Final Report

TED ÜNİVERSİTESİ

Project Title:

3DifyMe

Team Members:

Bengisu Tuğrul, Hamze Emin Hacıoğlu, Elif Arslan, Deniz Caner Akdeniz

Course Name and Code:

CMPE 492

Supervisor:

Gökçe Nur Yılmaz

Jury Members:

Müslim Bozyiğit, Tolga Kurtuluş Çapın

Submission Date:

27.12.2024

Table of Contents

1. Introduction

The 3DifyMe project is focused on creating a system that processes visual data and enhances it for different uses. The main goal is to work on improving image quality and creating 3D effects from 2D content. By dealing with challenges like processing speed, accuracy, and scalability, the project aims to prepare for real-time applications like live video and camera feeds.

Right now, the project is concentrating on processing live video, static videos, and static images. This approach allows the system to be tested and improved in various scenarios, ensuring its flexibility and reliability. A key feature of the project is creating anaglyph images, which combine two slightly offset versions of an image to simulate a 3D effect when viewed with red-cyan glasses. This process relies heavily on generating accurate depth maps to determine the relative distances of objects in the scene.

3DifyMe can be useful in different areas. For example, in cinema, it could make creating 3D content easier and more accessible. It could also be helpful in industries like automated quality control by analyzing images to spot defects or improve production quality.

The project also looks at important technical aspects like depth map generation, reducing noise in images, and using AI to enhance visuals. These features are important for achieving high-quality results. For instance, using tools like “MiDaS” pretrained artificial intelligence model for depth maps shows the focus on using advanced technologies to achieve goals. Depth maps are essential because they provide information about the distance of objects in an image, which is crucial for creating convincing 3D effects and improving overall image quality.

We have designed 3DifyMe with the aim of helping users achieve high-quality 3D effects and enhanced visuals in an accessible way. By focusing on both simplicity and flexibility, the system caters to a wide range of needs, from casual users interested in 3D imagery to professionals working in cinema or industrial applications. Each component has been developed with usability and scalability in mind, ensuring that the platform can grow to meet future demands. Our goal is to allow users to transform their visual data into immersive 3D experiences with ease.

2. Final Architecture and Design

2.1. Image Processing Architecture and Design

2.1.1 Image Processing Pipeline

In the image processing component of 3DifyMe, the primary goal is to create a stereoscopic version of the input image. Stereoscopic images offer a 3D visual experience by presenting two slightly offset images to each eye, mimicking the way humans perceive depth. Traditional methods for achieving this effect often involve using dual cameras positioned at different angles or capturing two images from varying viewpoints. While effective, these approaches can be cumbersome, requiring additional hardware or precise alignment.

Our approach leverages advanced image processing techniques and machine learning models to eliminate the need for such hardware. The solution focuses on generating an anaglyph version of the image, which creates a 3D effect when viewed with red-cyan glasses. Anaglyph images combine two color channels (typically red and cyan) from slightly different perspectives, simulating depth perception. Additionally, 3DifyMe incorporates several enhancement tools, including contrast adjustment, noise handling, and image upscaling, to provide users with customizable options for improving their stereoscopic images.

2.1.2 Anaglyph Generation

The process of creating an anaglyphic image begins with generating a depth map, a grayscale representation of the scene where pixel values indicate relative distances. Two methods are available for depth map generation in 3DifyMe:

1. **Pretrained Model (DPT-Large)**: This robust model, developed by Intel, has been trained on extensive datasets to produce high-quality depth maps. It is reliable and performs well across diverse scenarios.
2. **Custom Model**: Our custom depth map generation model is based on a U-Net encoder architecture with a ResNet-34 backbone and additional custom layers. This design offers flexibility and adaptability, allowing users to experiment with configurations tailored to their specific needs.

The depth map generation process involves the following steps:

- Preprocessing the input image using a feature extractor.
- Generating the depth map with the selected model.
- Normalizing and resizing the depth map to match the original image dimensions.
- Enhancing the depth map with bilateral filtering to preserve edges while smoothing depth values.

Finally, the anaglyph is created by shifting pixels in the left and right images based on the depth map values, combining them into a red-cyan format. The resulting image provides a realistic 3D effect when viewed with anaglyph glasses.

2.1.3 Image Enhancement Tools

2.1.3.1 Contrast Adjustment

Enhancing contrast improves image quality by modifying pixel intensity values. Three methods are implemented in 3DifyMe:

- **Gamma Correction**: Adjusts image brightness by altering gamma values. Increasing gamma brightens the image, while decreasing it darkens the image. This method is ideal for balancing overly bright or dark images.
- **Histogram Equalization**: Redistributes intensity values to enhance contrast. It is particularly effective for underexposed or overexposed images, highlighting details and improving visual appeal.
- **Convert Scale Abs**: Allows precise adjustment of brightness and contrast using alpha (contrast) and beta (brightness) values. This method is highly customizable and suitable for various image types.

2.1.3.2 Noise Handling

Noise reduction ensures image clarity by addressing random pixel variations. Two noise-handling techniques are included:

- **Bilateral Filtering**: Removes noise while preserving edges, making it ideal for images with intricate details and textures.
- **Wavelet Denoising**: Uses wavelet transformations to reduce noise selectively across frequency components. This method is effective for complex images with varying noise levels, ensuring high-quality outputs.

2.1.3.3 Upscaling

Upscaling enlarges an image without degrading quality. 3DifyMe uses the "LapSRN" model to provide three upscaling options:

- **2x Upscaling**: Doubles the image size, suitable for moderate enhancements.
- **4x Upscaling**: Quadruples the image size, ideal for detailed viewing or printing.
- **8x Upscaling**: Increases the image size eightfold, ensuring the highest possible resolution for extreme magnification needs.

The process involves initializing the super-resolution model, converting the image to RGB format, and applying upscaling techniques. This approach balances computational efficiency with output quality, allowing users to achieve exceptional results.

2.1.4 Diagram

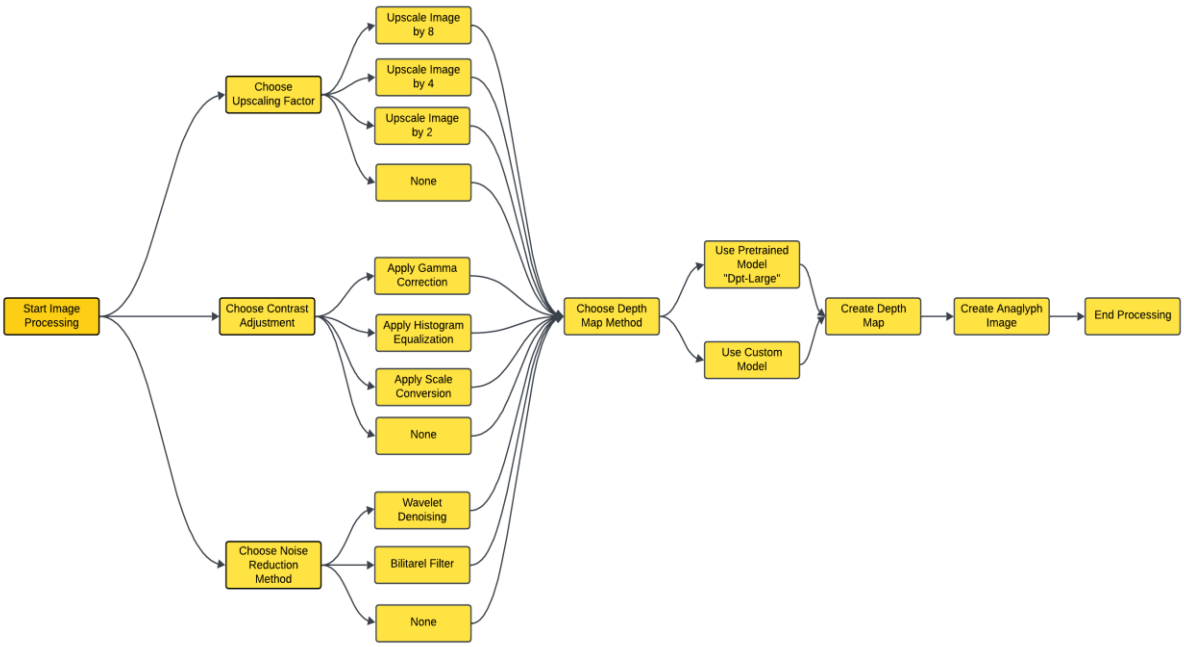

Figure: Diagram of image processing part of 3DifyMe application.

2.2. Video and Live Processing Architecture and Design

The system implements a real-time video and image processing pipeline with anaglyph 3D effect generation. The architecture is built on OpenCV for video processing, featuring both live camera feed processing and video file handling capabilities.

2.2.1. Video Processing Pipeline

The system utilizes two primary video processing paths:

- Live camera feed processing (implemented via update_frame())
- Video file processing (implemented via update_video_frame())

Both paths converge into a common processing pipeline that generates anaglyph 3D effects.

Method: update_video_frame(self)

- Fetches the next video frame using cv2.VideoCapture.read().
- Resizes the frame to a standard resolution of 640x480 pixels.
- Calls process_and_display_image to process and display the frame.

2.2.2. Performance Optimizations

Pre-calculation Strategy

The system implements performance optimizations through the setup_optimization_maps() method:

- Pre-calculated coordinate matrices
- Pre-computed shift maps
- Pre-defined Sobel kernels for edge detection
- Look-up table (LUT) for color intensity adjustments

Timer Configuration

Camera Timer: Set to ~60 FPS (16ms intervals)

Video Timer: Matched to camera timing for consistent performance

2.2.3. Anaglyph Generation Process

The create_anaglyph() method implements a processing pipeline:

Depth Map Generation

- Grayscale conversion
- Gaussian blur application (3x3 kernel)
- Gradient calculation using pre-calculated Sobel kernels
- Depth map normalization

Channel Processing

- Channel separation (BGR)
- Depth-based shift calculation
- Independent channel manipulation for 3D effect

Color Enhancement

- LUT-based color intensity adjustment
- Brightness factor application

Method: create_anaglyph(self, frame)

- Converts the frame to grayscale for depth analysis.
- Applies Gaussian blur to reduce noise.
- Calculates gradients (grad_x, grad_y) using precomputed Sobel kernels to determine edge intensities.
- Combines the gradients into a depth map using the Euclidean norm.
- Shifts color channels (red, green, blue) based on the depth map to simulate the anaglyph effect.
- Enhances the processed image using a precomputed Look-Up Table (LUT) for color intensity and brightness adjustment.

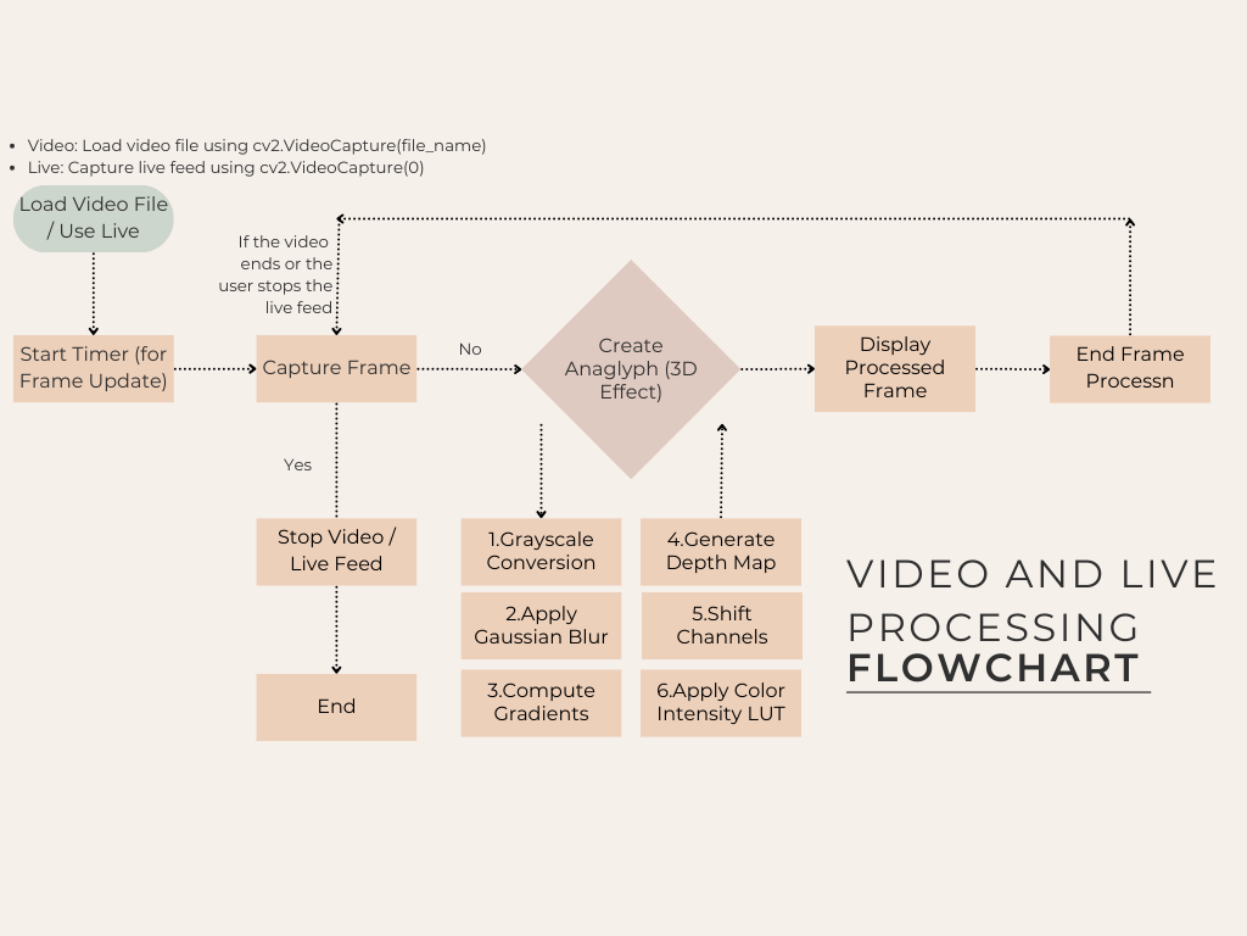

Figure: Video and Live Processing Flowchart

2.2.4. Performance Features

Memory Management

- Efficient memory usage through numpy's vectorized operations
- Pre-allocated arrays for coordinate mapping
- Optimized channel processing without redundant calculations

Processing Optimizations

- Vectorized operations for bulk pixel processing
- Look-up tables for color transformations
- Pre-calculated maps for coordinate transformations
- Minimal memory allocation during processing

2.2.5. Architecture Implementation Details

Video Input Handling

The system supports two primary sources for video input: camera and video files. For camera input, we utilize the OpenCV function cv2.VideoCapture(0, cv2.CAP_DSHOW), which allows us to access the default camera on the device. Additionally, the system can process video files in various formats, including *.mp4, *.avi, *.mov, and *.mkv. The standard resolution for both camera and file inputs is set to 640x480 pixels, ensuring consistency across different sources.

Feature	**Details**
Camera Input	cv2.VideoCapture(0, cv2.CAP_DSHOW)
Video File Formats Supported	*.mp4, *.avi, *.mov, *.mkv
Standard Resolution	640x480 pixels

Real-time Processing Parameters

To enhance the quality of video processing in real-time, several parameters are adjustable. The shift amount is set to 15 pixels, allowing for minor adjustments in frame alignment. The color intensity is amplified by a factor of 1.2x to enrich the visual output, while the brightness factor is reduced to 0.8x to prevent overexposure and maintain detail in brighter areas of the frame.

Parameter	**Value**
Shift Amount	15 pixels
Color Intensity	1.2x
Brightness Factor	0.8x

Frame Processing

The input format for frames is BGR (the default format used by OpenCV), which is subsequently converted to RGB for display purposes. The processing resolution remains at 640x480 pixels to maintain alignment with the input resolution. Furthermore, the depth map resolution matches that of the input frame, facilitating accurate depth perception during processing.

Aspect	**Specifications**
Input Format	BGR (OpenCV default)
Processing Resolution	640x480
Color Space Conversion	BGR → RGB for display
Depth Map Resolution	Same as input frame

Performance Considerations

Achieving a target frame rate of 60 frames per second (FPS) is essential for smooth video playback and real-time analysis. To meet this performance goal, several optimizations are implemented:

- Vectorized Numpy Operations: These operations allow for efficient processing of pixel data by leveraging optimized numerical libraries.
- Pre-calculated Transformation Maps: By calculating transformation maps in advance, we reduce computational overhead during frame processing.
- Optimized Memory Usage: Efficient memory management ensures that resources are utilized effectively, minimizing latency.
- Efficient Channel Manipulation: This technique allows for quick adjustments to color channels without significant processing delays.

2.3 User Interface Architecture and Design

2.3.1 Integration and Workflow

The integration process ensures seamless interaction between the core processing modules and the user interface (UI), delivering a cohesive and user-friendly experience. This design prioritizes accessibility, usability, and adaptability to accommodate varying user needs and processing tasks. It ensures that both novice and advanced users can effectively engage with the platform's features, offering clarity, guidance, and efficiency in every step. Inspired by the provided diagram and layout concepts, visual interactivity, dynamic pathways, and modularity are further emphasized to refine the workflow and provide a comprehensive user experience.

2.3.1.1 Image Workflow

- **File Selection**: The user selects an image file from the directory. The interface provides intuitive file browsing options, ensuring that users can easily locate and upload their desired images.
- **Validation**: If the image format is invalid, the user is redirected to file selection with a warning message that clearly explains the issue and suggests supported formats. This ensures minimal confusion and a smooth experience.
- **Tool Selection**: Once an image file is validated, users are presented with an array of processing tools, such as depth mapping, noise reduction, and upscaling. Interactive previews allow users to experiment with and confirm tool settings before proceeding.
- **Pipeline Creation**: A dedicated processing pipeline is created based on the selected tools. Users are guided through the pipeline visually, with indicators showing progress and tool execution status.

- **Output Options**: After processing, users can choose to save the enhanced image, exit the session, or rerun the pipeline for further adjustments. Clear save options and file format customization enhance flexibility.

2.3.1.2 Video Workflow

- **File Selection**: Similar to the image workflow, users select a video file via an intuitive browsing interface. The system supports various file types, ensuring broad compatibility.
- **Validation**: Unsupported formats trigger detailed warnings, redirecting users to reselect their files with proper guidance on supported video formats. This minimizes errors and enhances usability.
- **Pipeline Creation**: Once validated, the selected video is added to the processing pipeline. Users can configure the pipeline to apply multiple tools, such as frame-by-frame enhancements, depth adjustments, and noise reduction.
- **Output Options**: After processing, users are given options to save the output video, rerun specific processing stages, or exit. Advanced options, such as saving in different resolutions or formats, cater to diverse user needs.

2.3.1.3 Live Feed Workflow

- **Camera Connection**: The system checks for an active camera feed. A device selection menu allows users to switch between multiple connected cameras seamlessly.
- **Warnings**: Alerts are displayed if no active camera is detected, with clear instructions to resolve the issue or select an alternative input source.
- **Pipeline Creation**: Once a camera is detected, the live feed is integrated into the processing pipeline. Frame-by-frame analysis enables real-time application of depth mapping, noise reduction, and other tools.
- **Real-Time Output**: Processed frames are displayed live with continuous updates. Users can toggle between processed and original views for comparison, enhancing the interactive experience. Additional options allow for pausing the feed, applying adjustments, and resuming processing.

2.3.1.4 Error Handling and Validation

- **Comprehensive Validation**: The system emphasizes error handling at each stage, from file uploads to live feed connectivity. Detailed error messages guide users toward corrective actions, ensuring a smooth and frustration-free experience.
- **User Redirection**: Clear warnings and redirection paths ensure that users can efficiently resolve issues, whether selecting a different file format or adjusting tool configurations.

2.3.1.5 Dynamic Processing Paths

- **Tailored Workflows**: The workflows for images, videos, and live feeds are modular and tailored, ensuring specific tools and processes are applied to each input type without overlap. This modularity allows for a streamlined and focused user experience.
- **Flexible Adjustments**: Conditional paths enable users to rerun workflows, make tool adjustments, or exit at any stage, ensuring adaptability to changing requirements or preferences.

2.3.1.6 Integration with UI Design

- **Input Selection**: Buttons for "Image," "Video," and "Live" input types are prominently displayed, reflecting diagram-based branching. These options guide users intuitively toward their desired workflow.
- **Tool Options**: Interactive menus present users with available tools dynamically. Drop-downs, sliders, and toggles allow for fine-tuning parameters while providing instant feedback on changes.
- **Real-Time Feedback**: The interface supports live previews and status updates, aligning with pipeline creation and output stages. Users are constantly informed of the progress and results, fostering transparency and engagement.
- **Contextual Help**: Integrated help options offer real-time guidance tailored to the current stage of the workflow, reducing the learning curve for new users.

By leveraging modular workflows and emphasizing user-centric design principles, the system ensures clarity, adaptability, and ease of use for diverse processing needs.

2.3.2 Diagram

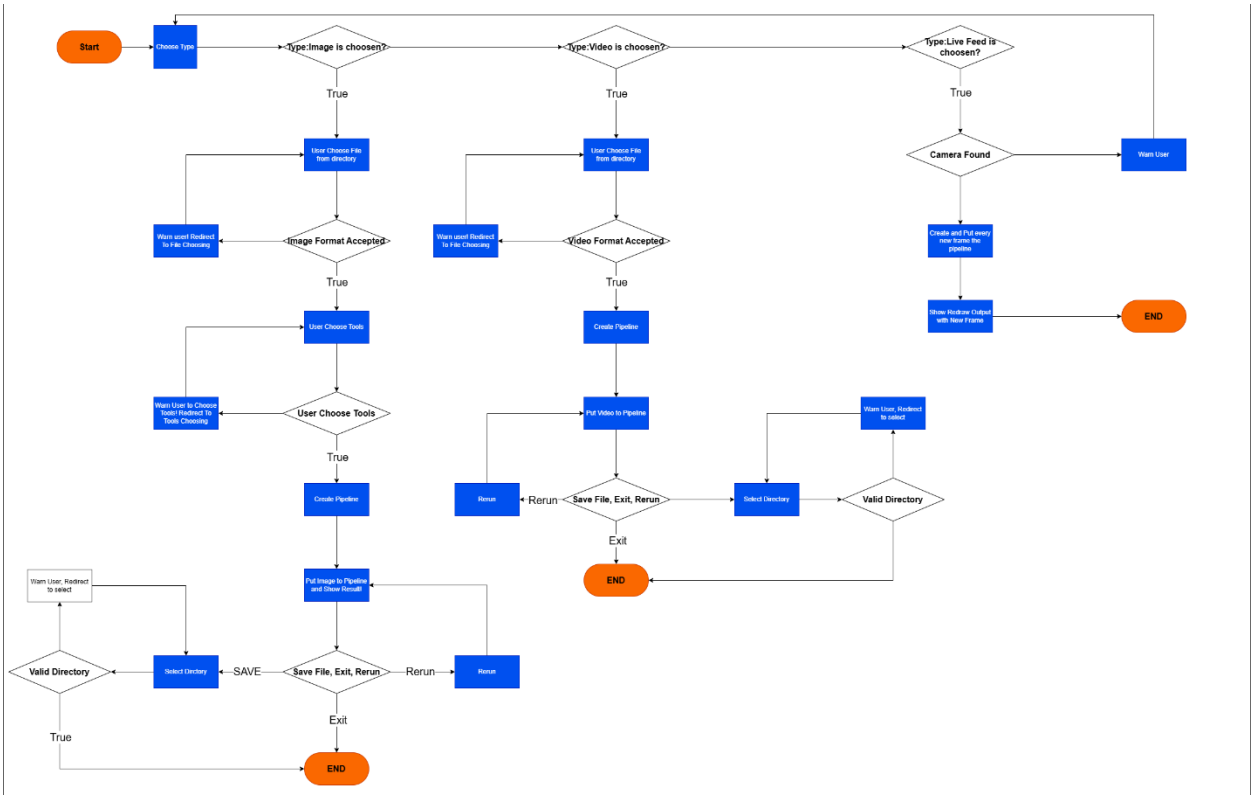

Figure: UI Workflow Diagram

2.3.3 Concept Art

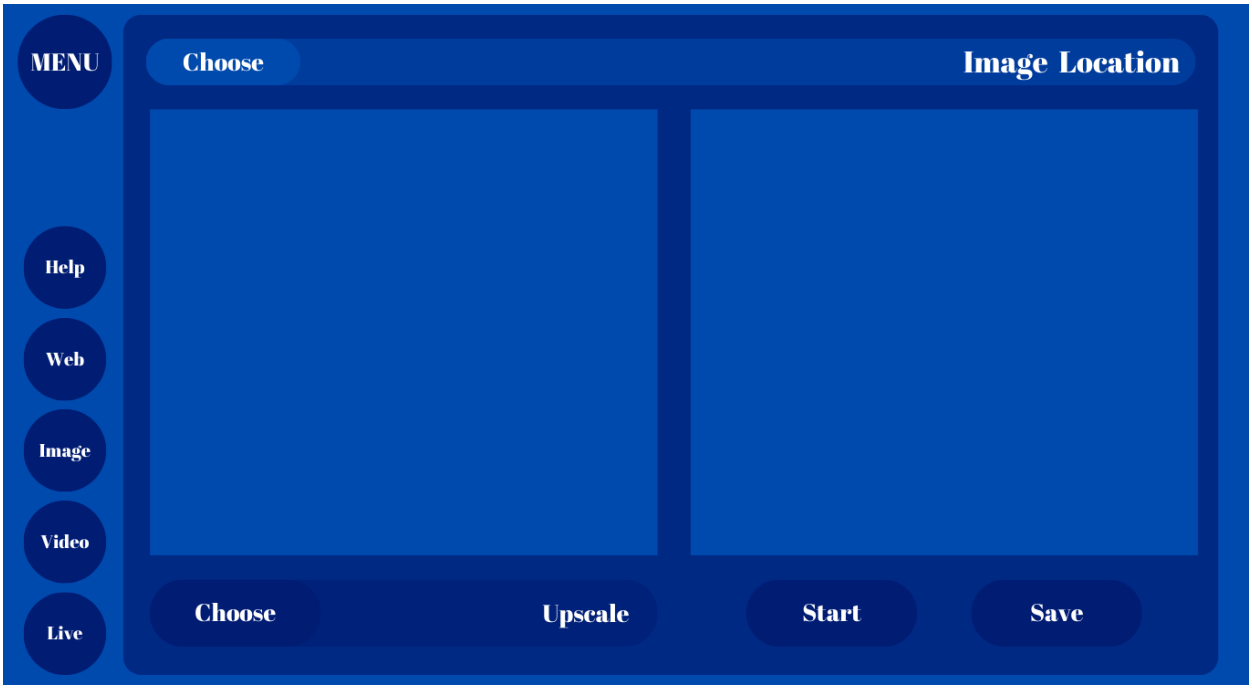

Figure: Concept Art

3. Impact of Engineering Solutions

Global Impact

While the current version of 3DifyMe may not significantly impact global industries, future developments hold promises for broader influence. The evolution of such a system could primarily affect the entertainment sector, enabling users to create 3D media for deeper and greater enjoyment. Over time, applications might extend to education and potentially security, fostering immersive experiences and new opportunities for interaction. However, its most immediate and impactful use case lies in enhancing entertainment, allowing users to transform their media consumption into richer, more engaging experiences.

Economic Impact

By automating complex image and video processing tasks, 3DifyMe reduces the need for expensive manual interventions and specialized equipment. This efficiency translates into cost savings for businesses while enabling smaller enterprises to adopt advanced media processing technologies. Furthermore, its ability to scale across various sectors can drive innovation and productivity, contributing to economic growth.

Societal Impact

The platform's accessibility and user-friendly design enable broader adoption across non-technical users, democratizing access to advanced media processing. This accessibility can lead to greater creativity and innovation, particularly in educational and artistic contexts. Additionally, its applications in public safety and surveillance can help improve societal security measures.

Environmental Impact

The computational demands of 3DifyMe raise concerns about energy consumption in resource-intensive tasks like live feed analysis and video upscaling. However, its limited usage scope and integration of energy-efficient algorithms help keep energy impact manageable. Detecting protected media for restricted processing and exploring cloud solutions further align the platform with sustainable practices.

Ethical Impact

The platform's ability to enhance and manipulate images raises ethical considerations, particularly regarding potential misuse for misinformation or altered media. Additionally, automation in 3DifyMe could reduce reliance on human labor in the entertainment sector, potentially affecting workforce demand. To address these concerns, safeguards like watermarking and detecting protected media to halt unauthorized processing can ensure media owners' rights are upheld. Protecting user data through encryption and secure storage further supports ethical standards.

4. Contemporary Issues

Computational Efficiency and Hardware Limitations

Modern image and video processing applications, like those employed in 3DifyMe, face challenges related to computational efficiency. While the system utilizes advanced algorithms for high-quality outputs, the reliance on current-generation CPUs without GPU support can result in performance bottlenecks. This limitation is particularly evident in tasks requiring real-time processing, such as live feed analysis and high-resolution video frame generation.

Scalability for Diverse User Needs

The adaptability of the system's architecture ensures its versatility across various use cases. However, the growing demand for more sophisticated features and support for higher resolutions or faster processing speeds may require further scaling of computational resources. This challenge highlights the need for ongoing optimization to balance resource consumption with performance.

Ethical Considerations in Image Processing

Ethical concerns remain paramount in the field of image and video processing. The potential misuse of stereoscopic enhancement tools for misleading visual representations or manipulated media poses significant risks. Addressing these issues requires implementing safeguards, such as watermarking or usage limitations, to maintain the integrity of processed outputs.

Accessibility and Inclusivity

While 3DifyMe aims to be user-friendly, accessibility remains a critical consideration. Ensuring that the system accommodates users with varying levels of expertise and differing hardware capabilities is essential. This includes providing simplified workflows for beginners while maintaining advanced configurations for experienced users. Furthermore, compatibility with lower-end devices is an ongoing challenge, especially for users in resource-constrained environments.

Environmental Impact of Computational Resources

The computational demands of image and video processing can have environmental implications, particularly due to high energy consumption associated with extended GPU or CPU usage. Exploring energy-efficient algorithms and leveraging cloud-based processing solutions could mitigate these impacts, aligning with sustainable development goals.

Privacy and Data Security

As users upload personal or sensitive images and videos for processing, safeguarding data privacy and security is crucial. Implementing robust encryption methods, anonymization techniques, and strict data handling policies is essential to protect user data and build trust.

By addressing these contemporary issues, 3DifyMe can not only remain relevant and effective but also ensure that it adheres to ethical, environmental, and user-centric principles in its ongoing development.

5. Tools and Technologies Used

- **Programming Languages: Python**
- **Libraries: PyTorch, torchvision, ResNet34 (as part of torchvision), U-Net architecture implementation, NYU-Depth V2 dataset utilities**
- **Development Environment: Python Scripts, Jupyter Notebook**

These tools and frameworks were selected for their capability to support machine learning model development, particularly in depth estimation tasks. PyTorch's flexibility and ResNet34's pre-trained weights were critical in creating a robust encoder-decoder architecture, while Jupyter Notebook provided an interactive environment for model experimentation and analysis. The NYU-Depth V2 dataset was chosen for its diverse range of indoor scenes, enabling effective training for depth map prediction.

5.1 Specification of Technologies Used on Image System

In this project, we developed a depth estimation model aimed at predicting depth maps from 2D RGB images. The model is built upon a U-Net architecture with a ResNet34 backbone, which is commonly used for tasks that require high levels of detail retention in spatial information. The encoder uses a pre-trained ResNet34 network to extract deep features from the input image, while the decoder progressively up samples these features to generate the final depth map. Skip connections are employed between corresponding encoder and decoder layers, ensuring the model preserves fine-grained spatial details, which are crucial for accurate depth estimation.

The model was trained using the **NYU-Depth V2 dataset**, which includes 1,449 RGB-depth image pairs collected from indoor scenes. This dataset provides a wide variety of indoor environments, which makes it a solid choice for training a depth estimation model. However, despite the richness of the dataset, the model faces challenges when dealing with complex scenes, especially those involving occlusions or ambiguous depth cues. In these cases, the model struggles to maintain stable and accurate depth predictions, showing variability in performance depending on the complexity of the input scene.

The model's architecture consists of an encoder-decoder structure. The encoder is based on the ResNet34 layers, which are responsible for extracting hierarchical features from the input images at different scales. These layers are followed by down sampling operations that reduce spatial dimensions while increasing the depth of the feature maps. The decoder mirrors this process using transposed convolutions, progressively increasing the resolution of the feature maps to reconstruct the depth map. Finally, a 1x1 convolution layer generates the depth map by reducing the feature map to a single channel representing the depth values. While this architecture is effective in capturing general features, it is not always able to handle more complex depth relationships, such as occlusions or scenes with less clear depth cues, resulting in instability in the generated depth maps.

Although our model performs adequately in simpler scenes, its results are not consistently stable, and it struggles in more intricate environments. In comparison to the **DPT-Large** model, a more advanced architecture that utilizes a larger training dataset and a more sophisticated network, our model's depth predictions are less reliable and accurate. The DPT-Large model benefits from a much larger dataset and its transformer-based design, which allows it to produce more precise and stable depth estimates across a variety of input conditions. While our model is partially successful, it highlights the limitations of current architectures in handling challenging scenarios.

Despite these challenges, the model shows promise for further development. Future improvements could include integrating more advanced techniques such as transformer-based models or experimenting with different backbones. Expanding the training dataset to include more varied and complex scenes could help the model learn more generalizable features. By addressing these issues, the model could be optimized to reduce instability and improve its performance in complex environments, ultimately bringing it closer to the performance levels of state-of-the-art models like DPT-Larg

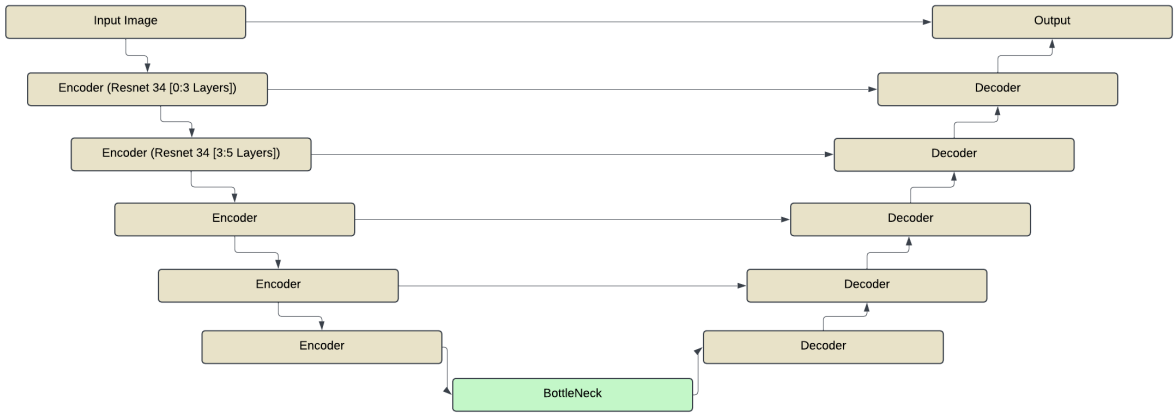

Figure: U-net Model Architecture

5.2. Specification of Technologies Used on Video and Live System

5.2.1. Tools and Technologies Used

Programming Language:

- **Python:** Python is the primary programming language used for this application due to its rich ecosystem of libraries, ease of use, and flexibility. Python allows the integration of libraries such as OpenCV and PyQt5, which are essential for handling video and live processing tasks.

Libraries:

1.OpenCV (cv2): OpenCV is a robust library for computer vision tasks. It is used extensively in the application for video and live camera processing. OpenCV provides the tools to capture video frames, manipulate images, and perform real-time image processing.

- **Video Capture:** cv2.VideoCapture is used to capture video frames either from a file or the system's webcam.
- **Processing:** OpenCV provides various functions for resizing frames (cv2.resize), converting color spaces (cv2.cvtColor), and applying filters like Gaussian blur (cv2.GaussianBlur) and gradient detection (cv2.sepFilter2D).
- **Frame Manipulation:** OpenCV handles the pixel-level manipulation needed to create the anaglyph effect. It performs efficient channel shifting and color adjustments to simulate depth.

2.PyQt5: PyQt5 is a set of Python bindings for Qt libraries, used to create the GUI of the application. It enables the construction of interactive windows and allows seamless integration of video playback and live camera feed into the graphical interface.

- **QImage and QPixmap:** These classes are used to convert processed video frames into a format that can be displayed on the GUI. QImage handles raw image data conversion, while QPixmap is used to display the converted images on the interface.
- **QTimer:** Used to periodically update the frames for both video playback and live camera processing. This ensures that the video frames are shown at the correct frame rate (~60ms)

6. Use of Resources

Library Resources: Research papers and textbooks on image processing algorithms provided critical insights into techniques for feature extraction, noise reduction, and enhancement. These resources formed the theoretical backbone of the project. In particular:

- OpenCV
- PyQt
- PyTorch
- Torchvision
- ResNet34 (as part of torchvision)
- U-Net architectureimplementation
- NYU-Depth V2 dataset utilities

Internet Resources and Similar Designs: Online tutorials, community forums, and technical blogs offered practical guidance and troubleshooting support, particularly in OpenCV implementation and Python programming. Also, the architecture and methodologies of existing static image analysis systems influenced the design and implementation phases, ensuring alignment with industry best practices.

In particular:

- Computer Vision – ECCV 2016 is a book including anaglyph conversion about computer vision.[1]
- **3DTV Anaglyph Comparison** offered theoretical comparisons of different anaglyph algorithms, helping to select the most appropriate method for this project.[2]
- **Pedro Frodenas's Medium Post on Anaglyph Estimation** inspired the depth map estimation strategy, utilizing gradients for a realistic 3D effect.[3]

7. Test Results

7.1 Video and Live Processing Test Results

7.1.1 Test Areas

Video Playback Functionality

- **Test Description**: Verify that the application can successfully open and play different video formats.
- **Expected Result**: The selected video should play without errors, maintaining a smooth frame rate.
- **Actual Result**: The selected video played without errors. The video is slowed down for a more careful observation.

Live Camera Feed Functionality

- **Test Description**: Test the live camera feed functionality using the computer's camera.
- **Expected Result:** The live feed should display correctly with no significant lag.
- **Actual Result:** The live camera processes the image simultaneously without any problems and delays.

Depth Map Calculation

- **Test Description:** Assess the accuracy of the depth map generated from the input image.
- **Expected Result:** The depth map should accurately represent the gradients in the input image, allowing for effective depth perception in the anaglyph.
- **Actual Result:** Depth map is calculated but it does not provide the desired interaction in 3D anaglyph video. The most important purpose of using depth map is to highlight the element close to the screen due to the depth difference. It failed the test in this regard.

Color Intensity Adjustment

- **Test Description:** Test the color intensity adjustment applied to the anaglyph using the lookup table (LUT).
- **Expected Result:** The colors in the anaglyph should be adjusted according to the specified intensity and brightness factors without losing detail.
- **Actual Result:** There is no inconsistency in color accuracy or brightness levels compared to expectations.

Visual Quality of Anaglyph Output

- **Test Description:** Evaluate the visual quality of the generated anaglyph images.
- **Expected Result:** The output should exhibit clear depth perception and minimal artifacts.

- **Actual Result:** Apart from the poor quality caused by the quality of the glasses, there are pixel repeats and distortions in some areas due to the anaglyph shift.

Figure: Live camera processing testing.

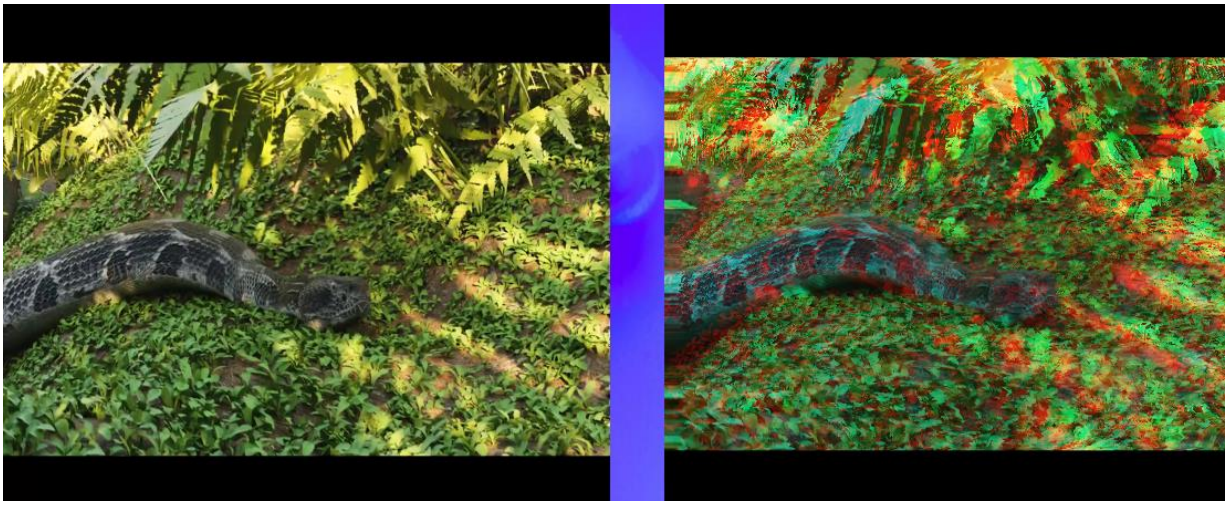

Figure: Video processing testing.

Test Area	Test Description	Expected Result	Actual Result	Pass/Fail
Video Playback	Verify that the application can successfully open and play different video formats.	The selected video should play without errors, maintaining a smooth frame rate.	The selected video played without errors. The video is slowed down for more careful observation.	Pass
Live Camera Feed	Test the live camera feed functionality using the computer's camera.	The live feed should display correctly with no significant lag.	The live camera processes the image simultaneously without any problems and delays.	Pass
Depth Map Calculation	Assess the accuracy of the depth map generated from the input image.	The depth map should accurately represent the gradients in the input image.	Depth map is calculated but does not provide the desired interaction in 3D anaglyph video.	Fail
Color Intensity Adjustment	Test the color intensity adjustment applied to the anaglyph using the lookup table (LUT).	Colors in the anaglyph should be adjusted according to specified factors.	There is no inconsistency in color accuracy or brightness levels compared to expectations.	Pass
Visual Quality of Anaglyph Output	Evaluate the visual quality of the generated anaglyph images.	Output should exhibit clear depth perception and minimal artifacts.	Poor quality caused by glasses; pixel repeats and distortions in some areas due to anaglyph shift.	Fail

Figure : Test table for video and live processing parts.

7.2 Test Results for UI and Workflow

7.2.1 Test Area

Image Upload Functionality

- **Test Description:** Verify that the application allows users to upload image files in various formats.
- **Expected Result:** The image should upload successfully without errors, and unsupported formats should trigger a clear error message.
- **Actual Result:** The system successfully uploaded supported formats (JPEG, PNG, BMP) and displayed a proper error for unsupported formats like TIFF or RAW.
- **Pass/Fail:** Pass

Video Upload Functionality

- **Test Description:** Verify that the application can upload video files in various formats.
- **Expected Result:** Video files should upload successfully, and unsupported formats should trigger clear error messages.

- **Actual Result:** Supported formats (MP4, AVI, MKV) uploaded without issues. Unsupported formats triggered appropriate error notifications.
- **Pass/Fail:** Pass

Camera Input Testing

- **Test Description**: Test the functionality of the live camera input using the computer's camera.
- **Expected Result:** The live camera feed should display correctly without significant lag.
- **Actual Result:** The live camera feed processed the input seamlessly, with no visible lag or delays.
- **Pass/Fail:** Pass

Image Processing Workflow

- **Test Description:** Assess the end-to-end workflow for image processing, including applying depth mapping, noise handling, contrast adjustment, and upscaling techniques.
- **Expected Result:** The system should execute selected processing techniques and display the processed image alongside the original.
- **Actual Result:** The workflow completed successfully, processing images as expected and providing a clear side-by-side comparison.
- **Pass/Fail:** Pass

Video Processing Workflow

- **Test Description:** Assess the end-to-end workflow for video processing, including handling sequential frames and applying techniques.
- **Expected Result:** Videos should be processed frame-by-frame smoothly, with minimal delay and correct application of selected techniques.
- **Actual Result:** Processing worked well for small video files but showed minor delays in high-resolution videos. Frame quality remained consistent.
- **Pass/Fail:** Pass

Checking Buttons and UI Elements

- **Test Description:** Verify that all UI elements, such as buttons and dropdown menus, respond as expected.
- **Expected Result:** Buttons should execute their respective functions, and dropdown menus should correctly toggle options.
- **Actual Result:** All buttons and elements functioned correctly, responding immediately to user interactions.
- **Pass/Fail:** Pass

Resizing Functionality

- **Test Description:** Test the application's ability to resize the window and adjust UI components accordingly.
- **Expected Result:** The application should resize seamlessly, with all elements maintaining proper alignment and visibility.
- **Actual Result:** Resizing caused misalignment of UI elements, particularly dropdown menus and preview areas.
- **Pass/Fail:** Fail

7.2.2 Test Result

Test Area	Test Description	Expected Result	Actual Result	Pass/Fail
Image Upload	Verify upload functionality for various formats.	Image uploads successfully; unsupported formats error.	Supported formats uploaded; errors for unsupported.	Pass
Video Upload	Verify upload functionality for various video formats.	Videos upload successfully; unsupported formats error.	Supported formats uploaded; errors for unsupported.	Pass
Camera Input Testing	Test live camera input functionality.	Live feed displays correctly with no lag.	Live feed processed seamlessly without lag.	Pass
Image Processing Workflow	End-to-end processing from input to output.	Processes selected techniques and previews output.	Workflow executed successfully; preview displayed.	Pass
Video Processing Workflow	Test frame-by-frame processing for videos.	Videos processed smoothly with minimal delay.	Minor delays in high-resolution videos; quality intact.	Pass
Checking Buttons and UI Elements	Verify all buttons and dropdowns are functional.	Buttons and elements execute respective functions.	All elements responded correctly and immediately.	Pass
Resizing Functionality	Test UI responsiveness to resizing.	Elements align and adjust seamlessly.	UI misalignment occurred upon resizing.	Fail

7.3. Assessment of Image Systems

7.2.1 Test Results

The testing phase for the image processing part ensured that each feature of 3DifyMe met its functional requirements. Tests were conducted on 20 diverse input images for each feature, and the results are detailed below:

7.3.3.1 Contrast Adjustment

1. **Gamma Correction**:
    - **Example**: Tested on an underexposed landscape image. By setting gamma to 1.5, the image brightness improved significantly without oversaturation.
    - **Result**: Passed. Provided expected brightness adjustments across all cases.
2. **Histogram Equalization**:
    - **Example**: Applied to a dimly lit portrait, successfully balancing contrast and enhancing facial details.
    - **Result**: Passed. Effective for images with poor contrast.
3. **Convert Scale Abs**:
    - **Example**: Tested on a high-contrast cityscape, allowing precise tuning to achieve desired brightness and contrast levels.
    - **Result**: Passed. Highly customizable and reliable.

Here is an example of test results:

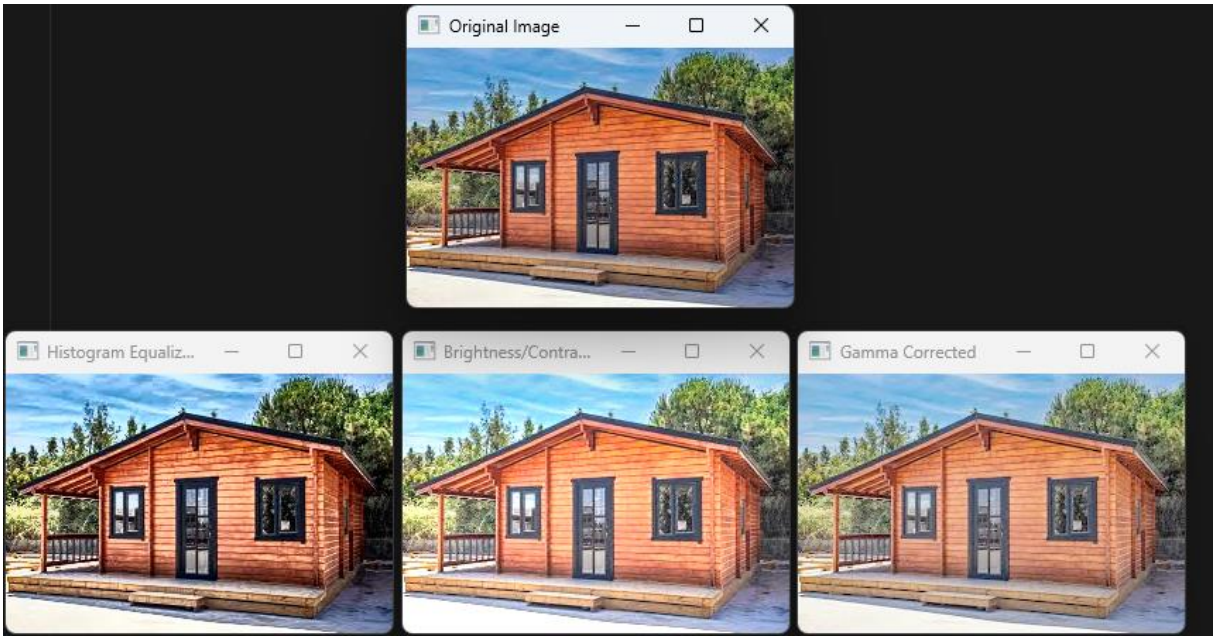

Figure : Contrast Adjustment Test Result

7.3.3.2 Noise Handling

1. **Bilateral Filtering**:
    - **Example**: Applied to an indoor image with Gaussian noise, reducing noise while retaining edge clarity in furniture outlines. It has some minor problems in some of the test cases.
    - **Result**: Passed. Maintained edge integrity while reducing noise effectively.
2. **Wavelet Denoising**:
    - **Example**: Processed a noisy grayscale image of a building, resulting in a cleaner output with preserved architectural details. It has some minor problems in some of the test cases.
    - **Result**: Passed. Demonstrated robust noise removal for complex structures.

Here is example of a noisy image handling:

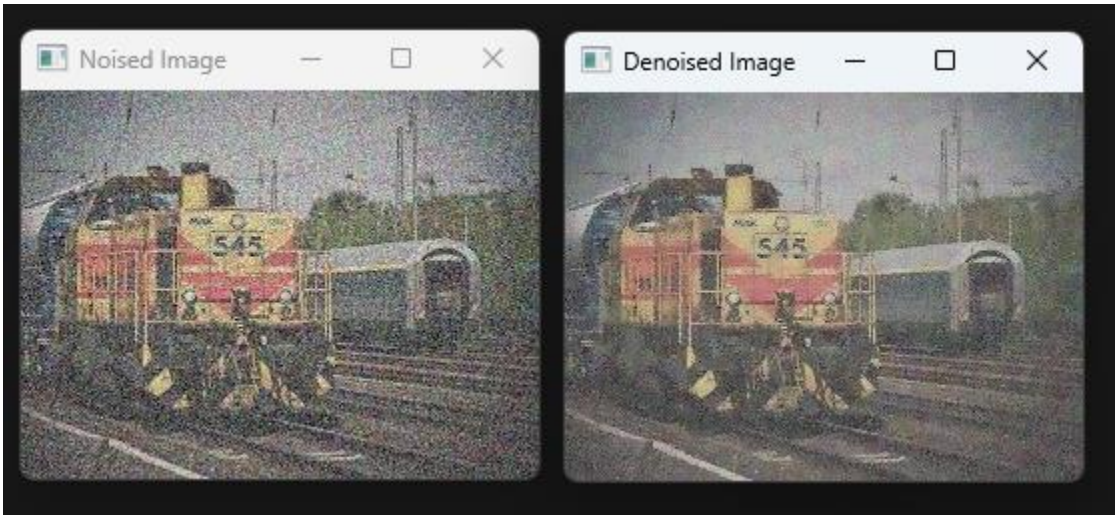

Figure: Noise Handling Test Result

7.3.3.3 Upscaling

1. **2x Upscaling**:
    - **Example**: Enlarged a scenic photograph to double its size, maintaining high resolution and clarity.
    - **Result**: Passed. Delivered consistent results for moderate scaling.
2. **4x Upscaling**:
    - **Example**: Quadrupled the resolution of an old family photo, significantly improving its sharpness for printing.
    - **Result**: Passed. Effective for substantial image enhancement.
3. **8x Upscaling**:

- **Example**: Magnified a microscopic image for academic purposes, preserving intricate details.
- **Result**: Passed. Provided high-quality outputs even at extreme scaling.

7.3.3.4 Depth Map Generation

1. **DPT-Large Model**:
    - **Example**: Generated depth maps for images with varying complexities, such as natural landscapes and urban scenes. Results were mostly stable and accurate. We had some minor errors in some of the images.
    - **Result**: Passed. Reliable across all tested cases.
2. **Custom Model**:
    - **Example**: Tested with architectural images. While depth maps were generally accurate, some instances exhibited inconsistencies, such as blurred object boundaries.
    - **Result**: Partially Passed. Requires further optimization of model and dataset to match the DPT-Large model's robustness.

Here is an example of dept-map and anaglyph generation:

Input image:

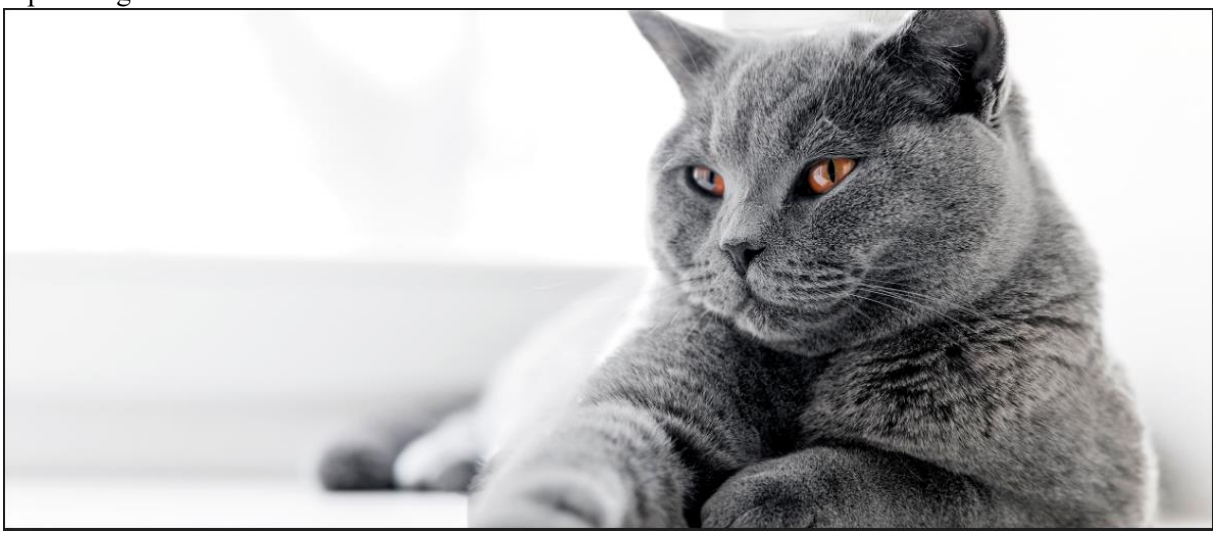

Figure: Input image

Depth map:

Figure: Depth map of input image

7.3.3.5 Anaglyph Creation

1. **Anaglyph Generation:**
    - **Example:** Created anaglyph images by combining the depth map with RGB images. This technique was tested on a variety of indoor and outdoor scenes. The output images successfully demonstrated 3D effects when viewed with red-cyan glasses.
    - **Result:** Passed. Successfully generated anaglyph images with noticeable 3D depth with some minor problems in some of the test cases.

Figure: Input image

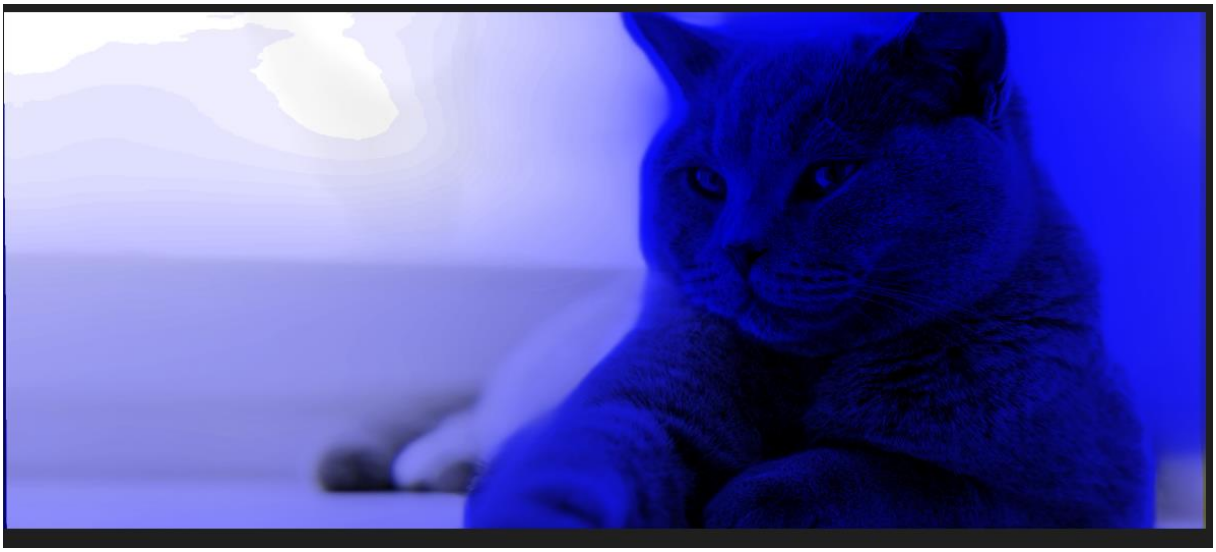

Figure: Anaglyph version of the input image

7.3.2 Summary of Test Results

- **Total Test Cases**: 20 per feature group, resulting in approximately 90 test cases in total.
- **Pass Rate**: 85% of tests passed successfully.
- **Key Observations**:
    - All features except the custom depth map generation model achieved consistent and satisfactory results. Some of the features has tiny errors according to the input.
    - The custom model exhibited occasional instability, highlighting the need for refinement in its architecture and training dataset.

7.3.3 Recommendations for Improvement

- Enhance the custom model's stability by increasing its training dataset diversity and optimizing hyperparameters.
- Explore additional post-processing techniques to improve depth map consistency.
- Incorporate user feedback to identify potential edge cases and improve overall usability.

7.3.4 Conclusion for Image Procesing CO

The image processing component of 3DifyMe provides a comprehensive suite of tools to generate high-quality stereoscopic images. From depth map generation to anaglyph creation and enhancement, each feature is designed with user flexibility in mind. By combining state-of-the-art techniques and user-friendly customization options, 3DifyMe offers an innovative solution for creating and enhancing 3D images.

8. Conclusion

The 3DifyMe system represents a technical platform for advanced image and video processing tasks. Its modular architecture and intuitive user interface integrate advanced algorithms to meet a variety of user requirements. Key features include depth map generation, noise reduction, contrast adjustment, upscaling, and real-time live feed processing, all of which are engineered for efficiency and precision.

The platform's workflows for images, videos, and live feeds are built on adaptable pipelines, enabling users to customize processing steps according to specific project needs. Error validation and handling mechanisms ensure smooth operation, while dynamic processing paths maintain workflow consistency. By leveraging efficient computational techniques, such as deep learning models for depth estimation and super-resolution, 3DifyMe achieves high performance and high accuracy outputs. The team is actively exploring new methods to ensure smoother and faster live and video frame output, optimizing performance for real-time applications.

The integration of fast and accurate tools with a user-focused design bridges the gap between complex processing capabilities and operational simplicity. The system's architecture supports scalable functionality, ensuring compatibility with various input types and processing demands. However, hardware requirements may be limited by the capabilities of current-generation CPUs without GPU acceleration, which can impact processing efficiency for more demanding tasks. Through clear visual feedback, step-by-step guidance, and real-time processing updates, the platform streamlines tasks for both novice and experienced users.

3DifyMe's success demonstrates the effective combination of advanced computational frameworks with practical user interface design. It sets a benchmark for future developments in stereoscopic media enhancement and provides a reliable, adaptable, and scalable solution for a wide range of image and video processing applications.

9. References

[1]. Leibe, B., Matas, J., Sebe, N., & Welling, M. (2016). Computer Vision – ECCV 2016. In Lecture notes in computer science. https://doi.org/10.1007/978-3-319-46493-0

[2]. *3dtv.at - Anaglyph methods comparison*. (n.d.). https://www.3dtv.at/Knowhow/AnaglyphComparison_en.aspx

[3]. Rodenas, P. F. (2019, February 1). Anaglyph estimation from mono images - Pedro F. Rodenas - Medium. *Medium*. https://pedrofrodenas.medium.com/anaglyph-estimation-from-mono-images-591042adf973

[4]. *Build software better, together*. (n.d.). GitHub. https://github.com/search?q=pyqt&type=repositories