Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 20.Б09-мм

Бочкарев Арсений Петрович

Автоматизация разметки видимого горизонта дороги по видеопотоку

Отчёт по учебной практике

в форме «Производственное задание»

Научный руководитель:
доцент кафедры системного программирования, к.т.н., Ю. В. Литвинов

Консультант:
Инженер-программист ООО «Тиквижн» М. С. Осечкина

Санкт-Петербург
2022

Оглавление

1. Введение **3**

2. Постановка задачи **5**

3. Обзор **6**

3.1. Инструменты, использующиеся для работы 6

3.2. Обзор существующих решений 7

4. Метод **9**

4.1. Детали реализации 11

5. Эксперименты **15**

6. Заключение **16**

6.1. Результаты 16

Список литературы **17**

1. Введение

Компания Apostera занимается разработкой системы дополненной реальности ADAS для помощи водителю на дороге. В число её функций, например, входит предупреждение водителя о покидании полосы, превышении скорости и прочее. Горизонт же в ней используется для отрисовывания объектов, их обрезания и стабилизации изображения.

Для валидации работы алгоритмов, связанных с горизонтом, необходимо сверять их вывод с эталонными данными, которые требуется разметить. Однако размечать горизонт на каждом кадре – слишком долгое и неточное занятие, поэтому для данной задачи работникам компании требуется автоматизированный инструментарий. Автоматизация подразумевает возможность влияния человека через пользовательский интерфейс на работу программы, например, прерваться на любом кадре и доуточнить горизонт (как вручную, так и с помощью внутренних методов из инструментария). Входными данными должны будут служить видео, отобранные сотрудниками заранее, результатом же работы будет файл с номерами кадров и координатами крайних точек горизонта на них.

Среди ближайших аналогов существуют лишь частичные решения, и, хотя алгоритмы и приёмы, используемые в них, могут быть применены для решения некоторых подзадач (таких, как, например, определение горизонта по статичному кадру или отслеживание перемещения объектов по видео), они не решают поставленную задачу целиком[1]. Например, в статье «Introduction to Motion Estimation with Optical Flow» (Lin, C., 2018) описывается процесс создания программы по отслеживанию объектов по видео, однако язык Python, используемый автором статьи, не позволяет полностью использовать приведённый код по причинам, которые будут описаны далее. Кроме того, в примере отсутствует пользовательский интерфейс. Аналогичные проблемы присутствуют и в утилите по определению горизонта по статичному кадру «horizon-detection» (Sall, S., 2019).

[1] более подробно в разделе 3.2

Таким образом, необходимость в создании требующегося инструмента, несмотря на наличие частичных решений, сохраняется. В рамках данной работы рассматривается его создание.

2. Постановка задачи

Целью работы является создание инструмента для автоматизации разметки видимого горизонта по видеопотоку. Для её выполнения были поставлены следующие задачи:

1. Провести анализ аналогов.
2. Реализовать первоначальное определение горизонта по статичному кадру, используя в том числе и уже существующие методы.
3. Реализовать отслеживание перемещения горизонта от кадра к кадру, используя в том числе и уже существующие методы.
4. Создать пользовательский интерфейс, позволяющий пользователю корректировать горизонт как вручную на любом кадре, так и путём изменения каких-либо технических параметров.
5. Написать документацию для работы с программой.
6. Провести эксперименты.

3. Обзор

3.1. Инструменты, использующиеся для работы

В требованиях к работе были заявлены следующие инструменты:

- Язык C++;
- Библиотека OpenCV;
- Фреймворк Qt.

Функции OpenCV, необходимые для работы программы:

- cv::cvtColor() – метод для преобразования изображения в другую цветовую гамму [1];
- cv::Canny() – метод определения границ на изображении [2];
- cv::HoughLines() – метод обнаружения прямых на изображении [3];
- cv::GaussianBlur() – метод по снижению шумов на изображении [4];
- cv::threshold() – метод выделения объектов на изображении [5];
- cv::morphologyEx() – метод для избавления от артефактов, которые могут появится при использовании threshold [6];
- cv::goodFeaturesToTrack() – метод для поиска выделяющихся точек на изображении («feature points»), таких, как углы или грани объекта [7];
- cv::estimateRigidTransform() – метод для получения матрицы изменений между двумя кадрами [8];
- cv::calcOpticalFlowPyrLK() – метод для определения сдвига точек между двумя кадрами [9].

Средства Qt, необходимые для работы программы:

- QPushButton – для выбора файла, переключения кадров, сохранения или загрузки и паузы или запуска автоматизированного определения горизонта;
- QSpinBox – для изменения численных параметров, таких, как, например, частота проверки кадров;
- QLineEdit – для ввода имени файла пользователем;
- QLabel – для вывода кадров на экран;

- QProgressDialog – для уведомления пользователя о том, какой процент работы остался во время пре-прохода по всем кадрам при выборе файла;
- QFileDialog – для выбора файла пользователем.
- QMessageBox – для вызова диалога с пользователем;

3.2. Обзор существующих решений

Среди существующих решений ближайшими аналогами являются:

- построение горизонта по статичному кадру.
  Определение линии горизонта двумя методами: с использованием функций cv::Canny() и cv::HoughLines() и с использованием cv::GaussianBlur(), cv::threshold() и cv::morphologyEx(). Некоторые шаги из первого способа (такие, как, например, использование функций cv::cvtColor(), cv::Canny() и cv::HoughLines()), решающего данную задачу, были использованы в работе целиком по причине своей реализации на языке C++ [11, 12], другие же (метод с использованием cv::GaussianBlur()), адаптированы под данный язык, из-за того, что изначально были написаны на языке Python [13];
- стабилизация видео [14].
  Делается для повышения точности отслеживания объектов в кадре. Проводится предварительный проход по всем кадрам видео и их стабилизация методами языка C++ с библиотекой OpenCV (используемые функции: cv::cvtColor(), cv::goodFeaturesToTrack(), cv::estimateRigidTransform(), cv::calcOpticalFlowPyrLK()). Код из приведённого источника был взят и немного изменён под уже имеющуюся на момент написания программу;
- отслеживание объектов в кадре [15].
  Приведённая статья демонстрирует теорию и процесс написания программы по отслеживанию объектов на видео для языка Python с использованием функций, аналогами которых в языке C++ являются cv::goodFeaturesToTrack() и cv::calcOpticalFlowPyrLK(). При-

ведённый в источнике код был адаптирован под используемые в работе инструменты.

- пользовательский интерфейс.
  Делается в соответствиями с поставленной задачей о возможности пользователя влиять на определение горизонта. В качестве замены Qt рассматривались GUI средства библиотеки OpenCV [10] и библиотека SFML [18], но оба варианта были отклонены по причине меньшего (по сравеннию с используемым на данный момент фреймворком) функционала[2], а также менее удобно организованной работы с ними[3].

Таким образом, лишь часть из аналогов может быть полностью использована в работе, в то время как оставшиеся аналоги необходимо было изменять под язык C++. Кроме того, никакие из источников не располагают пользовательским интерфейсом, для которого было принято решение о разработке с нуля с использованием фреймоворка Qt.

[2]Например, в обоих вариантах изначально отсутствуют аналоги QProgressDialog

[3]Например, в GUI OpenCV отсутствует аналог Qt Creator, а существующие аналоги Qt Creator для библиотеки SFML требуют дополнительного времени на установку

4. Метод

В процессе работы было реализовано приложение, позволяющее находить горизонт, отслеживать его перемещение от кадра к кадру, доопределять его на любом кадре в случае необходимости, а также загружать и сохранять найденные координаты на каждом кадре. Горизонт отображается поверх изначальных кадров. Алгоритм предварительной стабилизации видео представлен на рисунке 1, алгоритм автоматизации процесса отслеживания горизонта представлен на рисунке 2. Пример использования программы представлен на рисунке 3. Исходный код находится на Github[4].

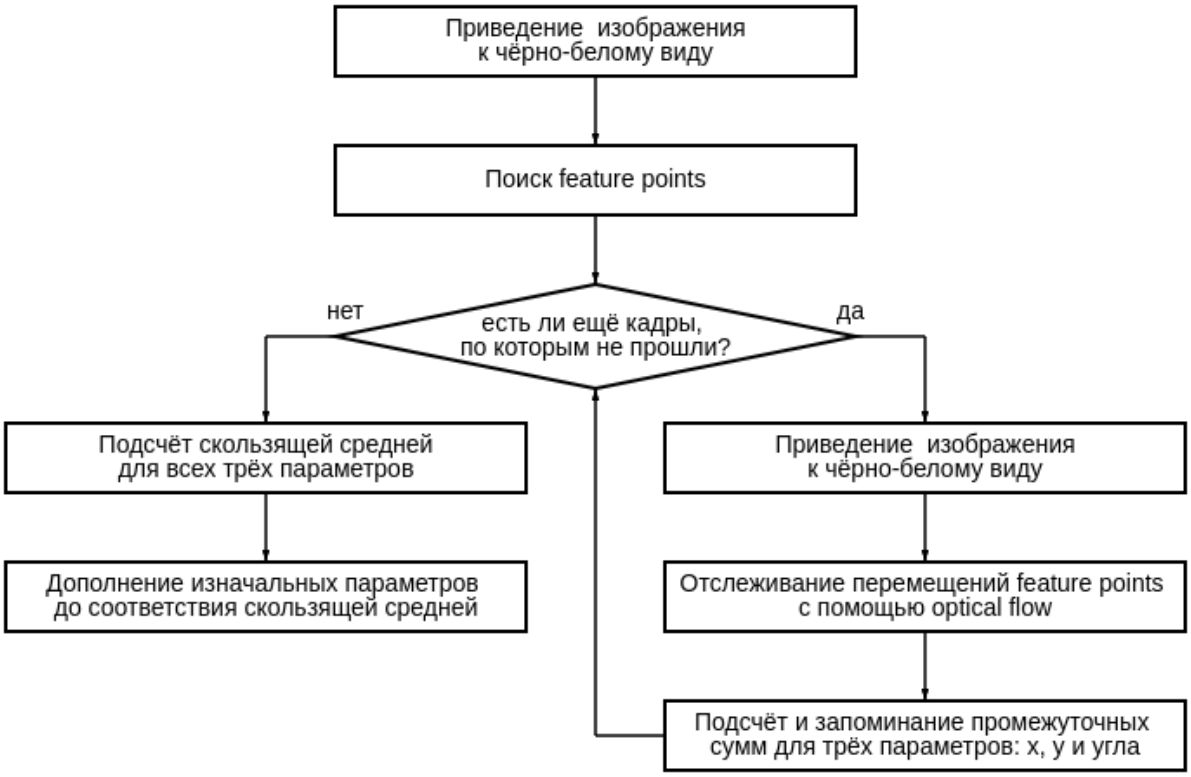

Рис. 1: Алгоритм стабилизации видео

[4]Исходный код: https://github.com/Mexanobar/sci_practice_21-22

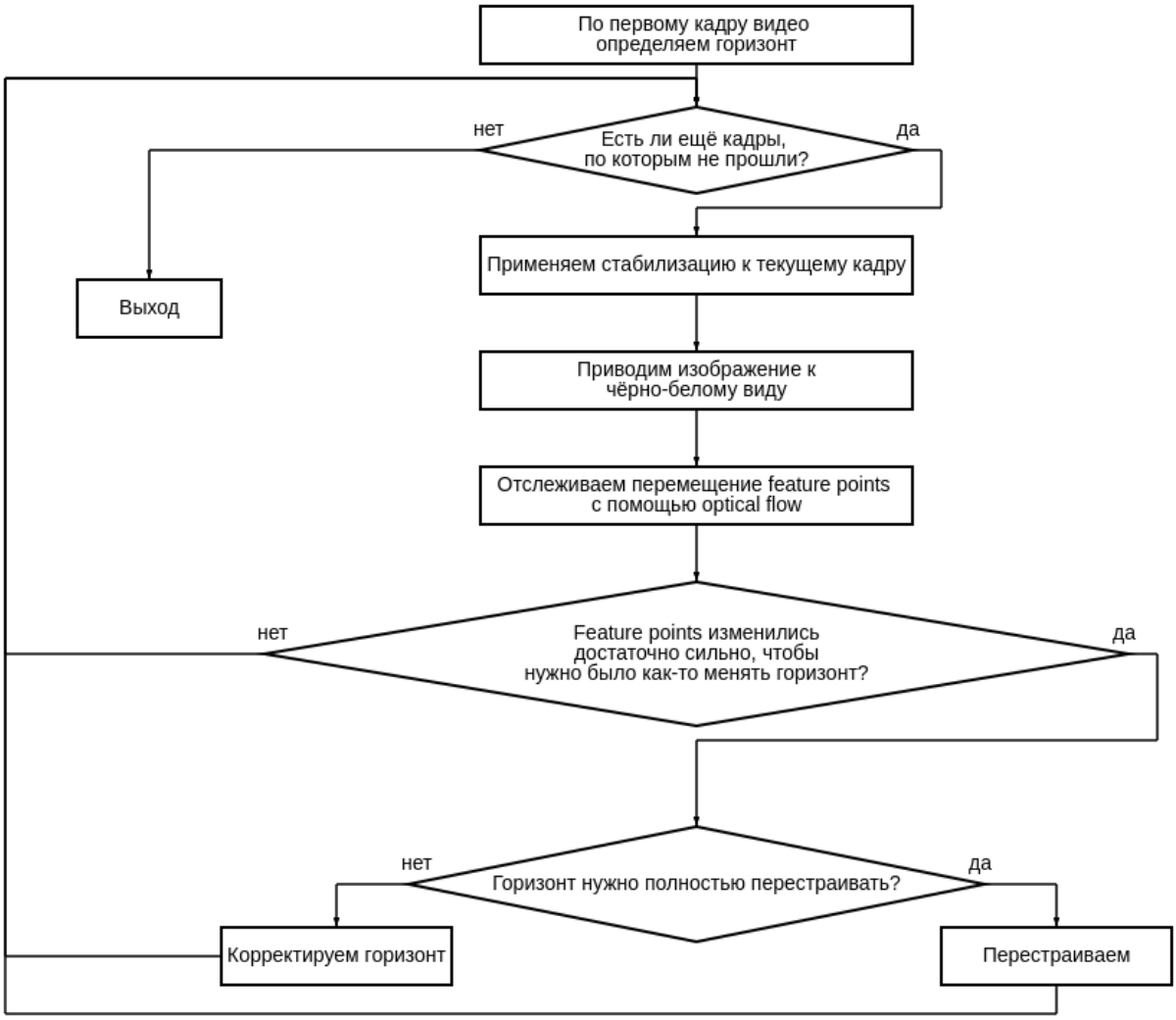

Рис. 2: Алгоритм автоматизации процесса отслеживания горизонта

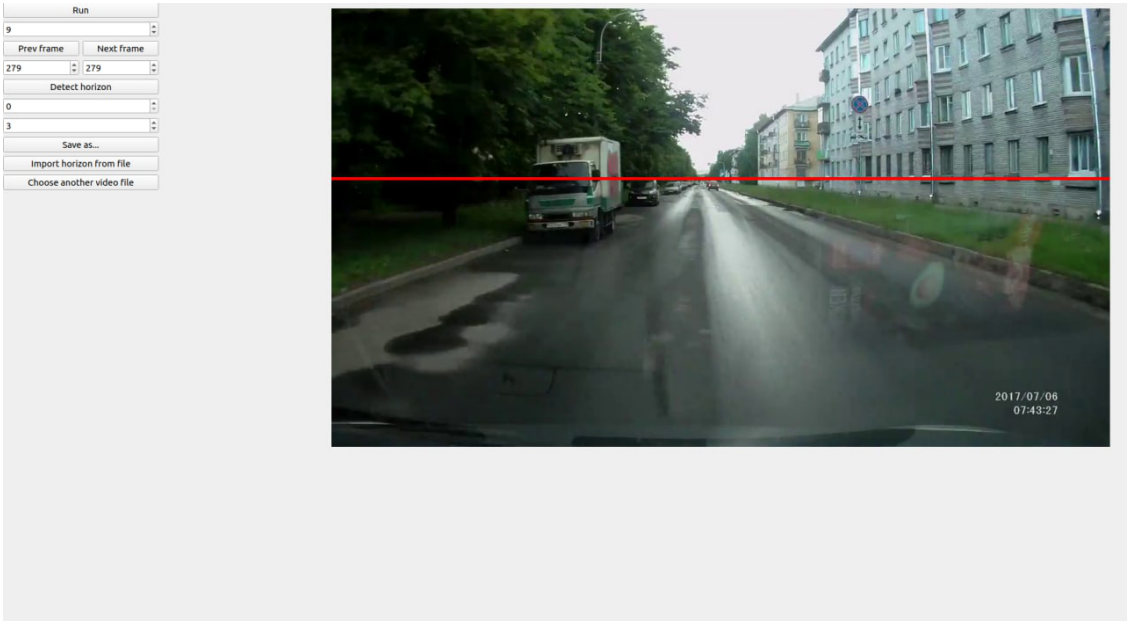

Рис. 3: Пример использования программы

4.1. Детали реализации

- Первоначальное определение горизонта:
    1. Метод через cv::Canny и cv::HoughLines описан в соответствующем источнике [12];
    2. Метод через cv::GaussianBlur, cv::threshold и cv::morphologyEx:
        (a) cv::cvtColor().
        Приведение изображения к черно-белому виду;
        (b) cv::GaussianBlur().
        При помощи данной функции на изображении снижаются шумы;
        (c) cv::threshold().
        На объектах выделяются изображения, в результате чего все пиксели, занимаемые ими, окрашиваются в чёрный цвет, остальные же – в белый;
        (d) cv::morphologyEx().
        Избавление от возможных артефактов, оставшихся после использования threshold() на предыдущем шаге;
        (e) Прохождение по точкам изображения, близким к правой и левой границе, сверху вниз для нахождения первых занятых объектами пикселей – тогда первые точки над ними и будут y-координатами горизонта.

    Данный метод подходит для специфических случаев, как, например, шоссе, тогда его применение более оправдано, нежели использование первого способа, по причине более высокой скорости работы.
- Отслеживание перемещения горизонта:
    1. Предварительная стабилизация видео, которое требуется разметить:
        (a) cv::cvtColor().
        Приведение изображения к черно-белому виду;
        (b) cv::goodFeaturesToTrack().
        Поиск как-либо выделяющихся точек на первом кадре

(например, углов или граней объектов);

(c) cv::calcOpticalFlowPyrLK().
С помощью данной функции определяется положение найденных на предыдущем шаге точек на следующем за ним кадре;

(d) cv::estimateRigidTransform().
Получение матрицы изменений между двумя идущими друг за другом изображениями;

(e) Из матрицы, полученной шагом выше, берутся три параметра: изменение по x, y и углу;

(f) Подсчёт промежуточных сумм для всех параметров;

(g) Подсчёт скользящей средней для всех параметров;

(h) Дополнение параметров на каждом кадре до соответствия скользящей средней;

(i) На этапе отслеживания горизонта дополненные параметры применяются к текущему кадру.

2. Применение дополненных параметров к текущему кадру;
3. cv::cvtColor().
Приведение изображения к черно-белому виду;
4. cv::goodFeaturesToTrack().
Поиск как-либо выделяющихся точек на первом кадре (например, углов или граней объектов);
5. cv::calcOpticalFlowPyrLK().
С помощью данной функции определяется положение найденных на предыдущем шаге точек на следующем за ним кадре;
6. Подсчёт изменений feature points между кадрами.
Всего существует два способа влияния на горизонт: корректировка и полное перестроение. Оба этих вида влияния опираются на изменения feature points.
   - Корректировка:
   Заметно значительное изменение feature points и можно однозначно понять, куда нужно смещать горизонт. Ве-

личина смещения определяется по расстоянию между одними и теми же точками на двух следующих друг за другом кадрах.

– Полное перестроение:

Заметно значительное изменение feature points, однако однозначно сказать, куда необходимо смещать горизонт, нельзя. В этом случае используется один из методов, определённых ранее.

- Пользовательский интерфейс:
  1. Входные и выходные даннные:

     На вход пользователь может подать видео форматов .AVI, .mp4 и .h264.

     На выход может быть (по желанию пользователя) подан файл формата .txt, число строк в котором соответствует количеству кадров в видео, с которым пользователь работал, на каждой строчке которого через пробел выписаны y-координаты линии горизонта (левая и правая соответственно).
  2. Хранение кадров:

     После предварительного прохода по всем кадрам видео, они будут сохранены в папке, имеющую то же название, что и имя видеофайла, с которым работает пользователь, и располагающуюся рядом с исполняемым файлом. В случае выбора другого видео или закрытия программы, эта папка будет удалена.
  3. Доопределение горизонта:

     На любом кадре возможно нажать на кнопку «Pause» и путём либо нажатия стрелочек, либо вводом координат в соответствующие поля доопределить горизонт. После этого при нажатии на кнопку «Run» горизонт будет отслеживаться с учётом изменённых координат. Также присутствует возможность определить горизонт программно по нажатию кнопки «Detect horizon».

4. Сохранение и загрузка:

   В любой момент работы можно сохранить текущие результаты нажатием на кнопку «Save». Также, в случае, если результат работы не был сохранён, и было закрыто окно программы, либо пользователь нажал на кнопку «Choose another video file», то пользователю будет предложено сохранить текущий прогресс. Присутствует возможность загрузки координат по нажатию на кнопку «Import horizon from file», но только в том случае, если количество кадров в выбранном файле совпадает с количеством кадров в текущем выбранном видео.

5. Эксперименты

Описание экспериментов:

1. Оценка точности работы первоначального определения горизонта.

   Данный аналог [12] после проведённых тестов на готовых датасетах «YorkUrbanDB» [16] и «NightTimeDrivingTest» [17] показал себя лучше, нежели описанный в предыдущем семестре.

2. Сбор отзывов пользователей.

   Существует проблема в работе стабилизации: при определении преобразования от кадра к кадру в ряде случаев методы OpenCV могут путать проезд автомобиля вперед с опусканием камеры, из-за чего горизонт может быть поднят вверх. С точки же зрения интерфейса и простоты использования отзывы положительные.

3. Структурированный анализ UX.

   По метрике «System Usability Scale» оценка составляет 90, в то время как средняя оценка (согласно данному [19] источнику) составляет 68 баллов.

6. Заключение

6.1. Результаты

- Проведён анализ аналогов.
- Реализовано первоначальное определение горизонта по статичному кадру;
- Реализовано отслеживание перемещения горизонта от кадра к кадру.
- Создан интерфейс, позволяющий пользователю корректировать горизонт как вручную на любом кадре, так и путём изменения каких-либо технических параметров;
- Создана документация для работы с программой;
- Проведены эксперименты.

Исходный код: https://github.com/Mexanobar/sci_practice_21-22 (Пользователь: «Mexanobar»).

Список литературы

[1] OpenCV: *Color Space Conversions* (2021), URL: https://docs.opencv.org/3.4/d8/d01/group__imgproc__color_-_conversions.htmlga397ae87e1288a81d2363b61574eb8cab (online; accessed 11.12.21)

[2] OpenCV: *Canny Edge Detection* (2021), URL: https://docs.opencv.org/3.4/da/d22/tutorial_py_canny.html (online; accessed 11.12.21)

[3] OpenCV: *Hough Line Transform* (2021), URL: https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html (online; accessed 11.12.21)

[4] OpenCV: *Image Filtering* (2021), URL: https://docs.opencv.org/4.x/d4/d86/group__imgproc__-filter.htmlgaabe8c836e97159a9193fb0b11ac52cf1 (online; accessed 11.12.21)

[5] OpenCV: *Miscellaneous Image Transformations* (2021), URL: https://docs.opencv.org/4.x/d7/d1b/group__imgproc__-misc.htmlgae8a4a146d1ca78c626a53577199e9c57 (online; accessed 11.12.21)

[6] OpenCV: *Image Filtering* (2021), URL: https://docs.opencv.org/3.4/d4/d86/group__imgproc__-filter.htmlga67493776e3ad1a3df63883829375201f (online; accessed 11.12.21)

[7] OpenCV: *Feature Detection* (2021), URL: https://docs.opencv.org/3.0-beta/modules/imgproc/doc/feature_-detection.htmlgoodfeaturestotrack (online; accessed 11.12.21)

[8] OpenCV: *Object Tracking* (2021), URL: https://docs.opencv.org/3.4/dc/d6b/group__video__-

track.htmlga762cbe5efd52cf078950196f3c616d48 (online; accessed 11.12.21)

[9] OpenCV: *Motion Analysis and Object Tracking* (2021), URL: https://docs.opencv.org/3.0-beta/modules/video/doc/motion_-analysis_and_object_tracking.htmlcalcopticalflowpyrlk (online; accessed 11.12.21)

[10] OpenCV: *Gui Features in OpenCV* (2022), URL: https://docs.opencv.org/4.x/dc/d4d/tutorial_py_table_of_-contents_gui.html (online; accesed 18.05.22)

[11] Nieto, M.: *Lane markings detection and vanishing point detection with OpenCV.* (2011), URL: https://marcosnietoblog.wordpress.com/2011/12/27/lane-markings-detection-and-vanishing-point-detection-with-opencv/ (online; accessed 11.12.21)

[12] Rahul, K.: *Vanishing Point Detection with OpenCV, Python C++* (2021), URL: https://kediarahul.medium.com/vanishing-point-detection-with-opencv-python-c-4d1ed00a8cd5 (online; accessed 11.12.21)

[13] Sall, S.: *Horizon Detection* (2019), URL: https://github.com/sallamander/horizon-detection (online; accessed 11.12.21)

[14] Nghia, H.: *Simple video stabilisation using opencv.* (2014), URL: http://nghiaho.com/?p=2093 (online; accessed 11.12.21)

[15] Lin, C.: *Introduction to Motion Estimation with Optical Flow* (2018), URL: https://nanonets.com/blog/optical-flow/ (online; accessed 11.12.21)

[16] Elder Laboratory: *York Urban Line Segment Database* (2008), URL: https://www.elderlab.yorku.ca/resources/york-urban-line-segment-database-information/ (online; accessed 15.12.21)

[17] Dai, D., Van Gool, L.: *Dark Model Adaptation: Semantic Image Segmentation from Daytime to Nighttime* (2018), URL: http://people.ee.ethz.ch/ daid/NightDriving/ (online; accessed 18.05.22)

[18] Gomila, L., SFML: *Simple and Fast Multimedia Library* (2022), URL: https://sfml-dev.org/ (online; accessed 18.05.22)

[19] Sauro, J.: *Measuring Usability with the System Usability Scale (SUS)* (2011), URL: https://measuringu.com/sus/ (online; accessed 20.05.22)