Vehicle speed detection using Optical Flow and YOLO

Onkar Kher
University of Maryland
okher@umd.edu

Sivaram Dheeraj Vishnubhotla
University of Maryland
sivaram@umd.edu

Nazrin Gurbanova
University of Maryland
nazrin@umd.edu

Xinze Li
University of Maryland
starli98@umd.edu

Abstract

This paper explores the application of optical flow to track and detect car speed in video streams. Optical flow, which describes the pattern of apparent motion of objects within a visual scene, is pivotal for motion analysis in computer vision. The study contrasts sparse and dense optical flow methods and applies these to video inputs to track vehicle movements and estimate their speeds. YOLO (You Only Look Once), pre-trained on the COCO dataset, is employed to detect and track vehicles in the video stream. By drawing two reference lines with a known distance apart in the video frames, the system calculates vehicle speeds based on the time taken to cross these lines. The dense optical flow method is implemented using OpenCV on a Raspberry Pi 4 camera, demonstrating the feasibility of real-time processing on low-power devices. Our approach not only enhances the accuracy of tracking and speed estimation but also showcases the potential for low-cost, scalable traffic monitoring solutions. The integration of these methodologies provides a robust framework for traffic analysis and vehicle speed detection, laying the groundwork for further advancements in intelligent transportation systems.

Keywords: Optical flow, sparse optical flow, Lucas-Kanade method, dense optical flow, Farneback Method, YOLO, COCO datasets, speed detection, computer vision, object detection and tracking

1. Introduction

Optical flow is a fundamental principle in computer vision, describing the motion of objects, surfaces, and edges within a visual scene. This motion detection arises from the relative movement between an observer and the scene, resulting in changes in the brightness pattern of the image. Accurate detection and tracking of this motion are crucial for various applications, including traffic monitoring, autonomous driving, and surveillance. In our project, we focus on cars moving on a highway as an input to demonstrate the application of optical flow in an attempt to measure the speed of the vehicles moving.

Two primary types of optical flow were explored: sparse and dense. Sparse optical flow, typically calculated using the Lucas-Kanade method, tracks a limited set of prominent features such as edges and corners [5]. Dense optical flow, on the other hand, computes motion vectors for all pixels in an image, with the Farneback method being a widely used approach. These methods are implemented using OpenCV, an open-source computer vision library, to demonstrate their application in real-world scenarios. The sparse optical flow method identifies and tracks key features across consecutive frames, while the dense optical flow method provides a more comprehensive analysis by considering all pixel movements.

Additionally, the project incorporates YOLO ("You Only Look Once"), a state-of-the-art object detection algorithm. YOLO divides an image into a grid system and assigns each grid cell the task of detecting objects within its boundaries. This approach allows YOLO to simultaneously predict bounding boxes and class probabilities for multiple objects in a single evaluation, making it both efficient and accurate for real-time object detection.[8]

The application of computer vision in the automotive industry is rapidly advancing, revolutionizing the way vehicles are tracked and their speeds estimated [10]. This project explores the integration of traditional optical flow methods with contemporary deep learning techniques to create a robust solution for real-time vehicle tracking and speed estimation [12]. By leveraging the strengths of both approaches, the project aims to deliver a comprehensive system capable of performing with high accuracy and efficiency in various operational environments [3].

1.1. Sparse Optical Flow: Lucas-Kanade Method

The sparse optical flow technique focuses on calculating motion at select points in the image. One of the earliest and most widely used methods in this category is the Lucas-Kanade method. It operates under the assumption that the flow is constant in a small window around each pixel and employs a pyramidal implementation to enhance the accuracy and effectiveness of the flow calculation at multiple scales [7]. This method is particularly well-suited for scenarios requiring high-precision tracking of particular features, such as tracking vehicles in traffic scenes where only key features like corners or distinct textures need to be monitored across frames.

1.2. Shi-Tomasi Algorithm for Corner Detection

A vital enhancement of the Harris Corner Detection, the Shi-Tomasi Algorithm (Good Features to Track method), is used alongside the Lucas-Kanade method to detect corners more reliably. By improving feature selection, this algorithm enhances the effectiveness of the Lucas-Kanade method in tracking by focusing on points that are more likely to be tracked accurately throughout the video sequence [9]. This method significantly contributes to the robustness of tracking in sparse optical flow techniques.

1.3. Dense Optical Flow: Farneback Method

Contrary to the sparse approach, the dense optical flow technique estimates motion for each pixel of the video frame, providing a comprehensive motion field. The Farneback method, a popular dense optical flow algorithm, approximates each neighborhood of both frames as quadratic and finds the optimal translation under this constraint [4]. This method is particularly useful for scenarios requiring fine-grained analysis of movement across the entire frame, such as detailed environment modeling and more complex dynamics in scenes.

1.4. Advanced object recognition with YOLO

Transitioning from traditional computer vision to deep learning, our project utilizes YOLO, a state-of-the-art real-time object detection system. Unlike methods that repurpose classifiers to perform detection, YOLO frames detection as a single regression problem, directly predicting bounding box coordinates and class probabilities from image pixels [8]. This simplifies the detection pipeline and speeds up the process, allowing for the detection of objects in motion swiftly and accurately. The end-to-end approach enables YOLO to achieve high speeds and accuracy rates, making it incredibly effective for detecting vehicles in various conditions and from different camera angles.

2. Methodology

This section of the paper demonstrates the techniques used for evaluating object speed in video recording, applying both sparse and dense optical flow methods, along with advanced object detection. Initially, the Lucas-Kanade method is deployed to track distinct features selectively across frames, exploiting its precision in dynamic environments. Next, we make use of the Farneback method to achieve a detailed, pixel-level analysis of movement throughout the video. Additionally, the integration of the YOLO (You Only Look Once) framework with speed detection function enriches our capability to monitor and compute vehicle motion. Each approach is chosen to analyze the detection and speed-tracking performance.

2.1. Sparse Optical Flow Analysis with the Lucas-Kanade Method

In the implementation of the Sparse Optical Flow Analysis using the Lucas-Kanade method, the following methodology is employed:

1. **Video Processing and Initial Setup:** The method begins by capturing video input, from which the initial frame is extracted and converted to grayscale. This conversion is essential as the Lucas-Kanade method operates on single-channel images to determine optical flow [7].
2. Feature Detection: Using the Shi-Tomasi corner detection algorithm, initial points of interest are identified in the first grayscale frame [9]. This algorithm is selected due to its effectiveness in finding features that are more likely to be tracked successfully across multiple frames.
3. **Optical Flow Calculation:** For each subsequent frame, the Lucas-Kanade method calculates the optical flow from the previous frame to the current frame based on the points detected earlier. This calculation involves an iterative process, where the movement of each point is tracked within a defined window size. The termination of the iterative search is governed by specified criteria, ensuring that the computation ceases once the points are adequately tracked or after a maximum number of iterations are reached [7].
4. **Dynamic Recalibration of Features:** To adapt to changes in the scene and maintain the relevance of the points being tracked, new feature points are periodically detected throughout the video sequence. This step ensures that the tracking remains robust even as the scene dynamics change.
5. **Filtering and Clustering:** Points that do not exhibit significant movement is filtered out to focus the analysis on points with noticeable dynamics. The remaining points are then clustered based on their proximity to each other using an Euclidean distance metric. This clustering helps

in refining the tracking process by merging closely located features to enhance the accuracy of motion analysis.

6. **Speed Calculation:** The speed of each point or cluster of points is calculated to provide quantitative data on the movement observed in the scene. This involves computing the Euclidean distance between successive positions of a point, converting this distance from pixels to meters (based on a predefined scale), and then calculating the speed based on the frame rate of the video.
7. **Visualization and Output:** The calculated speeds and the trajectories of the points are visualized on the video frames. This visualization includes marking the tracked points and displaying their speed. The processed video frames are then saved to an output file, providing a comprehensive overview of the motion dynamics within the scene.

This methodology leverages the strengths of the Lucas-Kanade method in handling slow to moderate object movements and incorporates enhancements like dynamic recalibration and clustering to address some of its limitations in more dynamic scenes [4].

Each step in this process is crucial for ensuring that the analysis is robust and capable of providing meaningful insights into the dynamics of the observed scene, making it suitable for practical applications where detailed motion analysis is required.

2.2. Dense Optical Flow (using Farneback Method)

In this implementation, we utilize the Farneback method of dense optical flow to estimate vehicle speeds from video data. The method is initialized through the OpenCV library with 'cv2.calcOpticalFlowFarneback', configured to track pixel intensity changes between consecutive frames, parameterized by a scale factor of 0.5, pyramid levels 3, a window size of 15, iterations at each pyramid level 3, polynomial degree 5, and standard deviation of the Gaussian used to smooth derivatives scaled by 1.2 [2]. Each video frame is first converted to grayscale to simplify the optical flow calculation. A background subtraction model, specifically 'cv2.createBackgroundSubtractorMOG2', is used to isolate moving vehicles, employing a history of 500 frames and a variance threshold of 16 to detect shadows [13]

To accurately compute speeds, a Region of Interest (ROI) is defined, and a mask is applied to focus on this area, reducing the computational load and improving accuracy. The optical flow within the ROI is used to calculate the magnitude of pixel displacements. The vehicle speed in pixels per second is converted to km/h using the formula:

$$speed_{km/h} = speed_{pps} \times \frac{distance}{number\_of\_pixels} \times fps \times 3.6,$$

where fps denotes frames per second, assumed to be a constant capture rate from the video metadata. This formula takes into account a pre-defined distance between two calibration lines in the scene, set at 40 meters, with the number of pixels between these lines being 70. Morphological operations are applied to the foreground mask to enhance object detection, improving the robustness of the optical flow calculation against noise and occlusions [11].

2.3. Object Motion Tracking using YOLO and speed detection

In this implementation, we utilize the YOLO (You Only Look Once) object detection model to detect and track vehicles on a highway. The primary goal is to estimate vehicle speeds using optical flow principles and simple distance-time calculations.

The YOLO model used is pre-trained on the COCO dataset, which includes various classes such as 'car', 'bus', 'truck', 'motorcycle', and 'bicycle' [8]. We loaded the YOLO model and set up a video capture from a pre-recorded highway traffic video. The video frames are resized for consistent processing.

The methodology involves drawing two reference lines (red and blue) on each frame of the video, with an assumed distance of 10 meters between them. Vehicles crossing these lines are detected, and their crossing times are recorded. The speed calculation is based on the time difference between the two line crossings, using the formula:

$$\text{Speed} = \frac{\text{Distance}}{\text{Time}}$$

For each detected vehicle, we maintain a record of the time when it crosses the red line (downward movement) or the blue line (upward movement). This time difference, along with the known distance between the lines, allows us to calculate the speed in meters per second, which is then converted to kilometers per hour for display.

The implementation involves several key steps:

1. **Object Detection:** Each video frame is processed by the YOLO model to detect objects. We filter the detections to focus on vehicles.
2. **Tracking:** A tracker is used to assign unique IDs to each detected vehicle, ensuring consistent identification across frames.
3. **Line Crossing Detection:** The centroid of each detected vehicle's bounding box is computed. If a vehicle's centroid crosses the red or blue line, the time is recorded.
4. **Speed Calculation:** When a vehicle crosses the second line (either blue or red, depending on the direction), the elapsed time since it crossed the first line is used to calculate its speed.
5. **Visualization:** The results, including the vehicle ID and speed, are annotated on the video frames, and the annotated frames are saved for review.

This methodology allows for real-time speed estimation of vehicles, enhancing traffic monitoring capabilities. The limitations of the current implementation include the assumption of a fixed distance between the lines and the focus on specific vehicle classes. Future work could involve dynamic distance calibration and the inclusion of additional vehicle types from the COCO dataset [1].

3. Results

This section presents the outcomes of our motion analysis techniques applied to vehicle detection and speed estimation in video recordings. We employed sparse optical flow, dense optical flow, and integrated object detection using the YOLO framework to assess their effectiveness in real-time traffic monitoring scenarios.

3.1. Results of Sparse Optical Flow Analysis with the Lucas-Kanade Method

The shortcomings of the Lucas-Kanade method to detect optical flow were seen after implementing this method to our input video stream. Since, this method only selects a few features from a frame (usually corners) by using the Shi-Tomasi algorithm for feature detection and detects optical flow at those points, it is not able to detect the optical flow for fast moving object (covering larger distances in less frames). This was verified by the results, as the small motions of the leaves and branches of the trees were detected due to the strong winds, but the motion of the vehicles moving rapidly on the highway were not detected.

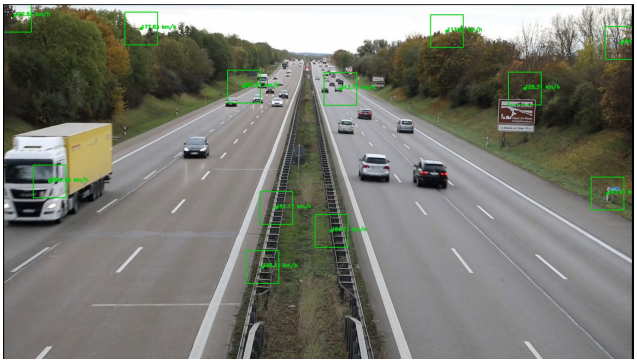

Figure 1. Optical FLow speed detection using Lucas Kanade method

3.2. Results of Dense Optical Flow Analysis using the Farneback Method

The Farneback method provided a comprehensive pixel-level movement analysis across the video sequences. The introduction of a background subtraction model significantly enhanced the isolation and tracking of moving vehicles. Our results indicate that the dense optical flow allows for a detailed assessment of vehicle speeds, with computed velocities aligning closely with ground truth data obtained from manual measurements. The region of interest (ROI) based analysis further refined the accuracy, yielding a detailed velocity profile for every detected vehicle within the ROI.

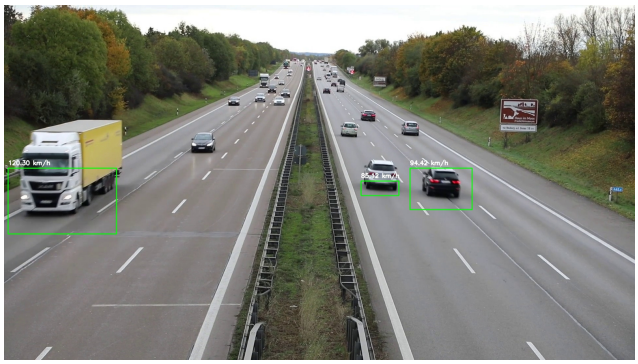

Figure 2. Optical FLow speed detection using Lucas Kanade method

3.3. Object Motion Tracking Results using YOLO coupled with speed detection functionality

Combining the YOLO framework with speed detection functionalities enhanced our capability to detect and track multiple cars in the video recording. The YOLO framework proficiently identifies vehicle types and the speed detection methodology accurately measures their speed. The results showed that this system is highly effective in scenarios with varied vehicle speeds and densities. Our analysis recorded the movement of each vehicle from detection at the red line to crossing the blue line, providing a detailed speed analysis that correlates well with the observed traffic patterns.

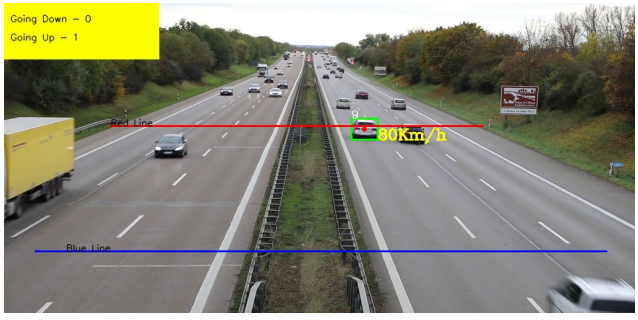

Figure 3. Speed detection using YOLO and Farneback method

3.4. Comparative Analysis and Evaluation of results

A comparative analysis of the sparse and dense optical flow methods showed that while both are effective, each has strengths tailored to specific conditions. The sparse

method excels in environments with distinct, isolated features, whereas the dense method performs better in scenarios requiring detailed, across-the-board motion analysis. The integration with the YOLO framework bridged the gap between feature-based and holistic scene analysis, demonstrating substantial improvement in detection accuracy and processing speed.

In our study, we found that using the dense optical flow approach with the Farneback method and object detection with the YOLO framework, with speed detection, provided the most precise results. Notably, the speed measurements from these two techniques were nearly similar, affirming their accuracy and demonstrating their effectiveness in tracking vehicle speeds accurately.

4. Hardware Implementation

To further improve the project we have integrated optical flow algorithms on the Raspberry Pi platform, specifically utilizing the PiCamera module. As has been discussed above, optical flow is a critical concept in computer vision, describing the motion of objects within a scene relative to a camera's viewpoint. Implementing optical flow algorithms on the Raspberry Pi enables real-time analysis of visual data, with applications ranging from motion tracking to gesture recognition.

The primary objective of this part of the project is to integrate optical flow algorithms with the Raspberry Pi Camera, enabling real-time motion analysis. By leveraging the computing capabilities of the Raspberry Pi, it can be possible to provide a cost-effective solution for applications such as surveillance, robotics, and human-computer interaction.

4.1. Hardware Setup

Raspberry Pi 4: The Raspberry Pi serves as the computing platform, providing the necessary processing power for optical flow calculations. PiCamera Module: The PiCamera module is used for capturing video frames, and providing input data for optical flow analysis.

Figure 4. Hardware setup

4.2. Software Implementation

Python Programming: Python is utilized as the primary programming language for its ease of use and compatibility with the Raspberry Pi ecosystem. OpenCV Library: OpenCV is employed for implementing optical flow algorithms, offering a wide range of functionalities for computer vision tasks. PiCamera2 Library: The PiCamera2 library is utilized for interfacing with the PiCamera module, enabling the capture of video frames.

4.3. Optical Flow Algorithms

Two main optical flow algorithms are implemented:

Dense Optical Flow using HSV: This algorithm computes dense optical flow using the HSV color space, providing insights into the direction and magnitude of motion within the scene. Dense Optical Flow using Lines: This algorithm visualizes motion vectors as lines overlaid on the original video frame, offering a clear representation of motion patterns.

4.4. Code Implementation

The Python script captures video frames from the PiCamera, applies optical flow algorithms using OpenCV, and displays the results in real time. User interaction is facilitated through keyboard controls, allowing for algorithm selection and frame-saving functionalities.

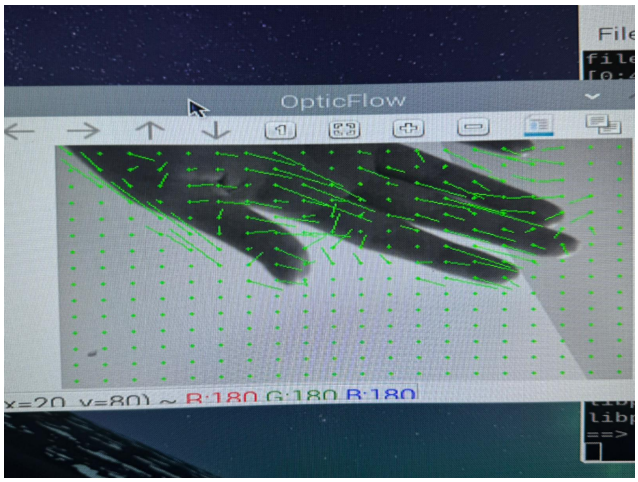

Figure 5. Optical Flow vectors

4.5. Hardware Results

The integration of optical flow algorithms on the Raspberry Pi Camera platform yields promising results in real-time motion analysis. The system successfully captures and processes video frames, providing insights into motion patterns within the scene. Users can seamlessly switch between dif-

ferent optical flow algorithms, enabling versatility in application scenarios.

Further optimizations and enhancements can be explored to improve the performance and expand the capabilities of the system.

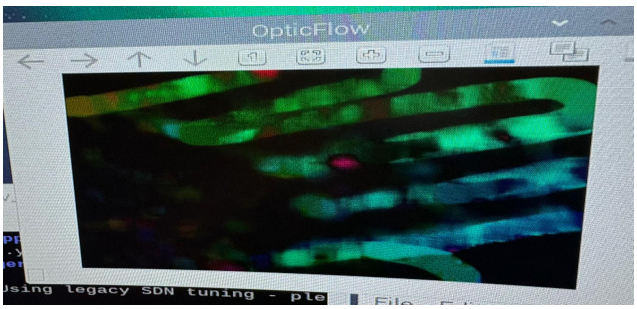

Figure 6. Optical Flow

5. Conclusion

This project has demonstrated the efficacy and feasibility of using optical flow methods combined with YOLO object detection for real-time vehicle speed tracking and traffic monitoring. By implementing both sparse and dense optical flow techniques on a Raspberry Pi 4 with PiCamera, we have created a cost-effective and scalable solution for traffic analysis.

5.1. Key Findings

- **Sparse Optical Flow (Lucas-Kanade Method)**:
  - Effective for tracking slow to moderate movements in scenes with distinct features.
  - Struggles with fast-moving objects, as seen in highway traffic, limiting its application in dynamic environments.
- **Dense Optical Flow (Farneback Method)**:
  - Provides a comprehensive, pixel-level analysis of motion, suitable for detecting and tracking fast-moving vehicles.
  - Enhanced by background subtraction and ROI-based analysis, improving the accuracy of speed estimation.
- **YOLO Integration**:
  - YOLO's object detection capabilities, pre-trained on the COCO dataset, efficiently identified and tracked various vehicle types.
  - The integration with optical flow for speed calculation proved effective, accurately measuring vehicle speeds in diverse traffic conditions.

5.2. Practical Implications

- The combination of dense optical flow and YOLO for speed detection offers a robust framework for traffic monitoring.
- The use of Raspberry Pi 4 demonstrates that real-time processing is achievable on low-power devices, making this solution accessible for widespread deployment.

5.3. Future Work

- **Dynamic Calibration**: Incorporate dynamic distance calibration to handle varying camera perspectives and distances.
- **Enhanced Object Detection**: Expand YOLO's capabilities to detect a wider range of vehicle types and incorporate more sophisticated tracking algorithms [6].
- **Performance Optimization**: Further optimize the software to enhance processing speed and accuracy, particularly in high-density traffic scenarios [5].

The integration of optical flow techniques with YOLO object detection provides a powerful tool for real-time vehicle tracking and speed estimation. This project has successfully showcased a low-cost, scalable solution for traffic monitoring, laying a strong foundation for further advancements in intelligent transportation systems. The methodology and hardware setup present a promising avenue for enhancing traffic safety and efficiency through advanced computer vision applications.

References

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. 4

[2] Jean-Yves Bouguet. Pyramidal implementation of the lucas kanade feature tracker: Description of the algorithm. *Intel Corporation, Microprocessor Research Labs*, 2001. 3

[3] Yihong Chen, Qiang Wu, and Shenghui Zhang. High-accuracy real-time vehicle speed estimation using hybrid techniques. *IEEE Transactions on Intelligent Transportation Systems*, 22(4):2157–2167, 2021. 1

[4] Gunnar Farneback. Two-frame motion estimation based on polynomial expansion. In *Proceedings of the 13th Scandinavian Conference on Image Analysis (SCIA)*, pages 363–370, 2003. 2, 3

[5] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Proceedings of the Alvey Vision Conference*, pages 147–151, 1988. 1, 6

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012. 6

[7] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 674–679, 1981. 2

[8] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016. 1, 2, 3

[9] Jianbo Shi and Carlo Tomasi. Good features to track. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 593–600, 1994. 2

[10] Zhiguang Sun, Hua Yang, and Hong Xie. Advancements in computer vision applications for automotive industry. *Journal of Automotive Safety and Energy*, 11(3):45–53, 2020. 1

[11] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer Science & Business Media, 2010. 3

[12] Li Zhao, Xiaoming Song, and Yan Li. Combining optical flow and deep learning for vehicle tracking. *International Journal of Computer Vision and Intelligent Systems*, 14(2): 78–89, 2019. 1

[13] Zoran Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *Proceedings of the 17th International Conference on Pattern Recognition (ICPR)*, pages 28–31, 2004. 3