

Pothole/bump Detection

Using ZED cameras

**University of Patras,
Visualization and Virtual Reality Group (VVR)**

1. Introduction

Road surface degradation such as potholes and bumps poses risks to vehicles and pedestrians, reduces ride comfort, and increases maintenance costs. Early detection of these defects is essential for timely intervention by municipal authorities.

This project explores the use of stereo camera data (ZED camera recordings) to detect and visualize anomalies (potholes, bumps, and uneven surfaces) in road scenes.

2. Purpose and Objectives

The main objectives of the project were:

- To process raw stereo vision data into reliable point clouds.
- To identify surface irregularities through geometric analysis.
- To classify anomalies as potholes, bumps, or flat regions.
- To visualize the detection results for interpretation.

3. Data

The dataset used is: *Villanova potholes dataset* consisted of recorded SVO files captured with a ZED camera. The ZED provides RGB and depth data, which was used to construct 3D point clouds of road surfaces. The frames analyzed span a variety of road segments, some containing potholes, cracks, or bumps.



The videos include several non-relevant elements such as trees, a pedestrian walkway, and other features outside the roadway. The objective was Pothole/bump detection, so attention was directed to specific parts of the videos, where a pothole/bump is clearly visible.

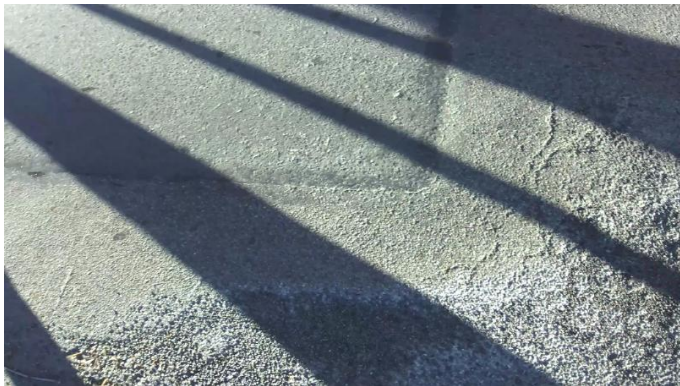
Some examples of frames that are a good input are:

HD2K_SN39967967_08-46-22.svo2 , pothole, frame:95

HD2K_SN39967967_08-43-29.svo2 , bump, frame:261

HD2K_SN39967967_08-46-22.svo2 , small bump, frame:200

HD2K_SN39967967_08-51-58.svo2 , flat road, frame:269



4. Implementation

The pipeline was implemented in Python using: ZED SDK (depth acquisition), Open3D (point cloud processing), SciPy KDTree (neighbor search), NumPy (statistical processing).

The main workflow:

1. Read frame → Extract 3D point cloud.
2. Preprocess (filter + downsample + smooth).
3. Fit plane using RANSAC.
4. Compute distances → detect anomalies.
5. Export annotated point cloud.

Methodology

4.1 Point Cloud Acquisition

- The **ZED SDK** was used to read .svo2 recordings.
- Depth mode: **ULTRA**, chosen for higher accuracy at the expense of speed.
- The acquired depth data was converted into **Open3D point clouds** for downstream processing.

4.2 Preprocessing

1. **Invalid point removal:** NaN or infinite values were discarded.
2. **Downsampling:** A voxel grid filter (voxel_size=0.005) reduced redundancy while preserving surface geometry.
3. **Outlier removal:** A radius-based filter (nb_points=20, radius=0.03) removed sparse or isolated points caused by depth noise.
4. **Smoothing:** A custom KNN-based smoothing method was applied, where each point was iteratively averaged with its 10 nearest neighbors over 5 iterations. Without this step, the point cloud appeared overly noisy and irregular.



Example of the reconstructed pcd without smoothing.

4.3 Plane Segmentation

- A **RANSAC-based plane fitting** algorithm was used to estimate the dominant road surface.
- The fitted plane has the form: $ax+by+cz+d=0$
- After estimation, the plane normal vector was normalized, and the signed distances of all points to this plane were computed.
- To establish a consistent reference, distances were centered around zero, effectively aligning the mean road surface to the baseline.

4.4 Anomaly Detection

Instead of using a fixed height threshold (which would not generalize well across varying road conditions), a **statistical approach** was employed:

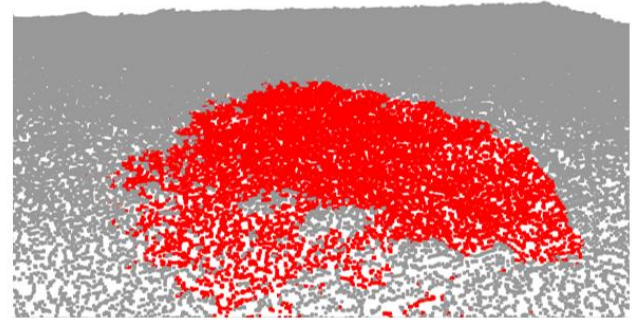
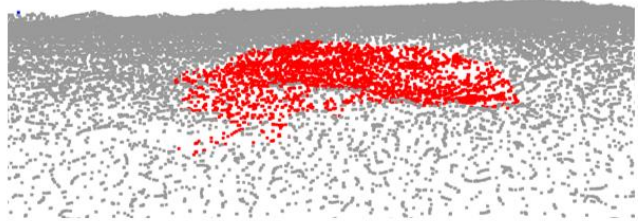
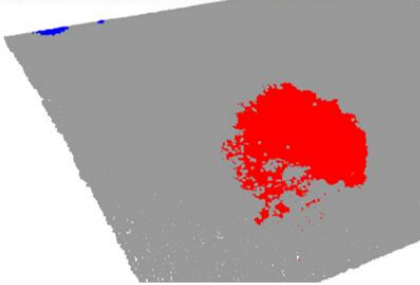
- **Percentile-based thresholds:**
 - Bottom ~7% of distances → candidate **pothole points**.
 - Top ~7% of distances → candidate **bump points**.
- **Verification step:**
 - Percentiles alone may highlight outliers even when no anomaly exists.
 - To reduce false positives, candidate regions were validated by checking whether their mean distance exceeded a multiple of the global standard deviation (experimentally set to $\sim 2.2\times$ for potholes and $\sim 2.5\times$ for bumps).
- **Flat-surface condition:**
 - If the standard deviation of distances was below **0.01**, the frame was classified as “flat” (i.e., no significant anomaly).
- **Cluster cleaning:**
 - Candidate pothole or bump points underwent a secondary outlier removal ($\text{nb_points}=70$, $\text{radius}=0.025$) to eliminate spurious detections.

4.5 Visualization

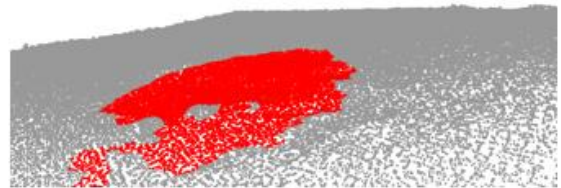
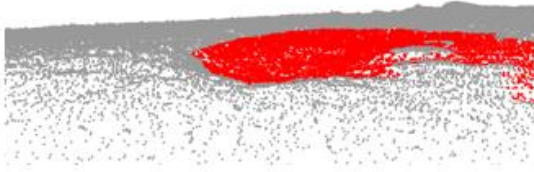
- Classified points were color-coded for interpretability:
 - **Red** = potholes
 - **Blue** = bumps
 - **Gray** = background road
- The results were exported as .ply files, enabling inspection in 3D viewers and visual validation of anomaly detection.

5. Results

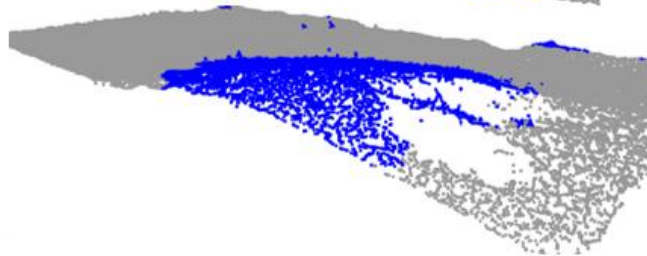
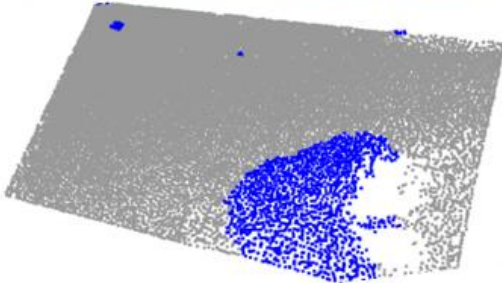
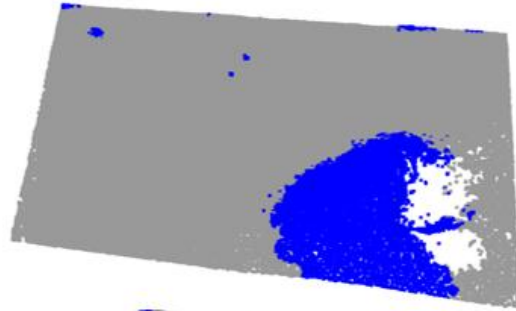
i) The pothole



Results after fine tuning some parameters:

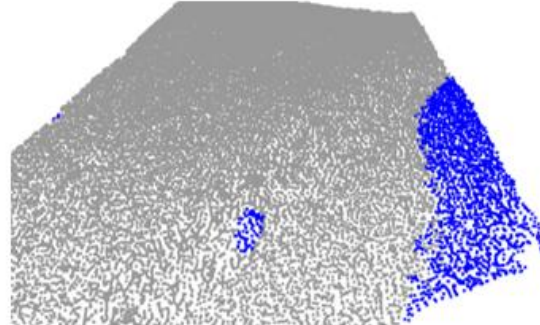


ii) The bump

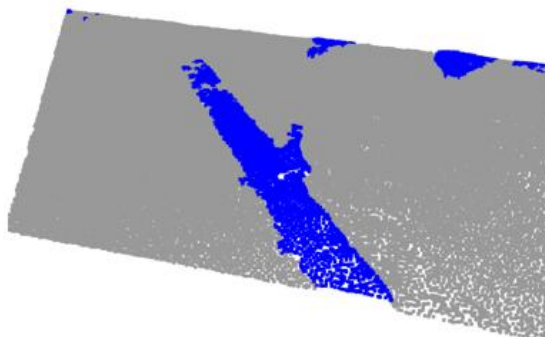


The 2 holes are explained because of the 2 completely white areas in the image (can't find the position of those pixels in the other image to determine depth)

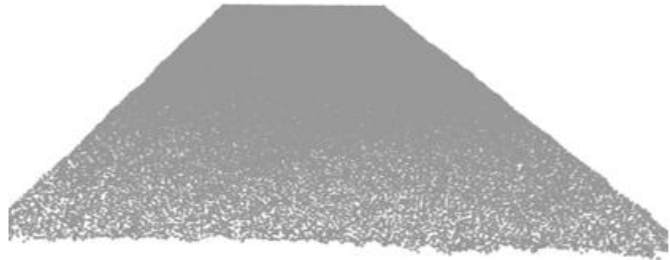
iii) Small roughness



iv) Poorly maintained curb



v) When there is nothing to detect



To find the appropriate values of some parameters I observed the statistics of each case:

```
Processing Villanova_potholes_dataset/HD2K_SN39967967_08-46-22.svo2 at frame 95 (
smoothing..
plane fitting..
Min Distance: -0.06763440818688604
Max Distance: 0.07223026357959705
Avg Distance: -1.867971856697242e-17
Threshold for pothole (distance): -0.035169854815915794
Threshold for bump (distance): 0.041806296742941416
Difference (avg - pothole threshold): 0.03516985481591577
Difference (avg - bump threshold): -0.04180629674294144
std : 0.01851287384254157
status: both
Drawing frame 95
Saved colored point cloud to: processed_frame95_colored.ply
Press Enter in this terminal to close the viewer.
```

```
Processing Villanova_potholes_dataset/HD2K_SN39967967_08-46-22.svo2 at frame 200
smoothing..
plane fitting..
Min Distance: -0.050355478695544686
Max Distance: 0.08241449582910443
Avg Distance: 1.1868687426506147e-16
Threshold for pothole (distance): -0.018337412538259004
Threshold for bump (distance): 0.03263715785093346
Difference (avg - pothole threshold): 0.018337412538259122
Difference (avg - bump threshold): -0.032637157850933345
std : 0.013813196628548435
status: bump
Drawing frame 200
Saved colored point cloud to: processed_frame200_colored.ply
Press Enter in this terminal to close the viewer.
```

```
Processing Villanova_potholes_dataset/HD2K_SN39967967_08-43-29.svo2 at frame 261
smoothing..
plane fitting..
Min Distance: -0.058493340265393545
Max Distance: 0.07373281185748981
Avg Distance: -3.3668190394770435e-16
Threshold for pothole (distance): -0.021896035863844164
Threshold for bump (distance): 0.0461428660592595
Difference (avg - pothole threshold): 0.021896035863843828
Difference (avg - bump threshold): -0.04614286605925984
std : 0.017577170400290172
status: bump
Drawing frame 261
Saved colored point cloud to: processed_frame261_colored.ply
Press Enter in this terminal to close the viewer.
```

```
Processing Villanova_potholes_dataset/HD2K_SN39967967_08-51-58.svo2 at frame 269
smoothing..
plane fitting..
Min Distance: -0.020709929806987493
Max Distance: 0.027943273140182523
Avg Distance: -6.300155986075622e-17
Threshold for pothole (distance): -0.0075646861094528
Threshold for bump (distance): 0.006895846390029271
Difference (avg - pothole threshold): 0.007564686109452737
Difference (avg - bump threshold): -0.006895846390029334
std : 0.004076096410424502
status: pothole
Drawing frame 269
Saved colored point cloud to: processed_frame269_colored.ply
Press Enter in this terminal to close the viewer.
```

The necessity of cluster cleaning:

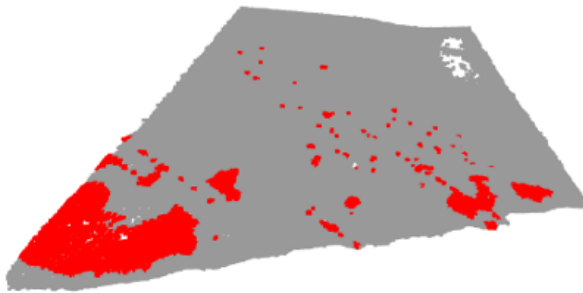
Cluster cleaning was necessary to get rid of noisy, isolated points and ensure that only coherent pothole/bump regions were retained.

Below there is an example of detection without cluster cleaning.

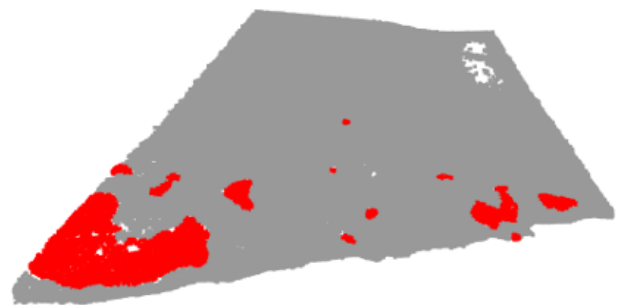


As seen above, there are many small regions that are marked but they are not really a pothole.

The solution: the algorithm checks that these points form a coherent cluster, not just random noise, by performing a second outlier removal on the marked points themselves (nb_points=70, radius=0.025).

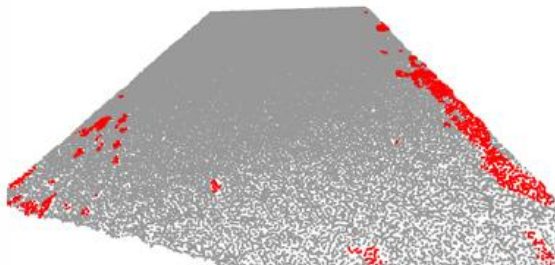


Before outlier removal

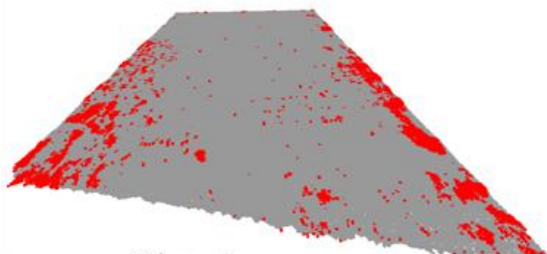


final result after outlier removal

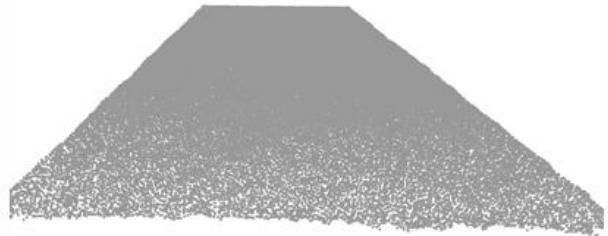
Importance of the std check and outlier removal:



After outlier removal



1st result



After outlier removal and std check

The application of the proposed method to a video sequence is presented below:

