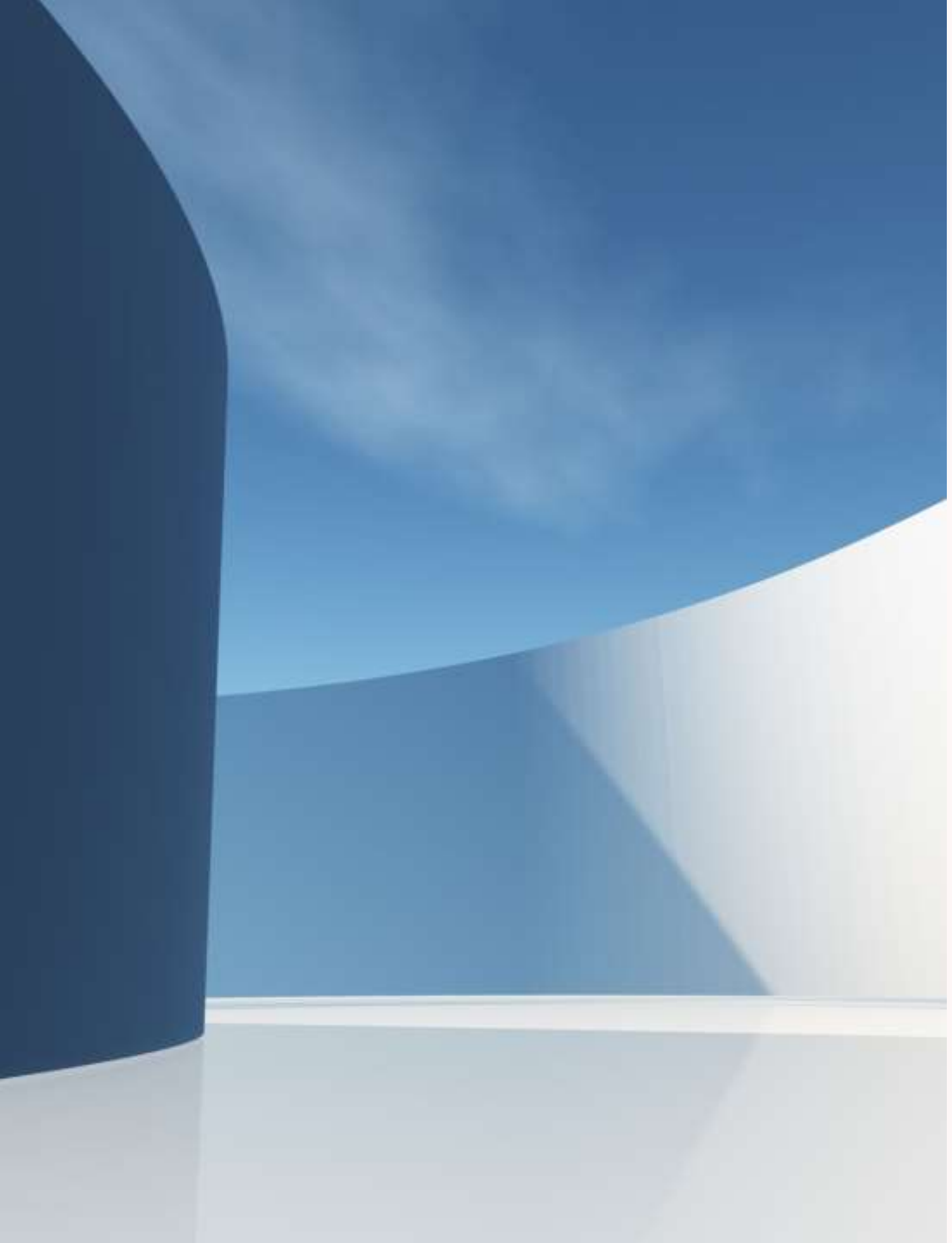


Automated Detection of Tactile Pavings Roads

- Hüseyin Emre Sekanlı
- 200104004049
- Digital Image Processing Project

- This project presents an innovative approach to detect tactile paving using video analysis in Python, with a focus on enhancing urban accessibility.
- Utilizing the Tkinter library for creating a user interface, the project incorporates advanced image processing techniques through OpenCV.
- The core functionality lies in the update_frame method, where frames from uploaded videos are processed to identify tactile paving.
- This process involves color space transformations, brightness and contrast adjustments, edge detection, and application of various image processing techniques such as CLAHE, bilateral filtering, and Gaussian blurring.
- The system aims to accurately identify tactile paving in urban environments, contributing to improved navigation for visually impaired individuals.
- This report details the development process, challenges encountered, and the effectiveness of the implemented methods in detecting tactile paving through real-time video analysis.

THE TABLE OF CONTENTS

Introduction

Tactile paving, essential for urban accessibility, aids visually impaired individuals in safely navigating public spaces. These textured ground surfaces, commonly found in pedestrian areas, serve as physical cues for potential hazards or changes in surroundings. The project's focus is on utilizing Python for the automated detection of such paving in video footage, enhancing the understanding and interaction of visually impaired individuals with their environment. This integration of technology and urban design represents a significant advancement in creating inclusive, accessible public spaces.

Project Objective

- The primary aim of this project is to develop a Python-based solution for automatically detecting tactile paving in video footage. This system is designed to analyze video data, identify tactile paving patterns, and potentially map these features in urban environments. The objectives include:

- Developing an Efficient Detection Algorithm: Crafting a reliable algorithm capable of accurately identifying tactile paving patterns in various lighting and weather conditions.

- Utilizing Video Analysis for Real-time Processing: Implementing real-time analysis of video data to ensure prompt and efficient detection of tactile paving.

- Enhancing Urban Accessibility: Contributing to the broader goal of urban accessibility by providing a tool that can assist in the planning and improvement of navigational aids for visually impaired individuals.

- The project endeavors to bridge the gap between technological advancements and their practical application in enhancing the quality of life for visually impaired individuals, promoting a more inclusive approach to urban design and accessibility.

Scope

- Python and OpenCV Utilization: Python, with its powerful libraries such as OpenCV, forms the backbone of the development process. The project leverages these tools for image processing tasks essential in detecting tactile paving patterns.

- Geographical and Environmental Considerations: While the algorithm aims to be versatile, its effectiveness is initially tested and demonstrated in specific urban settings. Factors like lighting, weather conditions, and pavement types in these areas are considered.

- Technical Limitations: The Project aims to give real-time result once the video is uploaded.This results in slow performance.

- Target Audience: The primary beneficiaries of this project are urban planners and organizations focusing on improving navigational aids for visually impaired individuals. The tool aims to assist in the assessment and planning of more accessible urban environments.

- Outcome Expectations: While the project aims for high accuracy in tactile paving detection, it recognizes potential limitations in varying urban scenarios and seeks to continuously evolve based on testing and feedback.

Tools and Technologies

- Python: The primary programming language used for developing the detection algorithm.
- OpenCV (Open Source Computer Vision Library): Employed for image processing and computer vision tasks.
- Tkinter: Utilized for creating a graphical user interface (GUI) for video upload and playback.
- DBSCAN:Used to create a mask after edge detection.

Data Collection and Preparation

- Video Acquisition: Videos are sourced from urban environments, showcasing various pedestrian areas where tactile paving is present.
- Format and Quality Considerations: Videos in different qualities are considered to test the robustness of the algorithm.

Algorithm Development and Implementation

- Detection Algorithm: Development of an algorithm to process video frames and detect tactile paving. This includes:
- Preprocessing: Techniques like color space transformation, brightness, and contrast adjustments.
- Feature Extraction: Identifying key features indicative of tactile paving.
- Pattern Recognition: Using machine learning or heuristic methods to recognize tactile paving patterns.
- Real-Time Analysis: Ensuring the algorithm can process video footage in real-time.

GUI Overview

The code features a Graphical User Interface (GUI) implemented using the tkinter library. The GUI allows users to interact with video files, play, pause, and replay them while displaying various video processing techniques in real-time. Here's an overview of the GUI components:

- Canvas: A canvas element is provided within the GUI to display the video frames and processed frames.

- Control Buttons:
  - "Upload Video": Allows the user to select a video file to be processed.
  - "Play": Initiates the video playback.
  - "Pause": Pauses the video playback.
  - "Replay": Restarts the video from the beginning.

- Time Labels:
  - "Total Time": Displays the total duration of the video.
  - "Current Time": Shows the current time of the video during playback.

Video Preprocessing:

In the update_frame function, the code performs a series of preprocessing steps on the video frames before displaying them on the GUI canvas. These preprocessing steps are designed to enhance the visual quality and extract specific features from the video. Below are the preprocessing methods, listed in the order in which they are applied:

- Histogram Equalization (CLAHE):

This technique enhances the contrast of the video frames using Contrast Limited Adaptive Histogram Equalization (CLAHE).

- Bilateral Filtering:

Bilateral filtering is applied to reduce noise while preserving important edges in the video frames.

- Color Range Isolation:

The code isolates specific color ranges from the video frames by thresholding in the HSV color space.

- Gaussian Blur:

Gaussian blurring is performed to further reduce noise and smoothen the video frames.

- Edge Detection:

Edge detection techniques are used to detect edges and contours in the video frames.

- Road Border Detection:

The code identifies and creates a mask for the road borders in the video frames by applying Hough Line Transform and DBSCAN clustering.
It detects the lines that could form a road , uses it as a mask to lose the remaining edges after edge detection.

- Additional Features:

The code also calculates and displays average brightness and contrast values on the processed frames.
Each of these preprocessing methods contributes to improving the visual quality of the video frames and extracting relevant information for further analysis.This step needs improvement and more testing to better capture the lighting conditions.

By incorporating these preprocessing techniques, the code enhances the overall viewing experience and assists in extracting specific information from the video content.

Results

ORIGINAL IMAGE

AFTER CLAHE

Results

BILATERAL

COLOR ISOLATION

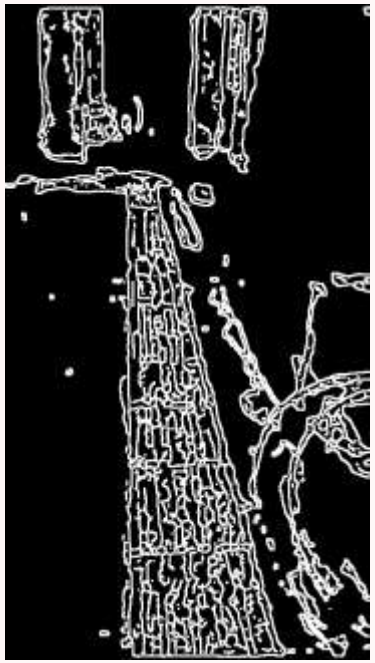
Results

GAUSSIAN BLUR

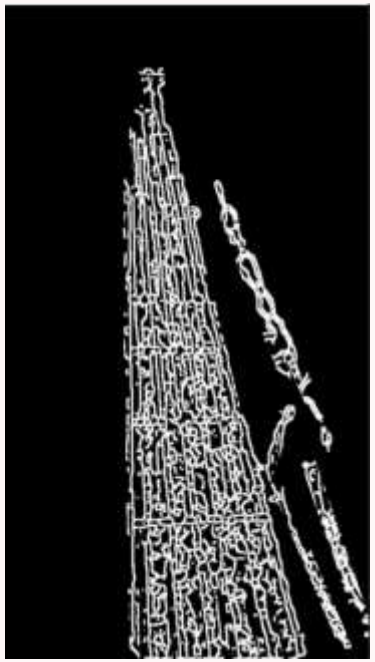
REMOVE NOISE

Results

EDGES



ROAD



Interpretation of Results:

- The preprocessing techniques applied to the video frames successfully enhanced visual quality, reduced noise, and isolated specific color ranges.
- Road border detection using Hough Line Transform and DBSCAN clustering helped identify and mask road boundaries effectively.
- Average brightness and contrast values were calculated and displayed on the processed frames, providing additional insights.

Challenges Faced and Solutions:

- Noise Reduction: Challenge: Noise in video frames. Solution: Bilateral filtering and morphological operations for noise reduction.

- Color Range Isolation: Challenge: Accurate isolation of specific color ranges. Solution: Dynamic HSV range adjustment based on brightness and contrast.

- Road Border Detection: Challenge: Detecting road borders reliably. Solution: Clustering and filtering of Hough Line Transform results for robust road border detection.

Interpretation of Results:

- Road Border Detection: Challenge: Detecting road borders reliably. Solution: Clustering and filtering of Hough Line Transform results and using mask to lose the non-road parts.

Limitations of the Approach:

- Robustness: The approach may not perform well under varying lighting conditions or when the road appearance changes significantly.

- Parameter Tuning: The effectiveness of some preprocessing techniques relies on parameter tuning, which may require adjustments for different video content.
- This is the most troublesome part.Too many test-runs required for varying conditions. If i give too much tolerance on parameters,it starts to include non-road parts,if i lower the tolerance some of the road data may be lost.

- Real-time Processing: The current implementation may not be too suitable for real-time processing of high-resolution videos due to computational demands.

- Road Variability: The approach assumes consistent road appearance and may not handle diverse road conditions (e.g., dirt roads, different road markings) effectively.

Conclusion:

My video processing application effectively enhanced video quality through preprocessing techniques.
Road border detection, achieved via Hough Line Transform and DBSCAN clustering, successfully segmented roads.
Additional insights, such as brightness and contrast data, were extracted and displayed.
There is still room for improvement, but this should suffice at variety-cases.

Implications & Potential Future Work:

- Optimization: Fine-tuning preprocessing parameters to adapt to various video conditions and make it more suitable for real-time application.
- Machine Learning Integration: Exploring integration with machine learning models for object detection and tracking.
- Real-time Efficiency: Enhancing real-time video processing capabilities for efficiency.
- Complex Road Scenarios: Expanding the system's capabilities to handle diverse road conditions, road markings, and road types for broader applicability.