Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Омский государственный технический университет»

Факультет информационных технологий и компьютерных систем
Кафедра «Прикладная математика и фундаментальная информатика»

Пояснительная записка
по дисциплине «Проектная деятельность»

Студента Курпенова Куата Ибраимовича
фамилия, имя, отчество полностью

Курс 3, группа ФИТ-222

Направление 02.03.02 Фундаментальная информатика
и информационные технологии
код, наименование

Руководитель доц., канд. физ.-мат. наук
должность, ученая степень, звание

Девятерикова М. В.
фамилия, инициалы

Выполнил
дата, подпись студента

Проверил
дата, подпись руководителя

Омск 2024

СПИСОК ИСПОЛНИТЕЛЕЙ

Ответственный исполнитель _____ Курпенов К. И.

дата, подпись исполнителя

Курс 3, группа ФИТ-222
Направление 02.03.02 Фундаментальная информатика и информационные технологии

РЕФЕРАТ

Пояснительная записка 19 с., 8 рис., 3 источн., 1 прилож.
ПРОЕКТ, КОМПЬЮТЕРНОЕ ЗРЕНИЕ, СТЕРЕОЗРЕНИЕ, ЛИДАР, КАРТА ГЛУБИНЫ, PYTHON, OPENCV.

Объект проекта — определение расстояний до препятствий при автопилотировании.

Цель проекта — изучить принципы эпиполярной геометрии и создать устройство на основе стереокамеры, которое сможет заменить лидар и не будет уступать ему в функциональности и цене.

В процессе выполнения проекта я изучил теоретическую основу стереоскопического зрения и запрограммировал устройство для построения карты глубины.

В результате выполнения проекта было создано устройство, которое может определять расстояния до различных объектов.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современные технологии трёхмерного восприятия окружающего мира находят все более широкое применение в различных областях, включая робототехнику, автономные транспортные средства, системы дополненной реальности, а также в области картографии и геодезии. Одним из ключевых аспектов этих технологий является способность точно измерять расстояния и строить карты глубины, что позволяет создавать детализированные трёхмерные модели окружающей среды.

Традиционно для получения данных о глубине используются лидары, которые обеспечивают высокую точность и качество измерений. Однако, несмотря на свои достоинства, такие системы имеют значительные недостатки, включая высокую стоимость, сложность в эксплуатации и чувствительность к погодным условиям. В результате этого, существует потребность в разработке более доступных и универсальных решений, которые могли бы предоставить сопоставимые результаты по более низкой цене.

В данной курсовой работе рассматривается разработка устройства на основе стереокамеры, которое способно строить карты глубины и выступает в роли недорогого аналога лидара. Стереозрение, как метод получения информации о глубине, основывается на анализе двух изображений, полученных с различных точек зрения, что позволяет вычислять расстояния до объектов в сцене. Этот подход не только снижает стоимость системы, но и делает её более доступной для широкого круга пользователей.

Важность данной работы заключается не только в создании нового устройства, но и в возможности его применения в различных сферах, включая автоматизацию процессов, улучшение навигации автономных устройств и восприятия ими окружающей среды, а также в образовательных проектах.

Таким образом, данная курсовая работа направлена на решение актуальной задачи создания недорогого и эффективного устройства для построения карт глубины, что имеет значительное значение в условиях современного технологического прогресса.

1 Постановка задачи

В данной курсовой работе ставится задача разработки устройства на основе стереокамеры, способного строить карту глубины, аналогичную по функциональности лидарам, но с более доступной ценой. Основные цели и задачи, которые необходимо решить в рамках данной работы, включают:

1. Получение фундаментальных знаний в области эпиполярной геометрии: найти статьи, учебники и другие материалы для большего погружения в тему стереоскопического зрения.

2. Исследование существующего решения на основе лидаров: провести анализ методов построения карт глубины. Определить их преимущества и недостатки, а также выявить возможности для улучшения.

3. Разработка архитектуры устройства: создать концепцию устройства, установить аппаратные и программные требования для достижения необходимой точности и производительности.

4. Имплементация алгоритмов обработки: реализовать алгоритмы для обработки данных, получаемых от стереокамеры, включая калибровку, сопоставление изображений и построение карты глубины.

5. Тестирование и валидация: провести испытания устройства. Сравнить полученные результаты с данными, полученными с помощью лидаров.

6. Анализ результатов: оценить эффективность разработанного устройства, выявить его преимущества и ограничения, а также предложить рекомендации по дальнейшему улучшению и возможным направлениям применения.

2 Решение

2.1 Описание математического аппарата стереозрения

Для написания эффективного программного кода требуются фундаментальные знания в области решения задачи. В моём случае такой областью является *эпиполярная геометрия*.

Эпиполярная геометрия это ключевая концепция в области стереозрения и компьютерного зрения, которая описывает взаимосвязи между трехмерными точками и их проекциями на двухмерные изображения, полученные с разных ракурсов. Эта геометрия позволяет ограничить пространство поиска соответствий между точками на изображениях, что значительно упрощает задачи, связанные с восстановлением трехмерной сцены. Приведём основные понятия эпиполярной геометрии:

1. Эпиполь — это точка на плоскости изображения, где проецируется центр камеры другой плоскости изображения. В стереозрении есть два эпиполя: $e1$ и $e2$, которые соответствуют центрам камер $C1$ и $C2$ соответственно

2. Базовая линия — это прямая, соединяющая центры двух камер $C1$ и $C2$. Она играет ключевую роль в определении эпиполярной плоскости.

3. Эпиполярная плоскость — создаётся при пересечении базовой линии с плоскостями изображений обеих камер. Эта плоскость содержит точку 3D объекта и центры проекции обеих камер.

4. Эпиполярная линия — это линия на плоскости изображения, которая соответствует проекции точки 3D объекта на другую камеру. Каждая точка на одной плоскости изображения соответствует линии на другой плоскости, что упрощает задачу поиска соответствий между изображениями.

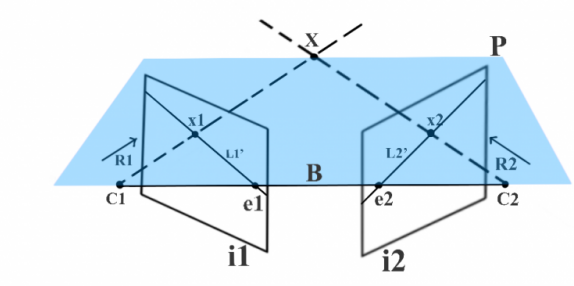

Рисунок 1 – Изображение, поясняющее эпиполярную геометрию

У нас есть эпиполярная плоскость P, созданная с использованием базовой линии B и луча R_1. e_1 и e_2 — эпиполи, а L_2 — эпиполярная линия. Исходя из эпиполярной геометрии данного рисунка, пространство поиска для пикселя в изображении i_2, соответствующего пикселю x_1, ограничено одной двумерной линией, которая является эпиполярной линией для i_2. Это называется эпиполярным ограничением.

Есть ли способ представить всю эпиполярную геометрию единой матрицей? Более того, можем ли мы рассчитать эту матрицу, используя только два захваченных изображения? Хорошая новость в том, что такая матрица существует называется фундаментальной матрицей. Не вдаваясь в подробности, привожу формулу для расчёта фундаментальной матрицы:

$$x_2^T F x_1 = 0$$

Как только F известна, мы можем найти эпиполярную линию L_2, используя формулу:

$$L_2 = F x_1$$

Если мы знаем L_2, то можно ограничить наш поиск пикселем x_2, соответствующим пикселю x_1, с помощью эпиполярного ограничения.

2.2 Исследование существующего решения на основе лидаров

Лидар (LiDAR, от английского "Light Detection and Ranging") — это технология, использующие лазерные лучи для определения расстояний до объектов и создания трёхмерных карт окружающей среды. Лидары работают по принципу измерения времени, необходимого лазерному лучу для отражения от объекта и возвращения к сенсору. Это время преобразуется в расстояние с использованием скорости света. Лидары могут генерировать до миллиона импульсов в секунду, создавая облака точек, которые затем обрабатываются для формирования детализированных 3D-моделей. Разберём преимущества и недостатки лидаров:

- Высокая точность — лидары обеспечивают измерения с точностью до нескольких миллиметров, что делает их идеальными для задач, требующих высокой детализации.
- Широкий угол обзора — некоторые системы могут охватывать до 360 градусов, что позволяет получать полную картину окружающей среды.
- Низкие затраты на большие площади — лидары позволяют быстро и экономично собирать данные на больших территориях.

- Ограниченное восприятие — лидары не могут распознавать дорожные знаки или цвета светофоров, что может ограничивать их использование в некоторых сценариях.
- Высокие технические требования — для оптимальной работы требуется соблюдение определенных условий (например, температуры), а также регулярное техническое обслуживание.
- Чувствительность к погодным условиям — хотя лидары работают в сложных условиях, сильные дожди могут негативно влиять на их производительность.

Рисунок 2 – Лидар, закреплённый на беспилотном автомобиле

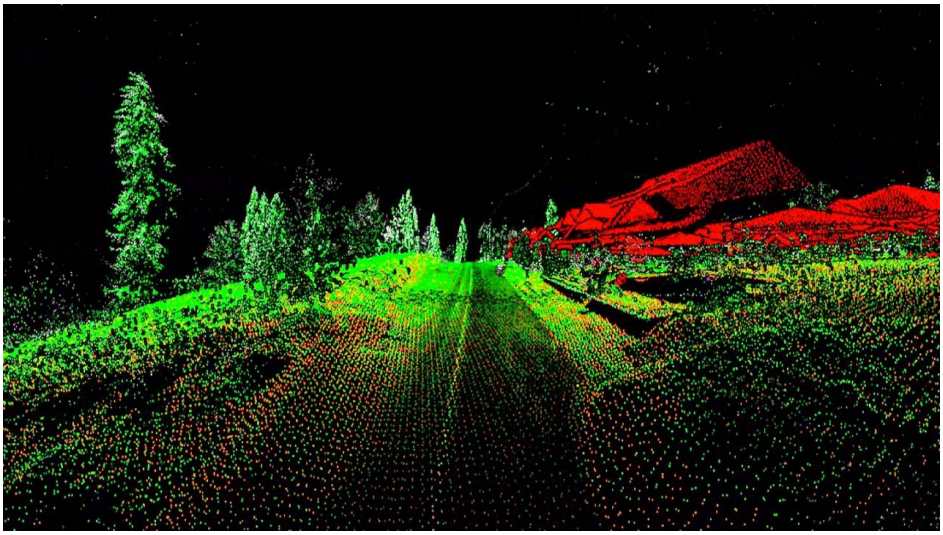

Рисунок 3 – Карта глубины, полученная с лидара

2.3 Описание устройства

Построение устройства будет происходить на основе стереокамеры:

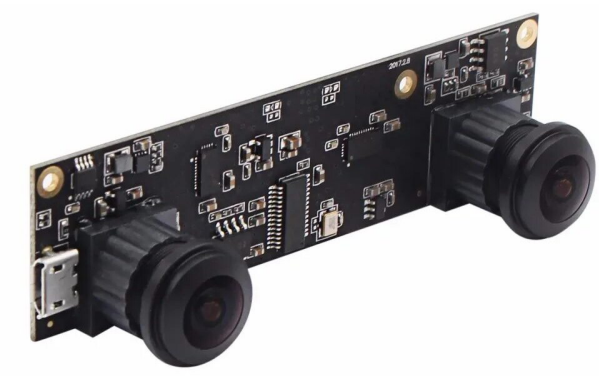

Рисунок 4 – Внешний вид стереокамеры

К сожалению, фотографию и название личной стереокамеры прикрепить не имею возможности, так как брал камеру в аренду и на момент написания курсовой работу уже вернул устройство владельцу.

2.4 Имплементация алгоритмов обработки

В стереосистемах важно, чтобы обе камеры были правильно синхронизированы. Калибровка обеспечивает правильное определение взаимного расположения камер, что позволяет точно сопоставлять изображения и получать корректные эпиполярные линии, поэтому перед получением разницы между изображениями, нужно откалибровать устройство:

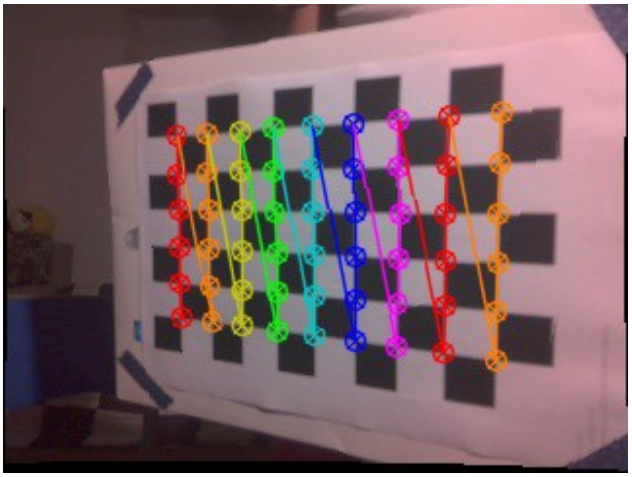

Рисунок 5 – Калибровка стереокамеры

После калибровки остаётся подобрать параметры для корректной работы библиотеки стереозрения:

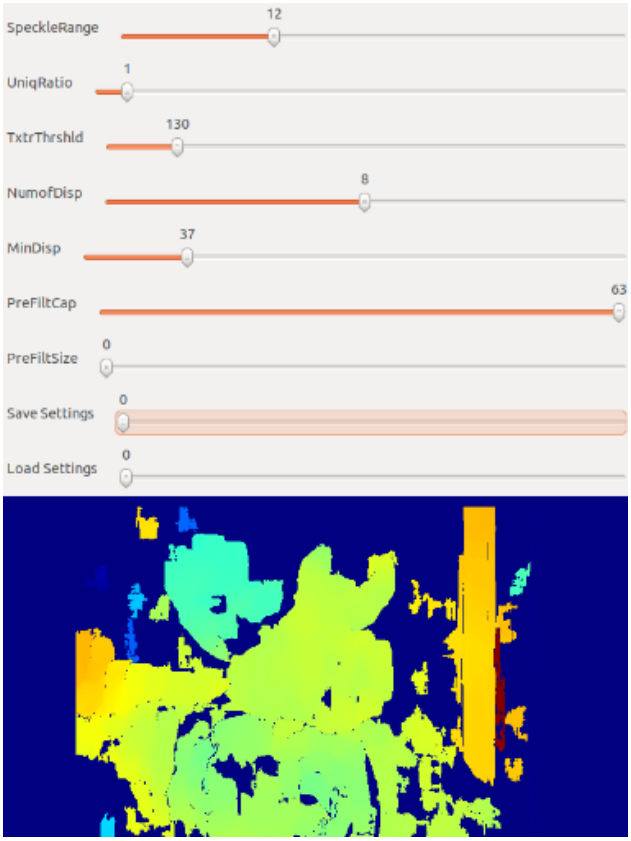

Рисунок 6 – Подбор параметров для получения корректной карты глубины

2.5 Тестирование и валидация

Так выглядит запущенное приложение для построения карты глубины:

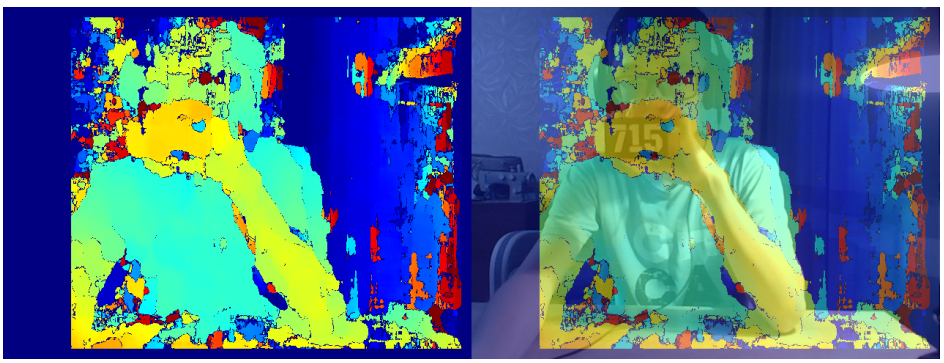

Рисунок 7 – Результат работы устройства

2.6 Анализ результатов

Решение на моей стереокамере вполне позволяет различать объекты на расстоянии от пяти сантиметров до трёх метров. Для промышленного робота, который будет курсировать по цехам производства, этого вполне достаточно. Для робота-уборщика этого вполне достаточно. Данное устройство

обойдётся примерно в десять раз ниже по цене, чем покупка лидара. Однако решение на стереокамере будет полезно только в том случае, если допустимо пренебречь качественным изображением карты глубины, поэтому выбор об использовании той или иной технологии нужно делать с умом.

3 Текущее состояние проекта и планы по дальнейшему развитию

На данном этапе разработка основного функционала завершена и устройство можно использовать для решения практических задач. Тем не менее есть некоторые улучшения, которые хотелось бы добавить позже: жёсткая фиксация физических параметров камеры (например, фокусное расстояние), получение конкретного расстояния до объекта, а также автоматический подбор параметров для любой стереокамеры.

Помимо программных компонентов, в условиях жёсткого санкционного давления, хотелось бы портировать данное решение на российскую электронную базу. Таким решением вполне может стать МВ77.07 от компании НТЦ «Модуль». Данная плата является микрокомпьютером, так как имеет свой центральный процессор, который в свою очередь поддерживает векторные операции и, предположительно, должен достаточно хорошо работать с матрицами.

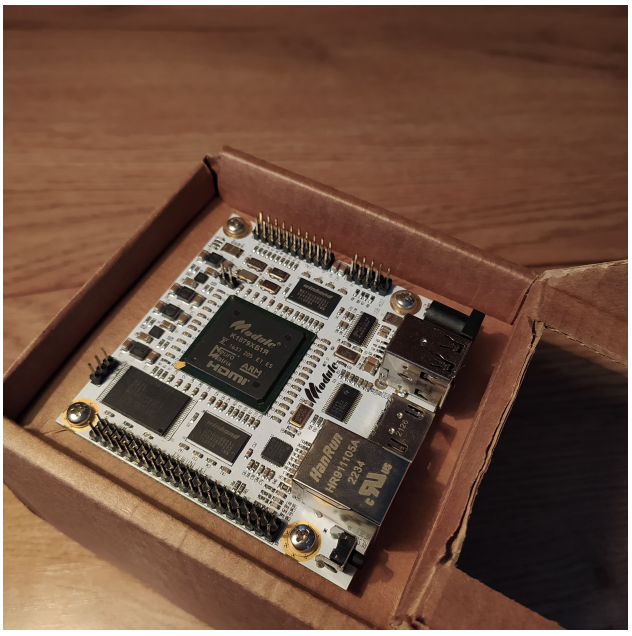

Рисунок 8 – Микрокомпьютер МВ77.07

Однако портирование моего программного продукта под данный процессор подразумевает переписывание всех библиотек на C/C++, что выходит далеко за рамки курсовой работы, поэтому оставляю эту тему как задел на будущее.

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта были успешно изучены основы эпиполярной геометрии, что позволило глубже понять принципы работы стереокамер и их применения в задачах компьютерного зрения. Я проанализировал ключевые концепции, такие как эпиполя, фундаментальная матрица и соответствие точек, что стало основой для реализации нашего устройства.

Разработка системы на основе стереокамеры для построения карты глубины продемонстрировала эффективность применения полученных знаний на практике. В результате были реализованы алгоритмы, позволяющие точно определять расстояние до объектов в сцене и визуализировать трехмерную структуру окружающей среды. Проведенные эксперименты подтвердили работоспособность системы и её способность генерировать качественные карты глубины в реальном времени.

Полученные результаты открывают перспективы для дальнейших исследований и усовершенствований, таких как улучшение алгоритмов обработки изображений, интеграция с другими сенсорами и применение в различных областях, включая робототехнику, автономные системы и дополненную реальность. Я уверен, что данный проект стал важным шагом в моём понимании стереозрения и его практических приложений, и надеюсь на дальнейшее развитие в этой захватывающей области.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Alexey Kurakin. Основы стереозрения, 13.10.2011. URL https://habr.com/ru/articles/130300/.

Kaustubh Sadekar. Introduction to epipolar geometry and stereo vision, 28.12.2020. URL https://learnopencv.com/introduction-to-epipolar-geometry-and-stereo-vision/.

honyaki. Машинное стереозрение для новичков: две камеры raspberry pi и python, 21.02.2022. URL https://habr.com/ru/companies/skillfactory/articles/652599/.

ПРИЛОЖЕНИЕ А

```python
#!/usr/bin/env python3

import os
import threading

import cv2
import numpy as np


class Capture:
    def __init__(self, grabbed: bool, frame: np.ndarray) -> None:
        self.grabbed: bool = grabbed
        self.frame: np.ndarray = frame.copy()


class StereoCamera:
    def __init__(
            self,
            sensor_id: int,
            frame_width: int,
            frame_height: int) -> None:

        self.sensor_id: int = sensor_id
        self.frame_width: int = frame_width
        self.frame_height: int = frame_height

        self.video_capture = cv2.VideoCapture(self.sensor_id)
        self.video_capture.set(cv2.CAP_PROP_FRAME_WIDTH, frame_width)
        self.video_capture.set(cv2.CAP_PROP_FRAME_HEIGHT, frame_height)

        grabbed, frame = self.video_capture.read()
        self.capture = Capture(grabbed, frame)

        self.read_thread: threading.Thread
        self.read_lock: threading.Lock = threading.Lock()

        self.running: bool = False

    def start(self) -> None:
        if self.running:
```

```
            print("[+] Video capturing already running!")
            return

        if self.video_capture:
            self.running = True
            self.read_thread = \
                threading.Thread(
                    target=self._update_camera,
                    daemon=True)
            self.read_thread.start()

    def stop(self) -> None:
        self.running = False
        self.read_thread.join()

    def _update_camera(self) -> None:
        while self.running:
            try:
                grabbed, frame = self.video_capture.read()
                with self.read_lock:
                    self.capture.grabbed = grabbed
                    self.capture.frame = frame
            except RuntimeError:
                print("[-] Could not read image from camera!")

    def read(self) -> Capture:
        with self.read_lock:
            grabbed = self.capture.grabbed
            frame = self.capture.frame.copy()
        return Capture(grabbed, frame)

    def release(self) -> None:
        if self.video_capture:
            self.video_capture.release()

        if self.read_thread:
            self.read_thread.join()


if __name__ == "__main__":
    os.environ["QT_LOGGING_RULES"] = "*.debug=false;qt.qpa.*=false"

    sensor_id = int(input("[>] Enter sensor ID: "))
```

```
    frame_width = 1280
    frame_height = 480

    camera = StereoCamera(sensor_id, frame_width, frame_height)
    camera.start()

    while True:
        capture = camera.read()
        grabbed = capture.grabbed
        frame = capture.frame

        if grabbed:
            cv2.imshow("[+] Camera image:", frame)

        if cv2.waitKey(1) == ord("q"):
            break

    camera.stop()
    camera.release()
    cv2.destroyAllWindows()
```

```
#!/usr/bin/env python3

import os

import cv2
import numpy as np
from stereovision.calibration import StereoCalibration

from camera import StereoCamera

SWS = 29
PFS = 5
PFC = 20
MDS = 0
NOD = 80
TTH = 500
UR = 1
SR = 0
SPWS = 0


def load_map_settings():
```

```
    global SWS, PFS, PFC, MDS, NOD, TTH, UR, SR, SPWS, \
        loading_settings, sbm
    sbm = cv2.StereoBM().create(numDisparities=16, blockSize=SWS)
    sbm.setPreFilterType(1)
    sbm.setPreFilterSize(PFS)
    sbm.setPreFilterCap(PFC)
    sbm.setMinDisparity(MDS)
    sbm.setNumDisparities(NOD)
    sbm.setTextureThreshold(TTH)
    sbm.setUniquenessRatio(UR)
    sbm.setSpeckleRange(SR)
    sbm.setSpeckleWindowSize(SPWS)


def stereo_depth_map(rectified_pair):
    dmLeft = rectified_pair[0]
    dmRight = rectified_pair[1]
    disparity = sbm.compute(dmLeft, dmRight)
    disparity_normalized = cv2.normalize(
        disparity,
        disparity,
        0, 255, cv2.NORM_MINMAX)
    image = np.array(disparity_normalized, dtype=np.uint8)
    disparity_color = cv2.applyColorMap(image, cv2.COLORMAP_JET)
    return disparity_color, disparity_normalized


if __name__ == "__main__":
    os.environ["QT_LOGGING_RULES"] = "*.debug=false;qt.qpa.*=false"

    # sensor_id = int(input("[>] Enter sensor ID: "))
    # frame_width = int(input("[>] Enter frame width: "))
    # frame_height = int(input("[>] Enter frame height: "))

    sensor_id = 2
    frame_width = 1280
    frame_height = 480

    camera = StereoCamera(sensor_id, frame_width, frame_height)
    camera.start()
    load_map_settings()

    cv2.namedWindow("Depthmap")
```

```
while True:
    capture = camera.read()
    grabbed = capture.grabbed
    frame = capture.frame

    if grabbed:
        left_frame = frame[:, : frame_width // 2]
        right_frame = frame[:, frame_width // 2 :]

        left_gray_frame = cv2.cvtColor(
            left_frame, cv2.COLOR_BGR2GRAY)
        right_gray_frame = cv2.cvtColor(
            right_frame, cv2.COLOR_BGR2GRAY)

        calibration_result_folder_path = "calibration_result"
        calibration = StereoCalibration(
            input_folder=calibration_result_folder_path)
        rectified_pair = calibration.rectify(
            (left_gray_frame, right_gray_frame))
        disparity_color, disparity_normalized = \
            stereo_depth_map(rectified_pair)

        output = cv2.addWeighted(
            left_frame, 0.5, disparity_color, 0.5, 0.0)
        cv2.imshow("DepthMap", np.hstack((disparity_color, output)))

        if cv2.waitKey(1) == ord("q"):
            break
        else:
            continue

camera.stop()
camera.release()
cv2.destroyAllWindows()
```