera setup is available, the 3-dimensional scene can be reconstructed via triangulation. Based on the point clouds of the static scene in two consecutive images, the iterated closest point (ICP) algorithm is often used for egomotion estimation as described in [7]. Monocular cameras mainly require tracking image features (e.g. corners) over a certain number of images. Using these feature tracks, also the scene structure can be computed using structure from motion [8]. In most cases, the multiocular algorithms yield better performances than monocular approaches [4]. Additionaly, if multi-camera approaches are used, the scale ambiguity present in the monocular case is eliminated [1]. Further approaches combine visual odometry with other sensors to increase the accuracy of the results and reduce drift, a problem inherent to all incremental positioning methods.

1. **Overview**

**2.1 Dataset used**

○ KITTI Vision Benchmark Suite : The KITTI Odometry dataset was used in our project. The dataset contains 21 sequences of stereo video sequences in greyscale.

○ Calibration Files: Camera and projection matrices are provided to the user with every video sequence in the form of a calibration file. Time stamps of every frame is also provided.

○ Poses files: The folder '[poses.txt](http://www.cvlibs.net/download.php?file=data_odometry_poses.zip)' contains the ground truth poses (trajectory) for the first 11 sequences.

○ The camera orientation is as follows and shown in Fig. 1 below:

* The X-axis is parallel to the ground and towards the right of the driver.
* The Y-axis is perpendicular to the ground and facing downwards.
* The Z-axis is is parallel to the ground and facing forward

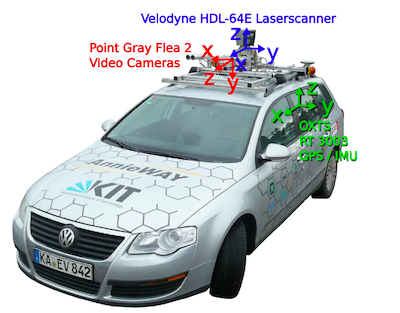


Fig. 1: KITTI dataset recording platform: VW Passat station wagon is equipped with four video cameras (two color and two grayscale cameras), a rotating 3D laser scanner and a combined GPS/IMU inertial navigation system. (Image taken from the KITTI dataset paper)

**2.2. Our Approach**

The basic algorithm is as follows: for a given pair of frames, (1) detect features in each frame (corner detection), (2) match features between frames (sum-of-absolute differences over local windows), (3) find the largest set of self of consistent matches (inliers), and (4) find the frame-to-frame motion that minimizes the re-projection error for features in the inlier set.

The inlier detection step (3) is the key distinguishing feature of the algorithm. The feature matching stage inevitably produces some incorrect correspondences, which, if left intact, will unfavorably bias the frame-to-frame motion estimate. A common solution to this problem is to use a robust estimator that can tolerate some number of false matches (e.g., RANSAC [4]).

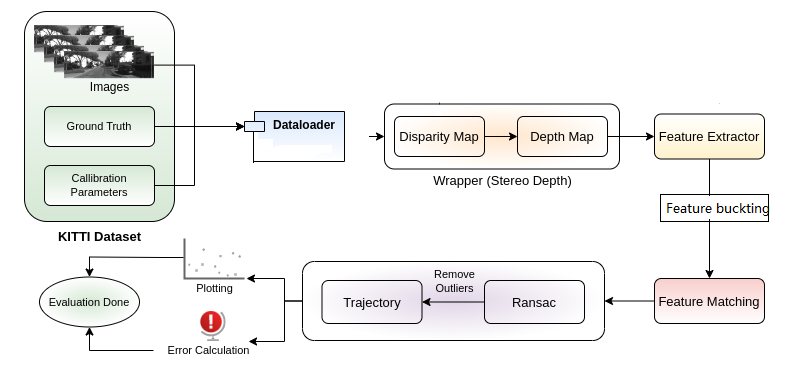


Fig. 2: Pipeline of Stereo Visual Odometry

**3. Detailed Methodology**

**3.1 Problem Formulation**

**Input** Our input consists of a stream of gray scale images obtained from a pair of cameras. In KITTI dataset the input images are already corrected for lens distortion and stereo rectified. Let the pair of images captured at time *t* and *t+1* be *(Il,t, Ir,t)* and *(Il,t+1, Ir,t+1 )* respectively. The intrinsic and extrinsic parameters of the cameras are obtained via any of the available stereo camera calibration algorithms or the dataset.

**Output** For every stereo image pair we receive after every time step we need to find the rotation matrix R and translation vector t, which together describes the motion of the vehicle between two consecutive frames.

**3.2 Disparity maps**

In case of a stereo camera the coordinate system of the camera is assumed to be situated midway between the two cameras. We will also be considering the left image as the reference image throughout the project. If we are considering a horizontal stereo camera, each matched keypoint will have the same y coordinate. Only the x coordinate of the pixels with the keypoints in left and right images will be different. The difference between these two x coordinates is known as the disparity.

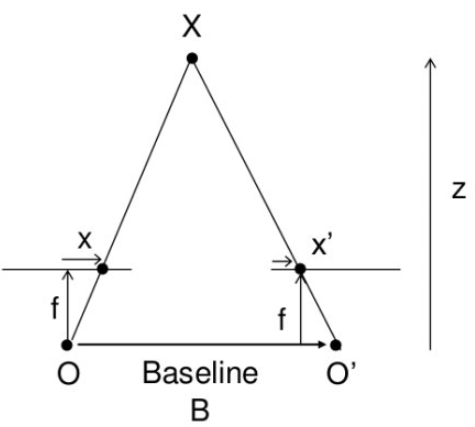
****

Fig. 3: Geometry for calculating the depth from disparity

To construct disparity map, we have used the [StereoSGBM](https://docs.opencv.org/3.4/d2/d85/classcv_1_1StereoSGBM.html) function from OpenCV which is an implementation of Hirschumuller Algorithm. A disparity map is depicted below:

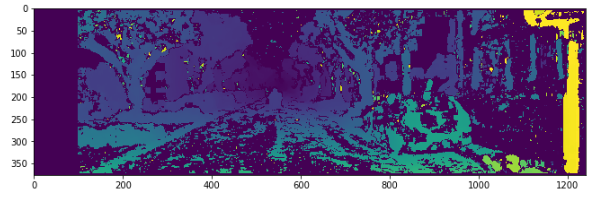


Fig. 4: Stereo disparity map of first sequence image

Using the disparity map, we get the depth of the scene using the formula:

https://latex.codecogs.com/gif.latex?L%3D%20%5Cfrac%7BBf%7D%7BZ%7D

**3.3 Feature detection**

**3.4 Feature buckting**

**3.7 Feature tracking**

**3.5 Feature Matching**

**3.6 Triangulation of feature points**

Whenever we detect key points in images, we can only find the position of that keypoint in the image. It is being assumed that the coordinate frame of the car is same as the coordinate frame of the camera and we are interested in finding the position of the car in real-world coordinate system. For this, we need to find the coordinates of the keypoints in the real world coordinate system. It is quite impossible or hard to get the accurate 3D world coordinates of the keypoints from a monocular camera or a single frame of the scene. Like the human eye, we need a pair of stereo images to accurately find the depth of the keypoints in the world.

Dataset provides left and right images of each instance. The keypoints detected in the left and right images are matched as described in the previous section. The best matches among all the matches are chosen so that we can we can find the corresponding 3D points. In order to find the 3D world coordinates, we need to have complete information about the camera coordinate system. Fig. is the orientation of the camera coordinate system. The z- coordinates comes out of the image plane. The complete method behind this computation is that we first find the disparity of the whole image.

* 1. **Estimating the transformation matrix**

1. **Stereo Visual Odometry Results**
2. **Experiments,**
3. **Findings,**
4. **Conclusion**

[1] M. Agrawal and K. Konolige, “Real-time localization in outdoor environments using stereo vision and inexpensive gps,” in Proceedings of the 18th International Conference on Pattern Recognition, 2006, pp. 1063 – 10 068.

[4] H. Badino, “A robust approach for ego-motion estimation using a mobile stereo platform,” in First International Workshop on Complex Motion, 2004, pp. 198 – 208.

[7] A. Milella and R. Siegwart, “Stereo-based ego-motion estimation using pixel-tracking and iterative closest point,” in Proceedings of the Fourth IEEE International Conference on Computer Vision Systems, 2006.

[8] D. Nister, O. Naroditsky, and J. Bergen, “Visual odometry,” in ´ IEEE Computer Society Conference Computer Vision and Pattern Recognition, vol. 1, 2004, pp. 652 – 659.