Visual Computing Kaggle Challenge 2023
Vehicle detection in dashcam videos

Carlos Santos García
CentraleSupélec
carlos.santos@student-cs.fr

1. Introduction

The objective of this project was to design a pipeline to detect vehicles in dashcam videos, using only classical machine learning techniques. The system I designed relied on classical feature descriptors like Histogram of Oriented Gradients [1], Bag-of-SIFT features [2], as well as handcrafted features from the images. My final system used a Gradient Boosting tree-based classifier that performed inference in a sliding-window manner over test images to detect whether each patch contained vehicles. This sliding window was designed in a way to find the trade-off between accuracy and inference runtime. My code is available here.

2. Pipeline Description

2.1. General description

Consider a binary classifier $\mathcal{C}$ trained on image patches to detect vehicle presence. The inference strategy I followed on test images was to extract features from overlapping patches from the test images in a sliding-window manner, computing for each patch the probability of containing a vehicle as estimated by $\mathcal{C}$. This gave a heatmap of probabilities for a given image (after summing the probabilities for each pixel and normalizing by the maximum aggregated value obtained). A threshold $\lambda = 0.63$ was then used to retain only the regions in the image where the aggregated probabilities $p \geq \lambda$. The remaining regions were binarized, and a connected component's algorithm was used to find connected regions in the image. For each of the regions, the smallest bounding box containing it was computed. If the bounding box was not smaller than 30×30 pixels, it was kept for the final submission. The next subsections will explain in more detail the pipeline.

2.2. Feature extraction

The classifier $\mathcal{C}$ needed features to assess whether a vehicle was present in a given patch, since the image by itself would have been too noisy and the classifiers used in this project were not as expressive as deep-learning-relying ones. The size of the patches to study was set to 64×64. The features I selected for the final system were:

- Histogram of Oriented Gradients features extracted
- Bag-of-SIFT features
- Histogram of colours values in the given patch
- Resized version of the patch (16×16)

Extracted features were normalized using Scikit-learn's `StandardScaler`. Using OpenCV's `HOGDescriptor` instead of Scikit-image's `hog` method reduced almost by ten times feature extraction runtime.

The vocabulary for the Bag-of-SIFT features was created using a 200 words vocabulary of SIFT features extracted from the training dataset, selected using the cluster centroids obtained with Scikit-learn's `MiniBatchKMeans`.

2.3. Training a binary classifier

The dataset used to train the binary classifier $\mathcal{C}$ was built extracting the bounding boxes of our training dataset containing vehicles. Bounding boxes with an area smaller than 30×30 pixels were not considered. This gave a number N_1 of training patches positively labelled (containing vehicles) that were resized to 64×64. If the original dataset contained N training images, I then extracted $\left\lfloor \frac{N_1}{N} \right\rfloor$ non-overlapping-with-labels 64×64 patches from each of the N images, randomly adding one extra patch with probability $p = \frac{N_1}{N} - \left\lfloor \frac{N_1}{N} \right\rfloor$. This ensured our training dataset was well-balanced, containing as many positive and negative samples.

Different machine learning classification models were considered, namely Support Vector Machine (SVM) classifiers, as well as boosting algorithms like Random Forest and Gradient Boosting classifiers. The final model was a Gradient Boosting classifier: this method clearly outperformed linear SVM classifiers in performance, and had similar performances to SVM using radial basis kernel function and a well tuned penalization parameter C obtained through cross-validation. Its main advantage was inference time,

since tree classifiers allow quick inference time and therefore highly reduce the runtime for sliding window inference compared to SVMs. Due to long training times ($+4$ hours) and low hardware availability, the parameters of the final Gradient Boosting classifiers could be optimized.

2.4. Sliding Window parameters

The sliding window strategy I followed needed careful tuning to perform reasonably well. Here are some of the essential aspects of my implementation.

A key aspect in my strategy to reduce inference time was to only consider patches that were contained between 20% and 80% of the total height of the images. This avoided looking for cars in the sky or on the front part of the recording car. In addition to this, the x-axis overlap for the sliding window was set to 75% as a trade-off between covering well the image and limiting the runtime of the algorithm. I also used a y-axis step of 2 pixels as a way to accelerate inference time.

As closer vehicles tend to be bigger on the image, the size of my sliding window started at 64×64 and was linearly increased as the height increased (i.e. as the window got closer to the recording camera). All patches were then resized to 64×64 before the extraction of features.

Overall, my inference strategy took in average 11 seconds to fully compute the bounding boxes of vehicles detected in an image.

3. Conclusion

This project shows how ambitious autonomous driving was before the deep-learning era, since classical machine learning algorithms seem unable to fully capture what vehicles look like, requiring time-consuming hyperparameter tuning. Further improvements for this project could be leveraging the temporal coherence present in videos, as well as combining different classifiers to perform more robust classification and adding external datasets to enrich the current one.

References

[1] N. Dalal et B. Triggs. Histograms of oriented gradients for human detection, 2005. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), IEEE, vol. 1, p. 886–893. 1

[2] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov. 2004. 1