PROJECT REPORT

CHROMA KEY
STATISTICAL METHODS I

Presented To
ARNAB CHAKRABORTY

Presented By
AMAN SINGH (BS2308)
SATYAPRAKASH SAHDEV JAISWAR (BS2342)
RONAK GUPTA (BS2339)
SAURABH BHIKAN NANKAR (BS2344)
SUBHADIP ROY (BS2353)
PALLAV SAHAY (BS2335)

INDEX

1. ABSTRACT
2. INTRODUCTION
3. APPROACH 1(MIN-MAX BOUND)
4. APPROACH 2(PIXEL PREDOMINANTLY GREEN)
5. APPROACH 3(APPROACH 1+2 AND GREY BOUND)
6. SHINY IMPLEMENTATION
7. CONCLUSION
8. GUIDE TO USE THE SOFTWARE
9. FURTHER IMPROVISATION / SCOPE

Abstract

This project unfolds as a journey through the development of an innovative approach to implement chroma keying, a pivotal technique in the realms of filmmaking, photography, and video production. The endeavor began with the exploration of different approaches, each aimed at achieving a seamless removal and replacement of specific-coloured backdrops.

Approach 1 (Min-Max Bound): The initial exploration centered around a Min-Max Bound methodology. While promising, this approach faced limitations and challenges, leading to the realization that a more nuanced strategy was essential.

Approach 2 (Pixel Predominantly Green): The second approach delved into the idea of Predominantly Green, seeking to leverage the prevalence of green screens in chroma keying scenarios. However, the findings revealed shortcomings and the need for a more comprehensive solution.

Approach 3 (Approach 1 + 2 and Grey Bound): Undeterred by setbacks, the project converged the strengths of both Approach 1 and Approach 2 while introducing a new element with the Grey Bound inclusion. This integrated approach emerged as the final solution, combining the strengths of its predecessors and addressing the shortcomings. Through meticulous experimentation and adjustment of parameters, this approach strives to offer a more robust and adaptable chroma keying solution.

This abstract encapsulates our journey of exploration, experimentation, and refinement, showcasing the resilience in the face of challenges and the emergence of a comprehensive chroma keying approach that holds promise for the future of visual media production.

Introduction

Within the framework of our project, our primary objective is to engineer an algorithm of notable efficiency, specifically tailored for the intricate task of background removal in photographs under specific conditions. Our endeavor encompasses a comprehensive process that extends beyond theoretical considerations, delving into practical implementation and refinement.

The initial phase involves the conceptualization and design of an algorithm that adheres to predefined conditions for background elimination. Subsequently, our focus shifts to the coding phase, where we meticulously craft a suitable program tailored to our algorithm. This code becomes the backbone of our project, serving as the tool that enables us to execute seamless background removal on a diverse range of photographs.

As we run our code across various images, a crucial aspect of our project unfolds — the identification and documentation of shortcomings in our algorithm. This iterative process is fundamental to our approach, as it allows us to systematically address and rectify limitations, gradually moving towards an outcome that meets our standards of acceptability.

A pivotal aspect of our algorithmic development centers around the definition of a "pixel of a special colour." This conceptualization lays the groundwork for our code's ability to discern and target specific colours within the photos, forming the basis for effective background removal. By refining our understanding of what constitutes a "pixel of a specific colour," we enhance the precision and reliability of our algorithm.

The continuous cycle of implementation, observation, and refinement characterizes the dynamic nature of our project. Through these iterative steps, we not only fine-tune the algorithm but also gain valuable insights into the intricacies of background removal in diverse photographic scenarios. Our commitment lies in achieving an algorithm that not only meets but exceeds expectations, setting a precedent for efficient and accurate background removal in the realm of visual media production.

Approach-1

Minimum and Maximum bound

A. Idea:

In our project, we adopt an iterative approach using a loop to navigate through each pixel, assessing whether the RGB components fall within a predefined range of green values. This methodology serves as a fundamental technique for green screen removal, akin to the principles of chroma keying. Essentially, we establish a threshold for green, and if a pixel's colour values reside within this defined range, we replace them with a new background.

B. Algorithm:

1. Image Retrieval:
   - Utilize the ***'readJPEG'*** function from the jpeg library or a similar function from the ***png*** library to read the image.

2. Colour Channel Range Definition:
   - Set minimum (min) and maximum (max) values for each of the three colour channels (Red, Green, Blue).

3. Original Image Plotting:
   - Plot the original image to provide a visual reference.

4. Result Array Initialization:
   - Initialize an array named 'result' with zeros, matching the dimensions of the input image.

5. Pixel Iteration:
   - Employ nested loops to iterate through each pixel in the image.

6. Colour Channel Check:
   - For each pixel, check if all three colour channels are within the specified range (min to max).

7. Result Assignment:

- If the condition is met, set the corresponding pixel in the result array to white (`c(1, 1, 1)`).

8. Preservation of Original Values:
   - If the condition is not met, retain the original pixel values in the result array.

This systematic algorithm encapsulates our approach to green screen removal, providing a structured overview of the methodology implemented in our project.

C. Implementation and Result:

INPUT	OUTPUT

D. Error:

1. Tolerance Levels:

Define tolerance levels or thresholds for comparing the green, red, and blue components. These thresholds should be adjustable to accommodate different images and lighting conditions. By allowing users to adjust these thresholds, you make your algorithm more versatile.

2. Colour Variation Issues:
   RGB values for green can vary significantly based on lighting conditions, shades of green, and camera settings. A fixed threshold might not account for these variations, leading to inaccurate removal or retention of green shades.
3. Complex Green Shades:
   Green screens may contain a variety of shades due to lighting variations or imperfections. A fixed threshold might struggle to distinguish between different green shades, causing issues in the removal process.
4. Spill and Reflections:
   The approach might not effectively handle colour spill or reflections from the green screen onto the subject. This can result in remnants of the green colour on the subject, diminishing the realism of the final composition.

Approach-2

Pixel Predominantly Green

A. Idea:

In our project methodology, we undertake a process of comparing the green component of each pixel with the red and blue components. We make decisions to remove pixels where the green component surpasses the red and blue components. This approach is deemed reasonable for green screen removal, aligning with the widely accepted notion that green screen backgrounds predominantly exhibit green hues. The rationale behind this lies in the expectation that the foreground subject should exhibit a more balanced colour distribution.

B. Algorithm:

1. Input:
   - We start by obtaining the pixel values (R, G, B) for every pixel in the image.

2. Comparison:
   - For each pixel, we engage in a comparison, evaluating the green component (G) against the red and blue components.
   - If WE find that G is greater than both R and B, we consider the green component to be dominant.

3. Decision:
   - Based on my analysis, if the green component is indeed dominating, we make a decision to either remove or replace the pixel.
   - If the green component doesn't dominate, we opt to keep the pixel unchanged.

4. Output:
   - The result of these decisions is a modified image with adjusted pixel values, reflecting the alterations made during the process.

C. Implementation and Result:

INPUT	OUTPUT

D. Error:

1. Green Balance Variation:
   While it's generally true that the green background should have a higher green component, the precise green balance can vary due to factors like lighting and colour calibration. You should allow for a certain level of flexibility in your algorithm to account for variations in the green background.

2. Foreground Variations:
   The foreground subject's colours can also vary, and in some cases, elements in the subject may have a higher green component than the background. This can happen if the subject contains green-coloured objects or if there's green spill from the background onto the subject. You need to be careful not to unintentionally remove parts of the subject.

3. Limited Sensitivity to Variations:
   Relying solely on the comparison of green, red, and blue components may not account for variations in lighting conditions, shadows, or colour intensity. The approach might mistakenly remove pixels in situations where the lighting causes variations in colour intensity on the green screen.

4. Foreground Colour Similarities:
   If the foreground subject exhibits green or similar colours, especially in clothing or certain objects, the methodology may erroneously remove these elements, leading to incomplete or distorted results.

5. Edge Artifacts:
   The approach may introduce artifacts along the edges of the subject due to the abrupt removal of pixels based on colour thresholds. This can result in a jagged or unnatural appearance at the boundary between the subject and the background.

Approach-3

The Final Approach

A. Idea:

While undertaking the simulation of an RGB colour transformation, our project involves a detailed exploration into the distinct characteristics of pixels featuring a greenish hue. Our approach adopts a pixel-by-pixel iteration, thoroughly examining each pixel within the image. Throughout this meticulous process, we selectively exclude pixels whose green component surpasses that of the red and blue components.

In our project, we have implemented a user-configurable threshold to enhance flexibility in the filtering process. This dynamic threshold allows users to actively influence the elimination criteria, ensuring pixels are excluded only when their green component exceeds the specified threshold. This customization empowers users to fine-tune the filtering process according to their preferences.

Furthermore, our project incorporates a green upper bound to mitigate the inadvertent removal of pixels primarily characterized by light green tones. This thoughtful addition ensures that pixels with predominantly light green shades are not mistakenly discarded, contributing to a more nuanced and controlled approach to pixel filtering.

Additionally, our project acknowledges the importance of retaining grey pixels with the maximum green component while having red and blue values in close proximity to green. This consideration prevents the unintended removal of such pixels, adding another layer of precision to our pixel filtering methodology.

B. Algorithm:

1. Objective:
- Implement the ***'process_image'*** function to analyse and modify an input image based on specific conditions.
2. Input:
- Accept four parameters:
  - 'image': The input image in RGB format.

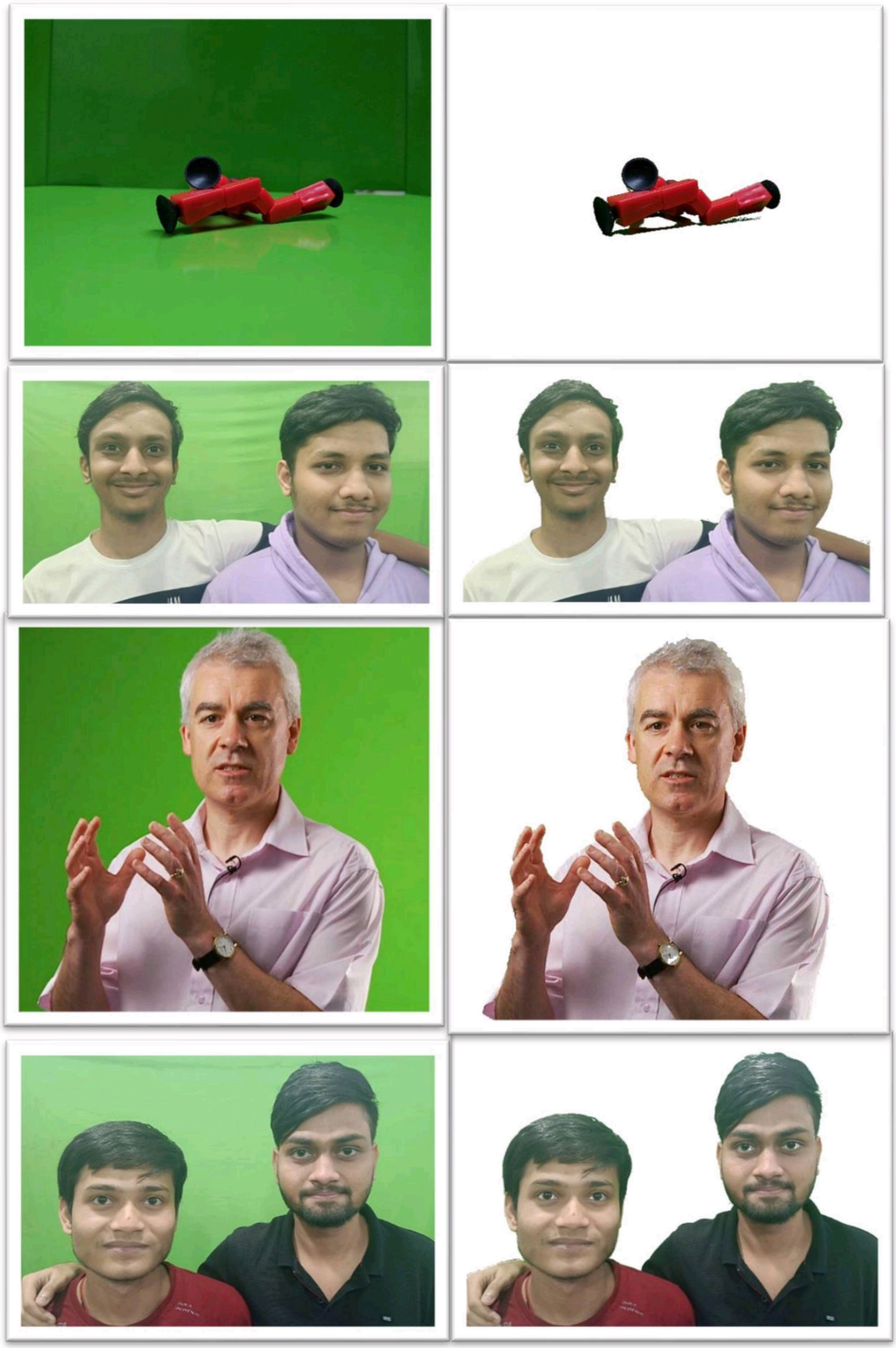
- ‘threshold’: User-defined threshold for the green component.
- ‘threshold2’: Another user-defined threshold for the green component.
- ‘threshold3’: User-defined threshold for absolute differences between RGB components.

3. Processing Steps:
   a. Initialization:
      - Create an array (`result`) with the same dimensions as the input image to store the processed result.
   b. Pixel Iteration:
      - Iterate through each pixel in the image using nested loops.
   c. Extract RGB Values:
      - Extract the red (`r`), green (`g`), and blue (`b`) components of the current pixel.
   d. Condition Checking:
      - Check the following conditions for each pixel:
        - The green component is the maximum among the RGB values.
        - The green component is within the user-defined thresholds (‘threshold’ and ‘threshold2’).
        - Absolute differences between red and blue, red and green, and blue and green are greater than or equal to ‘threshold3’.
   e. Pixel Replacement:
      - If all conditions are met, set the pixel in the `result` array to white (full intensity); otherwise, retain the original pixel values.
   f. Result Array:
      - The processed image is stored in the `result` array.
4. Output:
- The ***`process_image`*** function returns the processed image in the form of the `result` array.

This algorithm outlines the systematic approach taken in the project, detailing the steps involved in processing each pixel of the input image based on specific user-defined conditions.

C. Implementation and Result:

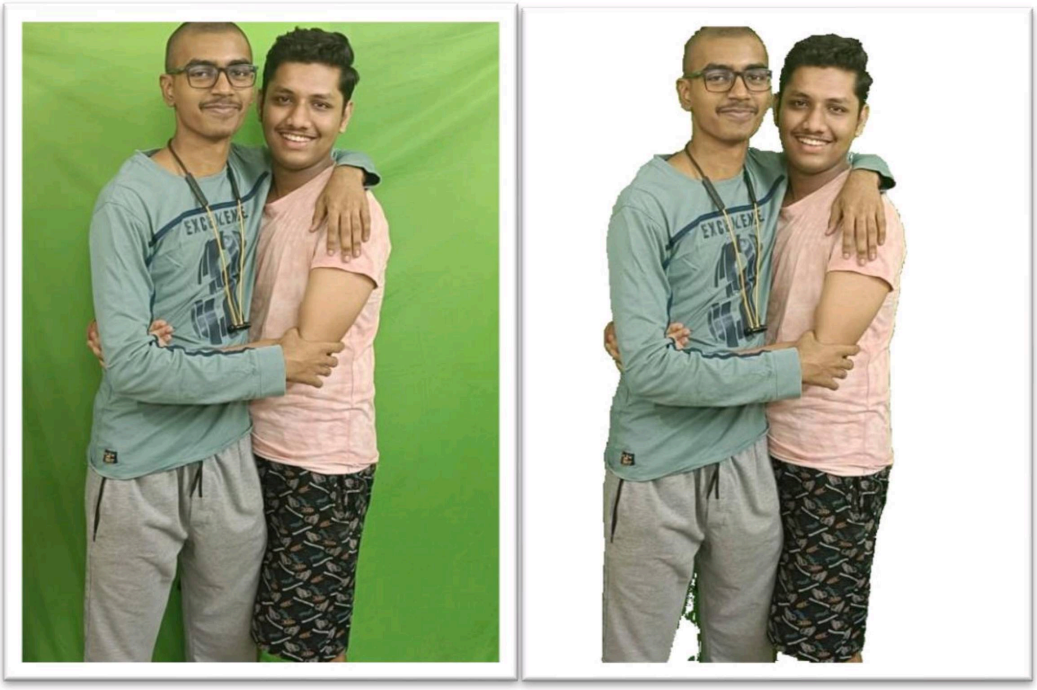
INPUT	OUTPUT



NOTE:- The grey bound helps us not to delete pixels that are a variation of green but not the green that's in the background, for example in the following output.

INPUT	OUTPUT (grey bound = 0)	OUTPUT

Similarly in the following : -



D. Error:

1. Edge Artifacts: Pixel-by-pixel iteration may lead to noticeable edge artifacts, especially in intricate or detailed subject outlines. This could result in jagged edges or halos around the composited subject, detracting from the overall visual quality.

2. Inconsistent Thresholding: Depending on the content of the image, the user-configurable threshold may lead to inconsistent results. Users might struggle to determine an optimal threshold for different scenes, potentially causing either an over-removal of the green screen or an insufficient removal, leading to colour spill artifacts.

3. Selective Exclusion Challenge: The selective exclusion of pixels based on the green component alone may not fully address variations in lighting conditions or shades of green. This could result in incomplete removal of the green screen, leaving residual artifacts and impacting the realism of the composited image.

4. Complexity for Varied Scenes: While the approach considers a green upper bound to preserve light green tones, it may face challenges when applied to images with varying lighting conditions or scenes with intricate green details. Adapting to

diverse scenarios could pose difficulties in achieving consistent and accurate green screen removal.

5. Handling of Grey Pixels: The consideration for retaining grey pixels with a maximum green component, accompanied by red and blue values close to green, may introduce complexities. Depending on the specific shades involved, this approach may inadvertently result in the removal of pixels crucial for the subject's fidelity.

SHINY Implementation

In our project, we've found ourselves encountering a familiar challenge, reminiscent of the obstacle we faced in Approach 1— the need to adjust the threshold. To address this, we've developed a Shiny app that facilitates dynamic threshold adjustment for all three thresholds.

In crafting this app, we harnessed the capabilities of Shiny, a web application framework for R that simplifies the creation of interactive web applications. Here's an overview of our project's Shiny app:

1. User Interface (UI):

a. We've created a fluid layout utilizing ***fluidPage***.

b. Integrated a file input feature for seamless image uploading.

c. Incorporated sliders enabling users to intuitively set image processing thresholds.

d. Introduced an action button to trigger the image processing operation.

e. Added output elements to visually present both the original and processed images.

2. Server Logic:

a. We've established a server function with parameters for input and output.

b. Utilized ***observeEvent*** to detect user clicks on the action button.

c. Loaded the uploaded image into the server.

d. Applied our image processing function (process_image) to the loaded image, using the specified thresholds.

e. Updated the output elements to showcase both the original and processed images.

3. User Input Handling:

a. Employed ***fileInput*** for streamlined image uploading.

b. Utilized ***sliderInput*** to facilitate user-friendly adjustment of processing thresholds.

c. Implemented robust validation and error handling mechanisms for user input.

4. Output Update:

a. Leveraged ***renderImage*** to dynamically display both the original and processed images, either side by side or in a layout optimized for clarity.

b. Optionally, introduced labels or captions for enhanced comprehension.

5. App Execution:

a. Executed the Shiny app using ***shinyApp***, seamlessly integrating the user interface and server functions to deliver a responsive and interactive tool for threshold adjustment in our image processing workflow.

Conclusion

Here, we conclude our exploration, with the third approach emerging as the final one. Our journey encompassed various methodologies, findings, errors, and invaluable lessons. The efficacy of chroma-keying is intricately tied to factors like lighting, colour consistency, and equipment quality. Green, a popular choice, minimizes the risk of colour interference with skin tones, clothing, and common scene elements. However, blue screens find use, especially when the subject incorporates green.

In our initial approach, Approach 1, we focused on delineating the range of red, blue, and green values to eliminate the unwanted green background. Challenges emerged in terms of precision, influenced by varying green background and lighting conditions. Edge handling complexities arose due to sharp foreground image edges. To address this, we explored techniques like alpha matting and feathering. For image processing, our arsenal includes colour correlation and contamination methods.

Approach 2 involved comparing green components with red and blue, removing instances where the green component significantly exceeded the others. Challenges included green balance variations from lighting and determining a flexible algorithm. Issues also arose from variations in foreground colour, particularly when subjects contained green, violating our condition. Identifying a consistent threshold for RGB values presented an additional challenge.

The culmination of our efforts in the third approach involved a revelation during the stimulation of an RGB colour simulator. Notably, green parts

exhibited more green pixels than red and blue. Consequently, we devised a strategy to remove pixels where the green component surpassed the red and blue components, with a user-adjustable threshold. This dynamic thresholding system accommodates various situations and ensures pixel removal when the green component exceeds the set threshold.

A guide to use the software: -

Guide: Using Chroma Keying Software for Green Screen Removal

Welcome to Chroma - Keying, a user-friendly software designed for achieving precise and aesthetically pleasing green screen removal. Follow this guide to optimize your experience and obtain high-quality results.

Step 1: Upload Your Image

- Click on the "Select an Image" button to upload the image you want to process. Ensure that the image has a green screen background.

Step 2: Adjust Green Screen Thresholds

- Green Lower Bound (threshold): Use the "Green Lower Bound" slider to set the minimum intensity of the green colour to be removed. Adjust this threshold until the green screen is adequately removed without affecting the subject.

- Green Upper Bound (threshold2): Utilize the "Green Upper Bound" slider to establish the upper limit of green intensity. This helps preserve light green tones and ensures a nuanced removal process.

- Grey Bound (threshold3): Adjust the "Grey Bound" slider to set the threshold for retaining greyish pixels with a maximum green component. This is essential for preserving/retaining certain shades during the removal process.

Step 3: Visualize Original and Processed Images

- Observe the original image in the left panel under "Original Image."

- Witness the processed image in the right panel under "Processed Image," showcasing the green screen removal based on your configured thresholds.

Tips for Varying Green Screens:

- Solid Green Screens: For uniformly coloured green screens, start with a conservative "Green Lower Bound" and gradually increase until the desired removal is achieved.

- Complex Green Screens: If your green screen has variations or intricate details, consider adjusting the "Green Lower Bound" and "Grey Bound" sliders to fine-tune the removal process.

Feel free to experiment with the sliders, and don't hesitate to fine-tune the thresholds to suit the nuances of your specific green screen. Happy editing!

FURTHER IMPROVEMENTS: -

Further enhancements to our chroma keying project involve refining key aspects such as edge smoothing and colour spill removal. One promising technique for enhancing the visual quality of our compositions is the utilization of connected components. By leveraging connected component analysis, we can intelligently identify and address regions that may exhibit artifacts or imperfections during the chroma keying process, thereby contributing to a smoother and more polished final result.

Moreover, envisioning the scalability and versatility of our project, we recognize the potential for extension to backgrounds of different colours. By expanding our chroma keying framework to accommodate a broader spectrum of backdrop colours, we aim to offer a more flexible and inclusive solution for diverse creative scenarios. This extension aligns with the project's commitment to adaptability and innovation, paving the way for a more comprehensive tool in the dynamic landscape of visual media production.