Visual analysis of moving vehicles

Niccolò Balestrieri[1], **Andrea Bertogalli**[2], **Nicolò Tombini**[3]

[1]Politecnico di Milano, Italy

niccolo.balestrieri@mail.polimi.it, andrea.bertogalli@mail.polimi.it nicolo.tombini@mail.polimi.it

1 Introduction

The project concerns the task of **Visual analysis of moving vehicles**: in particular model-based car localization and road reconstruction: specifically, identifying the visible wheels of a moving car (potentially using deep learning as an aid); extracting the rims due to their higher contrast; using information about their dimensions (and possibly about the wheelbase) to follow the road profile through their envelope (especially for non-straight and non-planar roads); collecting 3D points on the road and interpolating the local surface.
All of this is done using a **single calibrated static camera**.

1.1 Task's motivation

The advantages of this project are many. Firstly, using a single static camera reduces the cost and complexity of the system, making it more accessible and easier to deploy. Secondly, the use of deep learning techniques to identify and track the wheels enhances the accuracy of the localization process, even in challenging conditions.
Moreover, the collection of 3D data and the interpolation of the road surface can be valuable for infrastructure maintenance and urban planning, allowing for the early detection of road damage and the efficient allocation of repair resources.

2 State of the art

In the literature, this type of problem is addressed in two main ways: Deep Learning techniques and classical Computer Vision techniques. Regarding the latter, the task of 3D road reconstruction is approached with three main techniques:

- **Stereo Vision**: this method uses two or more cameras to capture images from slightly different perspectives. By analyzing the differences between these images, it is possible to reconstruct the depth and 3D structure of the road surface.

- **Structure from Motion**: this technique involves capturing a sequence of images from a moving camera. By tracking the movement of features across these images, the 3D structure of the road can be reconstructed. SfM does not require calibrated cameras but relies on the motion and overlapping views to infer depth.

- **Multi-view Photogrammetry**: this approach uses images taken from multiple viewpoints to create a detailed 3D model of the road surface. It typically involves precise camera calibration and advanced image matching algorithms to ensure accuracy and detail in the reconstructed model.

3 Our proposed solution

The developed pipeline aims to be the more genereal and flexible as possible archiving qualitatively good results. Our pipeline is composed by three stages:

1. **A Region Proposal Network (RPN)**: which is basically an object detection network trained to detect the wheels of the cars.

2. **Ellipse detection**: An ellipse detection algorithm have been designed to detect the ellipses corresponding to the visible wheels inside the proposed regions.

3. The last step is the estimation of the **3D point cloud** and the subsequent surface interpolation.

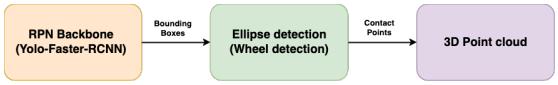


Figure 1. Our proposed generic pipeline.

3.1 Camera

For the necessary video acquisitions, an iPhone 14 Pro Max camera was used, mounted on a photographic tripod. The iPhone 14 Pro Max was selected due to its advanced camera system, which offers exceptional performance in a variety of shooting conditions. This choice was made because it met the project requirements, necessitating the use of a single non-stereo camera.

Figure 2. Camera

With this camera, several videos have been recorded under the conditions shown in the following sections, with a resolution of 1920x1080 at 60 FPS.

3.2 Camera calibration procedure

By calibrating the camera, we obtain the intrinsic calibration matrix which contains the camera's parameters and it is also possible to retrieve the lens distorsion coefficient. The camera was calibrated following the well known **Zhang method**. The Zhang method is a widely used technique for camera calibration: this method requires multiple images of a planar calibration pattern (e.g., a checkerboard) taken from different orientations. The Zhang method involves the following steps:

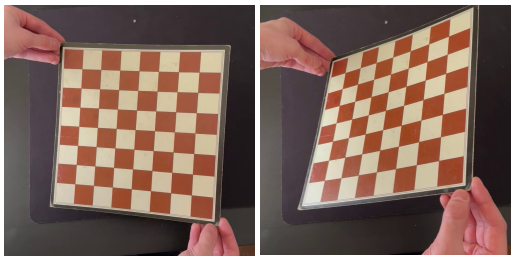
1. **Image Capture**: Capture multiple images of a planar calibration pattern from different angles and positions.

2. **Feature Extraction**: Detect feature points (in this case, corners of the checkerboard) in the images.

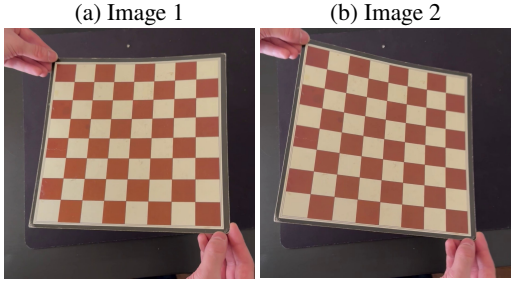
3. **Homography Estimation**: For each image, compute the homography that maps the points on the calibration pattern to the image points. This can be done using the Direct Linear Transformation (DLT) algorithm.

4. **Intrinsic Parameter Estimation**: Use the homographies to solve for the intrinsic camera parameters.

In this project, a chessboard of known dimensions (7x7) was filmed using the same camera used for filming the cars. Frames were extracted from this video and provided as input to the calibration algorithm.
Below there is an example of a simple frame that was given as input:



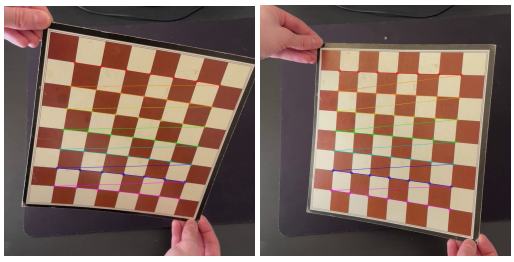
(a) Image 1 (b) Image 2



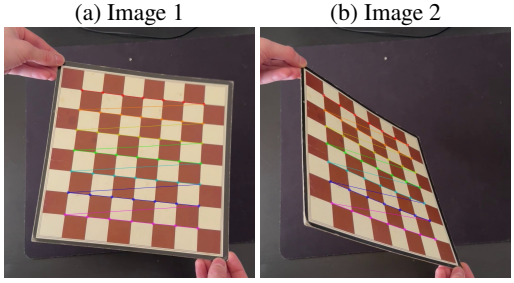
(c) Image 2 (d) Image 2

Figure 3. Four frames as input

And as an output:



(a) Image 1 (b) Image 2



(c) Image 2 (d) Image 2

Figure 4. Four frames as output

3.3 Data

Data was collected by recording videos of cars on various roads, assuming that the rims would not be parallel to the image plane and would not be filmed from above. Specifically, three types of recordings were conducted:

- Cars making a large curve:

Figure 5. Cars making a large curve

- Cars making a sharp and distant curve:

Figure 6. Cars making a sharp and distant curve

- Cars on a straight road:

Figure 7. Cars on a straight road

- Cars going uphill:

Figure 8. Cars going uphill

4 Implementation description

4.1 Network architectures

4.1.1 Introduction

The following section elucidates the methodology and network architectures employed for extracting bounding boxes from images. This process involves two network used **independently**, as two different approaches:

- a pre-trained deep learning model (**ResNet50**), particularly within the TensorFlow framework, for object detection.
- a lighting and Rotation Invariant Real-time Vehicle Wheel Detector based on **YOLOv5** within PyTorch framework.

The extracted bounding boxes are subsequently processed to identify specific features, such as wheels, within the images.

4.1.2 ResNet50

An object detection model pre-trained on the COCO 2017 dataset was selected, specifically the SSD-ResNet50 V1 FPN (derived from ResNet K. He et. al, 2016 [1]) from TensorFlow 2. For the region proposal, a Momentum optimizer with a learning rate of 0.01 was used, with a batch size of 4.

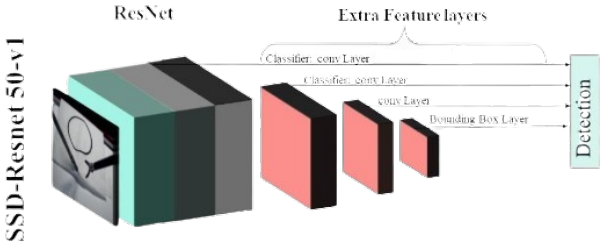


Figure 9. SSD-ResNet50 V1 FPN Architecture, A. Rashidi et al. [2]

As shown in Figure 9, the Feature Pyramid Network (FPN) generates multi-level features that are input to the SSD-ResNet50 architecture. The FPN acts as an extractor and provides the extracted feature maps to the object detector.

This network have been fine-tuned by feeding the network with annotated images of cars. The fine-tuning is done by unfreezing all the network layers, in this way the weights of the network are adjusted according to the wheels detection task.

4.1.3 YOLOv5

YOLO is one of the best architecture for object detection providing sate of the art performance with its focus on real-time detection and classification.
The model consists of two parts, the Backbone and Head. Backbone is responsible for extracting low level features

using convolutional layers in combination with Spatial Pixel Pair Features (SPPF) layer. The Head is responsible for extracting the high level feature maps and performing the detections, applying anchor boxes on the feature maps to generate the final output vector of classes probabilities and bounding boxes.

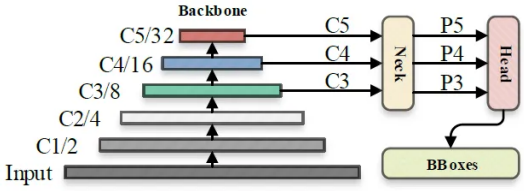


Figure 10. General YOLOv5 architecture

The medium size has been chosen for the wheel detector to provide a good combination of accuracy and performance.

Model	Size (Pixels)	mAP @0.5:0.95	mAP @0.5	Time CPU b1 (ms)	Time V100 b1 (ms)	Time V100 b32 (ms)	Params (M)	FLOPS @640 (B)
YOLOv5n	640	28.0	45.7	45	6.6	0.6	1.9	4.5
YOLOv5s	640	37.4	56.8	98	6.4	0.9	7.2	16.5
YOLOv5m	640	45.4	64.1	224	8.2	1.7	21.2	49.0
YOLOv5l	640	49.0	67.3	430	10.1	2.7	46.5	109.1
YOLOv5x	640	50.7	68.9	766	12.1	4.8	86.7	205.7

Figure 11. YOLOv5 version benchmark

In this particular use-case it was selected a wheel detector (from M Shenoda, 2023 [3]), based on YOLOv5 that was pre-trained on a set of cars captured to cover various camera orientation as well as slight shadows and sun light conditions.

Bounding Box Detection

Bounding boxes of the wheels are detected from the frames of the videos.
Initially, the videos are divided into frames, and each frame is used as input for the network. Only for ResNet50, it is important to note that the image size is crucial; in this case, the input dimensions of the neural network are fixed at 640x640 pixels, so the images are resized accordingly.

4.2 Wheels detection

For what concerns the wheels detection part the bounding box obtained by the RPN is not sufficient, in fact the exact shape of the wheel is needed. To archive this we designed an algorithm which selects the most probable ellipse proposed by a classic ellipses detection algorithm. After the ellipse detection, it was obtained a discretized set of image points representing each of the wheels perimeter and from this it was selected the contact point $C = \begin{bmatrix} u & v & 1 \end{bmatrix}^T$ according to the following equation:

$$C = \sigma_k \quad \text{where} \quad k = argmax_j(\sigma_{i,j}) \tag{1}$$

where N is the number of discretized points and σ is the set of discretized points.

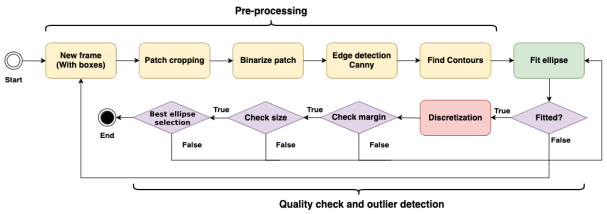


Figure 12. The proposed wheel detection method.

As can be seen from figure 12 the ellipses detection algorithm is composed by various steps, in particular the first step is a pre-processing stage where the patch (corresponding to the bounding box of a wheel) is first binarized to enhance the accuracy of the Canny edge detection algorithm and consequently of the contour detection algorithm. After this ellipses are fitted using Least Squares A. W. Fitzgibbon et al. 1999 [4], and then filtered according to some euristics on the size and the relative position in the bounding box. In particular the ellipses are filtered by checking the size according to this inequality $\begin{cases} S_{maj} > \frac{w}{2} \\ S_{min} > \frac{h}{2} \end{cases}$ where S_k are the major and minor semiaxes of the ellipses and h, w are respectively the height and the width of the bounding box. Another euristic used is that the detected ellipse must be inscribed in the bounding box $\sigma \subseteq B$. The last step for the selection of the best ellipse is to perform a ranking of ellipses mainly based on two possible criterions, the first criterion that can be used is to rank ellipses according to the euclidean distance between the center of the ellipse and the center of the bounding box:

$$E_{best} = argmin(\|\mathbf{e}_c - \mathbf{b}_c\|) \tag{2}$$

$$= argmin(\sqrt{(x_e - x_b)^2 + (y_e - y_b)^2}) \tag{3}$$

The other criterion that can be used is pattern matching, which consist in ranking the ellipses according to the similarity to a binarized wheel pattern, in particular the scoring function used to score the ellipses according to the template is the **normalized correlation score** (`cv2.TM_CCOEFF_NORMED`) defined as:

$$R(x,y) = \frac{\sum_{i,j}\left(T_{i,j} - \overline{T}\right)\left(I_{x+i,y+j} - \overline{I_{x,y}}\right)}{\sqrt{\sum_{i,j}\left(T_{i,j} - \overline{T}\right)^2 \sum_{i,j}\left(I_{x+i,y+j} - \overline{I_{x,y}}\right)^2}} \tag{4}$$

Where $T_{i,j}$ is the value of the template pixel at position (i, j), $\overline{T}$ is the mean of the template pixel values, $I_{x+i,y+j}$ value of the image pixel at position $(x + i, y + j)$ and $\overline{I_{x,y}}$ mean of the image pixel values in the current region.

Figure 13. The wheel template used for template matching

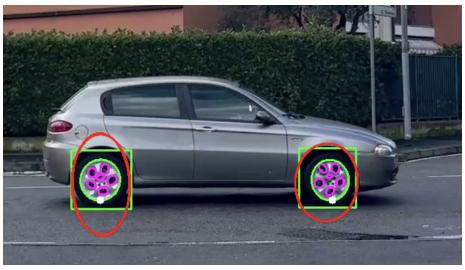


Figure 14. In this picture it can be noticed the output of the wheel detection pipeline, in red the ellipses discarded from the inscription constraint, while in pink the ellipses discarded by the size constraint, in green the best fitted ellipse and finally, in white, the estimated contact point.

4.3 3D Reconstruction

The last stage of the proposed pipeline is the reconstruction of the 3D point cloud of the road surface, this task is very challenging from a single static monocular camera. In fact the 3D reconstruction is mainly addressed with a stereo setting or with SLAM. Despite the difficulty of the task it was managed to propose two approaches which reached quite good result: A deep learning based approach and a geometric approach.

4.3.1 Deep learning depth estimation approach

For what concerns the Deep Learning based approach a monocular depth estimation model was used, this kind of networks are able to infer a depth-map from a single image. Given a depth map it could be extract the Z-Axis coordinate of an image point in the following way:

$$z_p = D_{x,y} \tag{5}$$

where x, y are the coordinates of a contact point in the image.

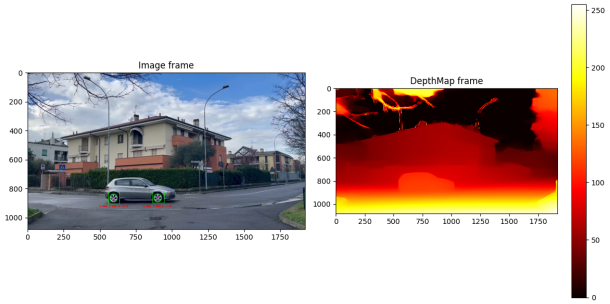


Figure 15. A n example of a monocular depth-map estimation with Deep Learning.

The model used for the depth estimation is DPT (Dense Prediction Transformer) R. Ranftl et al. (2021), [5] which is a S.O.T.A model based on ViT (Vision Transformer) A. Dosovitskiy et al. (2020) [6]. Dense vision transformers are introduced as an architecture that utilizes vision transformers instead of convolutional networks as a backbone for dense prediction tasks. Tokens are assembled from various stages of the vision transformer into image-like representations at multiple resolutions, which are progressively combined into full-resolution predictions using a convolutional decoder. The transformer backbone processes representations at a constant, relatively high resolution and maintains a global receptive field at every stage. These properties enable the dense vision transformer to deliver finer-grained and more globally coherent predictions compared to fully-convolutional networks.

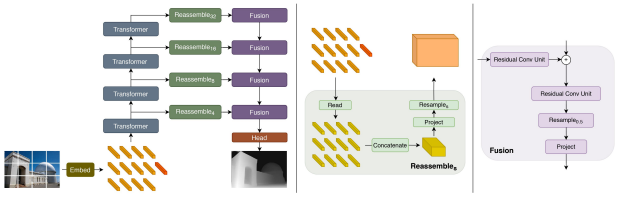


Figure 16. The DPT architecture.

The model used in our case is DPT-Large which was pre-trained on MIX 6 dataset which contains around 1.4M initialized with ImageNet-pre-trained weights. The Deep Learning pipeline gives for sure a better estimation of the 3D cloud point, but it takes much more time than the geometric pipeline.

4.3.2 Perspective-n-Point approach

The proposed geometric approach is based on the well known PnP algorithm, which allows to recover the extrin-

sics parameters from at least 3 correspondences between image points and word points. The first thing needed are indeed the correspondences, to obtain this it was proposed a reference model for a wheel assuming the dimensions with reference to the most common sizes of wheels mounted on vehicles. Our wheel model is characterized as follow:

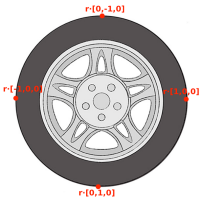


Figure 17. the 3D word model used in the PnP algorithm. r represent the wheel radius that was supposed to be equal to 20 cm

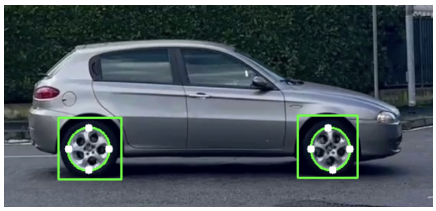


Figure 18. The image points corresponding to wheel model

There are various implementations of the PnP method, the one used in the project is the iterative solution, based on non linear optimization.
This approach requires 4 correspondeces, although there are methods with 3 correspondences (EPnP) V. Lepetit et al. 2009 [7]. The optimization process is based on the minimization of the reprojection error between the 2D projections and the observed points on the image. Below it was introduced a more formal definition of the PnP method.
The problem of pose estimation with PnP involves determining the position and orientation (translation and rotation) of a camera relative to a known three-dimensional object, using correspondences between known 3D points (in the object space) and their corresponding observed 2D points in the image.
Let us consider the following data:

- **3D Points**: $\mathbf{X}_i = (X_i, Y_i, Z_i)^T$, where $i = 1, \ldots, N$, are the known points in the object space.
- **2D Points**: $\mathbf{x}_i = (u_i, v_i)^T$, where $i = 1, \ldots, N$, are the corresponding observed points in the image.
- **Pose Parameters**: $\mathbf{R}$ (rotation matrix) and $\mathbf{t}$ (translation vector) define the camera's position and orientation relative to the object.

The projection relation could be defined as the relation which connects the 3D points in the object space to the 2D points in the image through the camera's pose parameters $\mathbf{R}$ and $\mathbf{t}$:

$$\lambda_i \mathbf{x}_i = \mathbf{K}(\mathbf{R}\mathbf{X}_i + \mathbf{t})$$

where:

- $\mathbf{K}$ is the camera's intrinsic matrix (parameters such as focal length and image center).
- λ_i is a scale factor.

The iterative method for solving PnP is based on minimizing the error between computed 2D projections and observed 2D points in the image, [8] Y. Zheng et al. 2013:

- **Initialization**: Initialize the pose parameters $\mathbf{R}$ and $\mathbf{t}$.
- **Iteration**:
  1. **Compute 2D Projections**: Using the current parameters $\mathbf{R}$ and $\mathbf{t}$, calculate the projections of the 3D points in the object space onto the image:
  $$\mathbf{x}_i^{\text{proj}} = \mathbf{K}(\mathbf{R}\mathbf{X}_i + \mathbf{t})$$
  2. **Compute Error**: Calculate the error between the computed projections $\mathbf{x}_i^{\text{proj}}$ and the observed 2D points $\mathbf{x}_i$:
  $$\mathbf{e}_i = \mathbf{x}_i - \mathbf{x}_i^{\text{proj}}$$
  3. **Minimize Error**: Use an optimization method such as the Levenberg-Marquardt algorithm to update the pose parameters $\mathbf{R}$ and $\mathbf{t}$ to minimize the overall error.
- **Convergence**: The iterative process continues until a convergence criterion is met, which may be based on reducing the error or reaching a maximum number of iterations.

Levenberg-Marquardt Algorithm:

The Levenberg-Marquardt algorithm is a nonlinear optimization method that combines gradient descent with conjugate gradient methods to find the minimum of a nonlinear error function. In this context, it efficiently updates the pose parameters $\mathbf{R}$ and $\mathbf{t}$ in each iteration, improving the estimation of the camera's position and orientation. This iterative approach is robust and widely used in the implementation of algorithms such as `solvePnP` in computer vision libraries like OpenCV, capable of handling various data types and input conditions, including noisy data or variable point configurations.

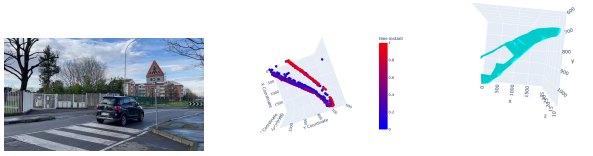
5 Analysis of the results

A first (qualitative) analysis of the result can be done by looking at the point clouds obtained with the different approaches in the various videos taken. Below there are the 3D plots of the cloud points:

Deep learning results

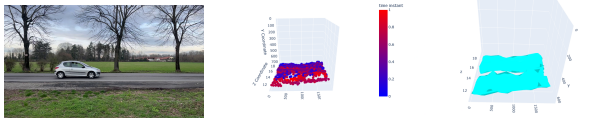
(a) Real video (b) 3D points (c) Road surface

Figure 19. Cars making a long curve



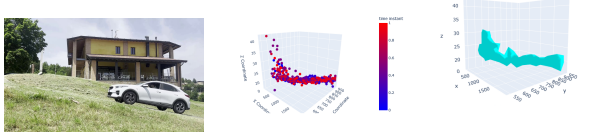
(a) Real video (b) 3D points (c) Road surface

Figure 20. Cars in a sharp and very distance curve



(a) Real video (b) 3D points (c) Road surface

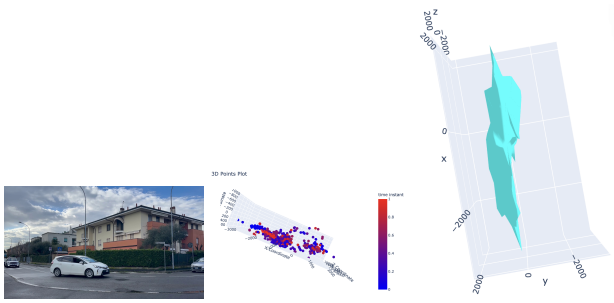
Figure 21. Cars in a straight road



(a) Real video (b) 3D points (c) Road surface

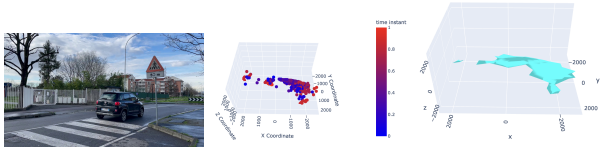
Figure 22. Cars on a hill

Geometric results



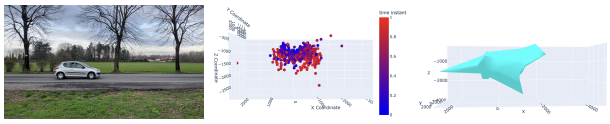
(a) Real video (b) 3D points (c) Road surface

Figure 23. Cars in a sharp and very distance curve



(a) Real video (b) 3D points (c) Road surface

Figure 24. Cars in a sharp and very distance curve



(a) Real video (b) 3D points (c) Road surface

Figure 25. Cars in a straight road

As it could be immediatly notice from the plots, the deep learning approach is much better than the geometric approach, this can have various causes, for example, few points were given to PnP, the assumption on the wheel radius does not take in to account the diversity of wheels in the videos. Although the reprojection error (which can be computed only in the geometric approach) is quite good. In particular the reprojection error was computed as:

$$\epsilon = \sqrt{(u - u')^2 + (v - v')^2} \tag{6}$$

where (u, v) are the observed point in the image and (u', v') are the re-projected point on the image. The average re-projection error obtained by averaging all the errors in each video and by averaging all the videos is 0.1 pixels, as a rule of thumb a re-projection error less than 1 pixel is considered good.

6 Conclusion

In conclusion, the result obtained is quite satisfactory and in line with reality. What could be further explored

is the geometric approach, where a deeper investigation could be conducted based on vanishing lines and points, extracting other characteristic features of the car from the images, such as the license plate and windshield.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[2] F Rashidi Fathabadi, J L. Grantner, S A. Shebrain, and I Abdel-Qader. Box-trainer assessment system with real-time multi-class detection and tracking of laparoscopic instruments, using cnn. *Acta Polytechnica Hungarica*, 19(2), 2022.

[3] Michael Shenoda. Lighting and rotation invariant real-time vehicle wheel detector based on yolov5. *arXiv preprint arXiv:2305.17785*, 2023.

[4] M. Fitzgibbon, A. W.and Pilu and R. B. Fisher. Direct least-squares fitting of ellipses. 21(5):476–480, May 1999.

[5] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 12179–12188, 2021.

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[7] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o(n) solution to the pnp problem. *International Journal of Computer Vision*, 81(2):155–166, 2009. doi:10.1007/s11263-008-0152-6. URL `https://doi.org/10.1007/s11263-008-0152-6`.

[8] Yinqiang Zheng, Yubin Kuang, Shigeki Sugimoto, Kalle Åström, and Masatoshi Okutomi. Revisiting the pnp problem: A fast, general and optimal solution. In *2013 IEEE International Conference on Computer Vision*, pages 2344–2351, 2013. doi:10.1109/ICCV.2013.291.