VisionEditCV

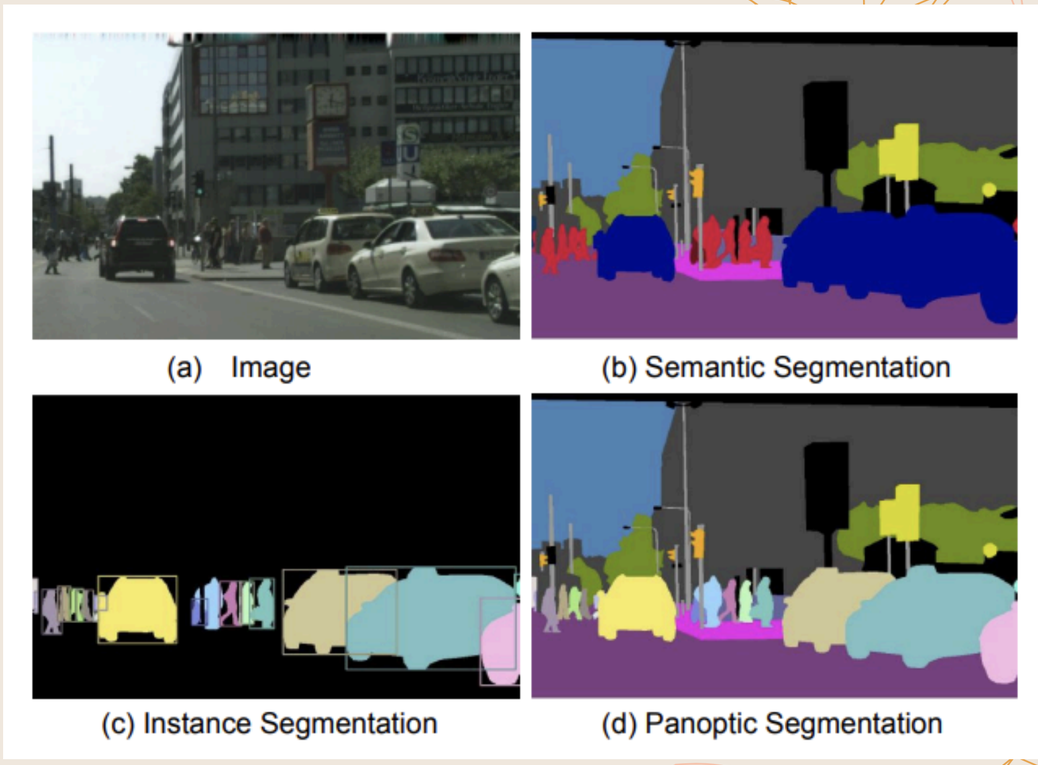
By The CV Geniuses

Group Members

Names	IDs
Lucky Agarwal	67011698
Poonnachote Kaewsuwan	67011649
Theechutha Suphachitkulchai	67011351
Papimon Leelamli	67011635

Background

- In image editing, being able to target only the desired area to edit is an important feature to have
- Rudimentary segmentation techniques usually requires the user manually tracing out the image
- Common algorithmic segmentation techniques include:
  - Edge detection based segmentation
  - Clustering based segmentation
  - Watershed segmentation
- Nowadays, deep learning models have been used to assist in segmentation such as U-Net, Mask R-CNN or ResNet



Problems

- Traditional segmentation techniques work mainly on simple or basic images.
- They usually provide only limited segmentation capability.
- In complex scenarios, these methods struggle to perform accurate segmentation.
- Achieving complex segmentation using traditional methods is very difficult.

OpenCV

Scope & Objectives

1. Build an app that utilizes SAM3 to segment objects based on text inputs or specific user-defined areas.

2. Ensure the system can accurately segment objects in static images.

3. Allow users to edit on identified segments, including background removal, adding colored backgrounds, and creating stickers.

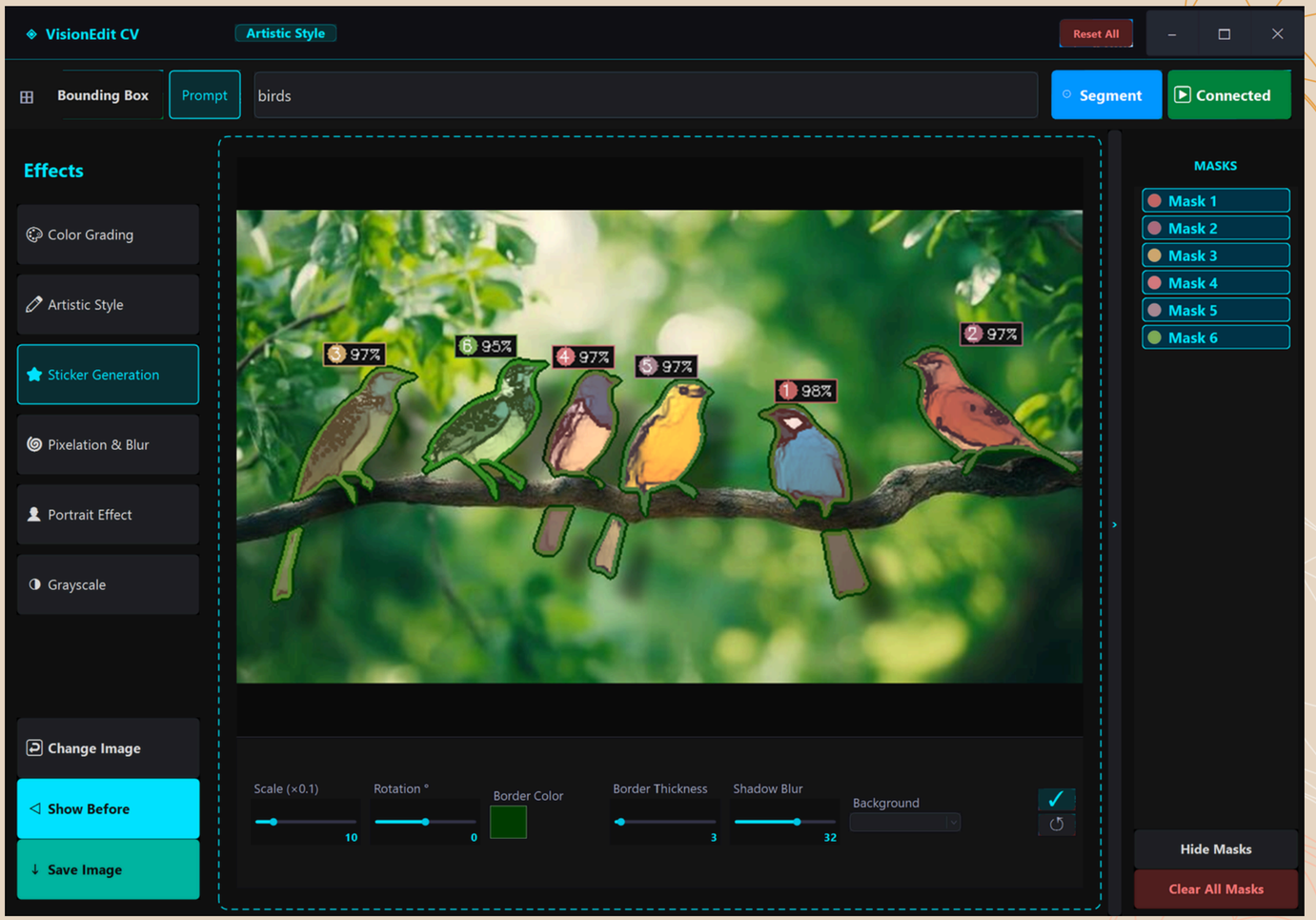
4. Allow users to apply effects individually or apply multiple effects together.

Introduction

VisionEdit CV is an Image Editing Application made using C# .NET and EmguCV.

Using the SAM3 model for image segmentation, it allows users to segment using bounding boxes or text prompts.

Once segmented, the application provides effects that can be applied independently to the masks.



Introduction - UI Details

- Built entirely in C# .NET 8 WinForms
- Custom buttons and navigation bar were programmed from scratch to achieve a modernized, sleek aesthetic since standard WinForms controls can look basic or outdated
- Designed with a cohesive dark theme to reduce eye strain and allow uploaded images to stand out, using a specific color palette:
  - Main Background: Deep Dark Gray (RGB: 18, 18, 18)
  - Side Panels: Slightly lighter (RGB: 24, 24, 24)
  - Main Text: Clear Light Gray (RGB: 220, 220, 220)
  - Secondary Text: Dimmed Gray (RGB: 140, 140, 160)

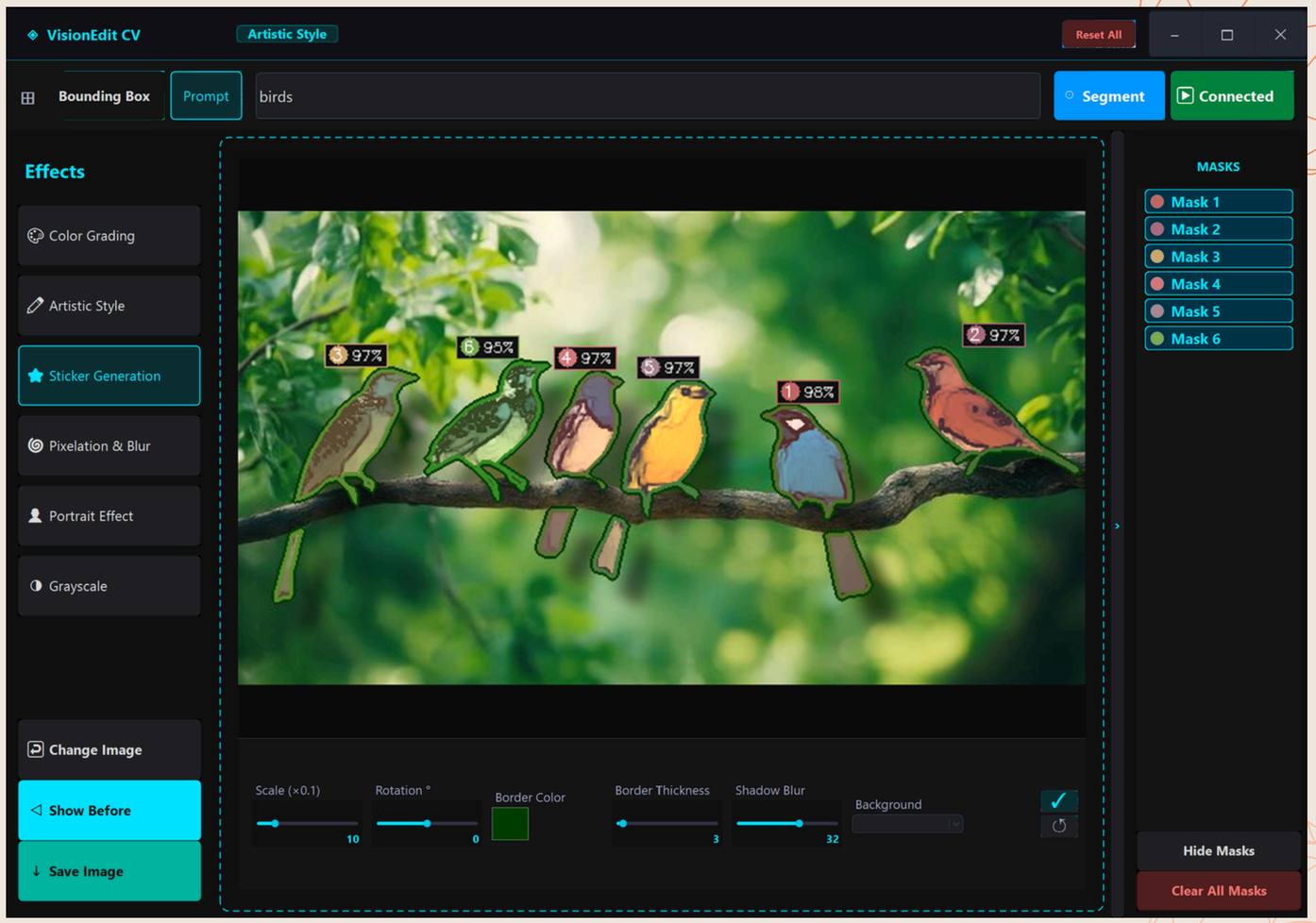
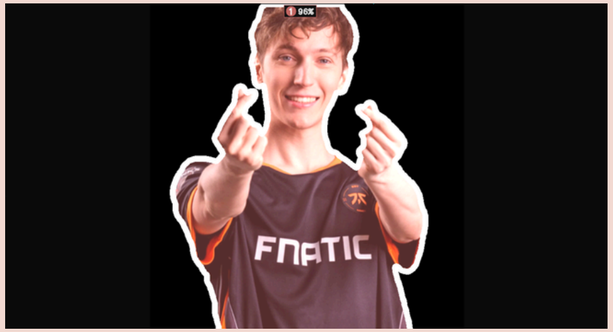


Image Effects

Color Grading

Artistic Style

Grayscale



Sticker Generation

Portrait Effect

Pixelation & Blur

Introduction - SAM3

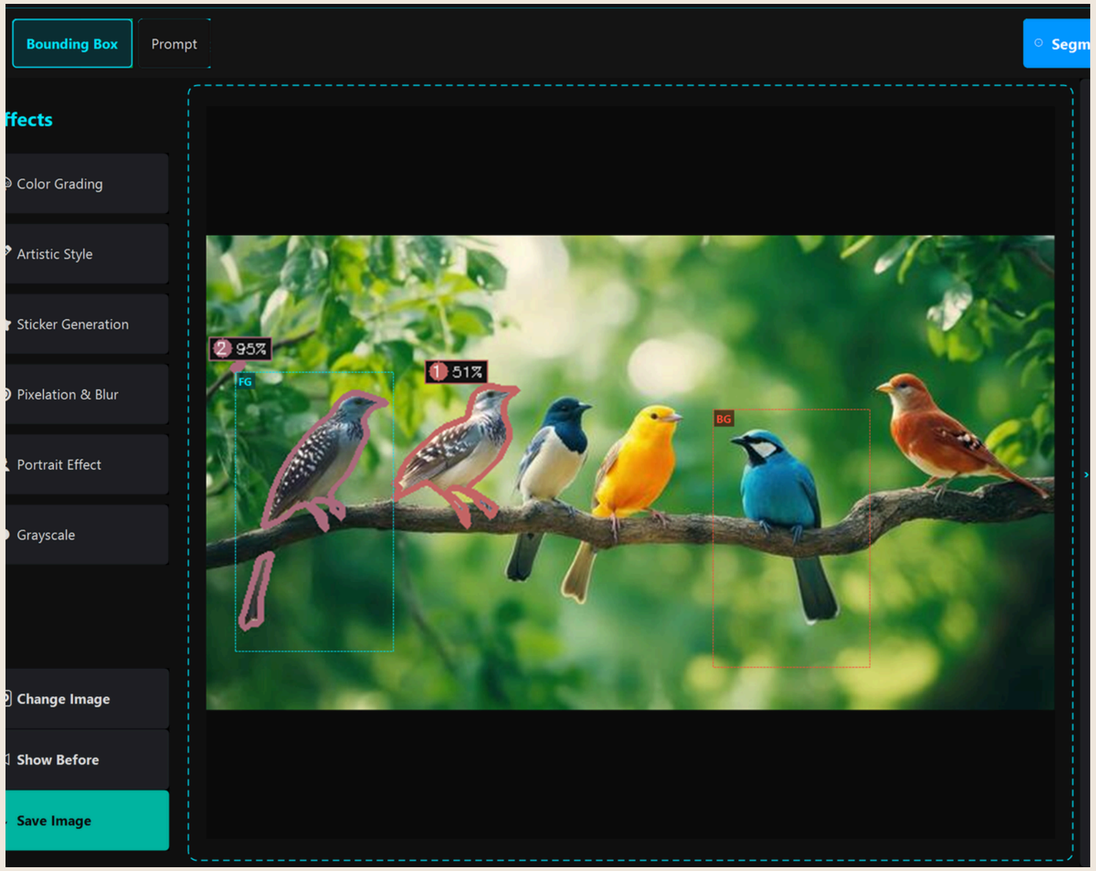
SAM3 is an image segmentation model by Meta AI which allows users to segment any parts in an image using either text prompts or bounding box defining the areas to segment.

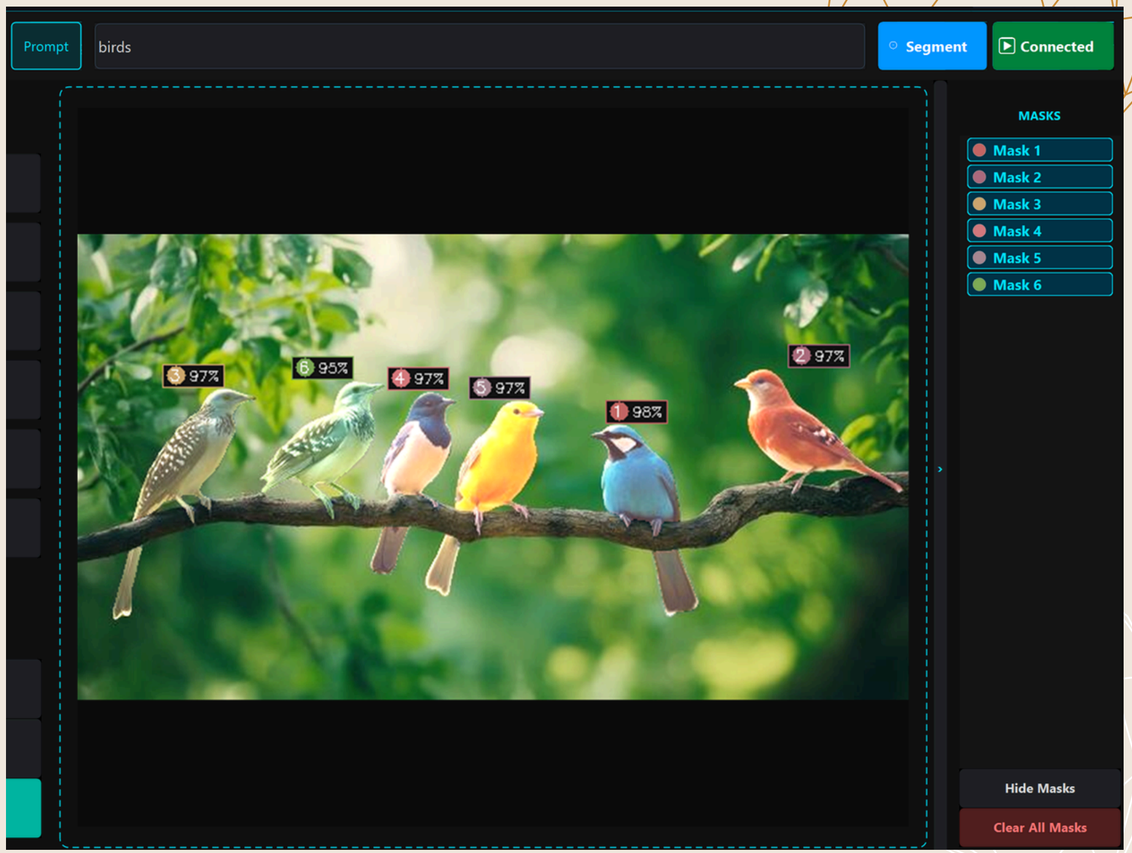
Reasons for using SAM3

ADVANTAGES OF USING SAM3

1. It can segment multiple objects simultaneously by utilizing intuitive text prompts, bounding boxes, or specific click points.
2. Can processes complex scenes fast.
3. Recognizes almost any object without needing preset categories
4. Provides highly accurate cutouts for seamless image editing.

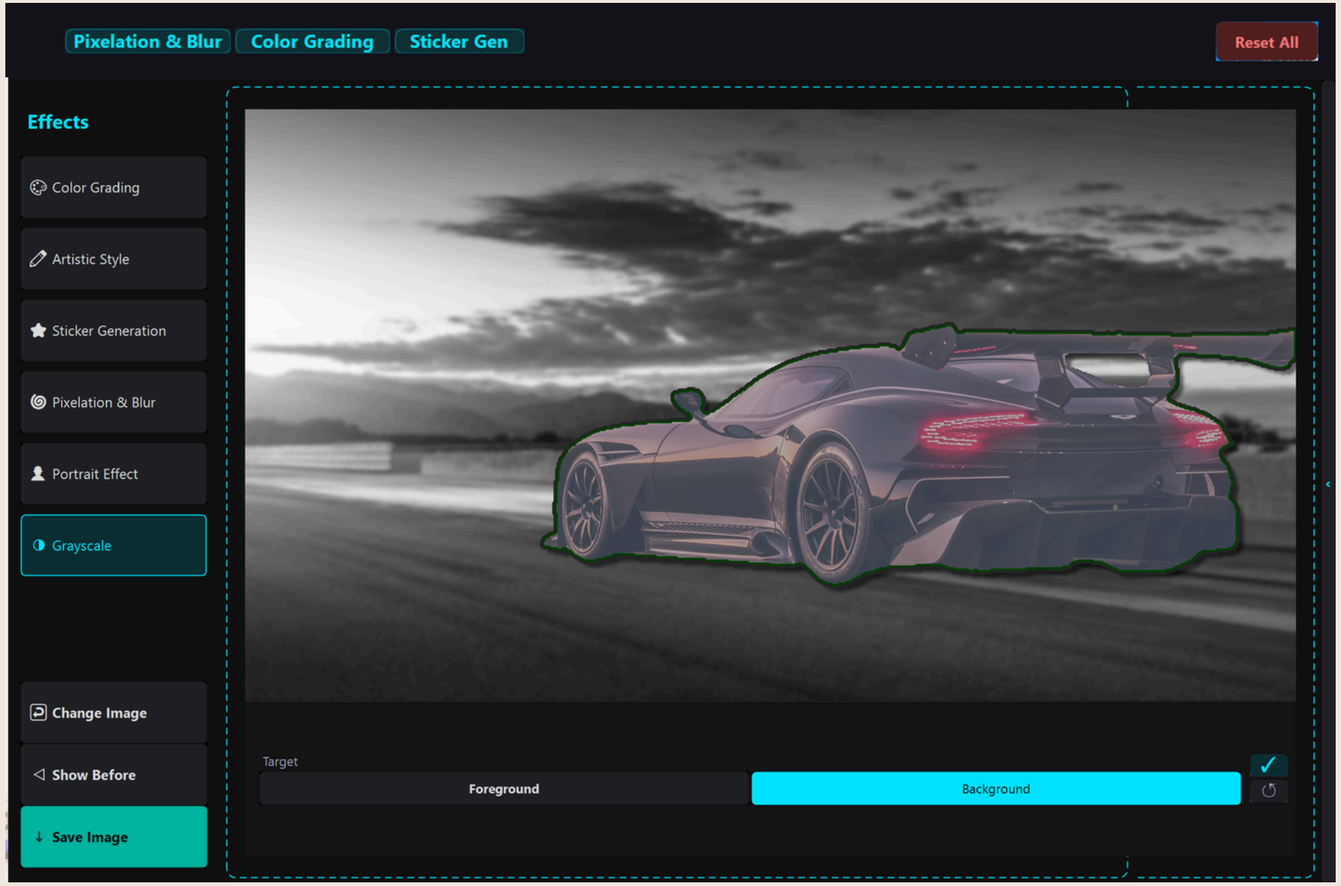
Masking (Box & Prompt)





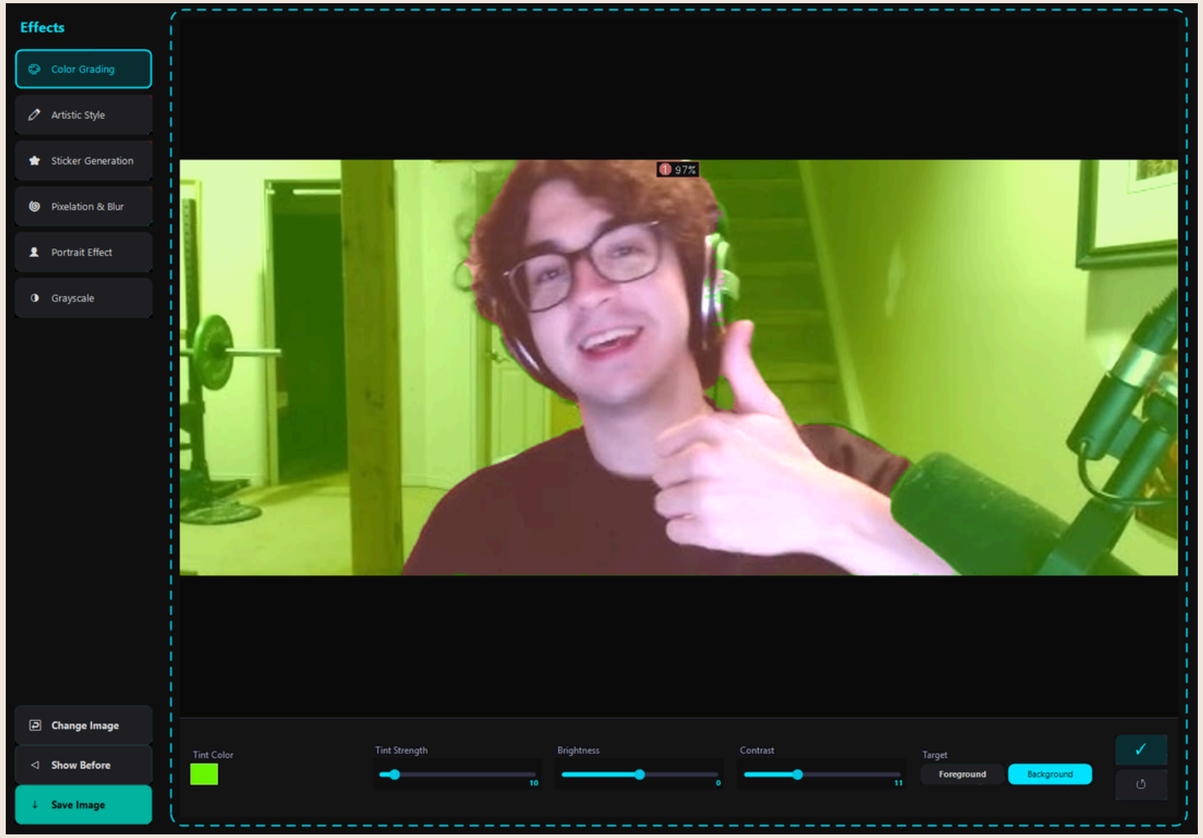
Users can choose whether to use ‘Bounding Box’ (positive or negative)and ‘Prompt’ to create a mask and can choose which mask to edit on the right sidebar. Additionally, they can hide and clear the masks.

Applying Effects



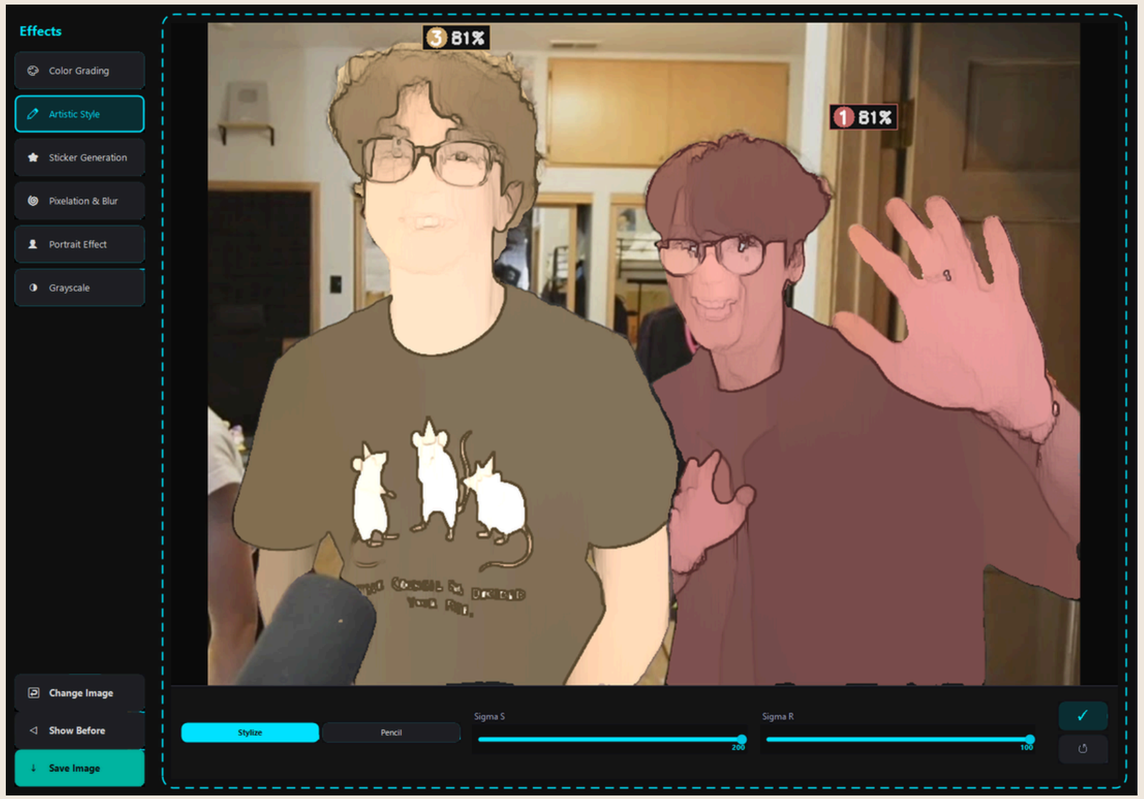
After selecting the mask, the user can choose what effects are to be applied. As well as configuring the effects as the user wants. They can apply mutiple effects together if needed

Color Grading



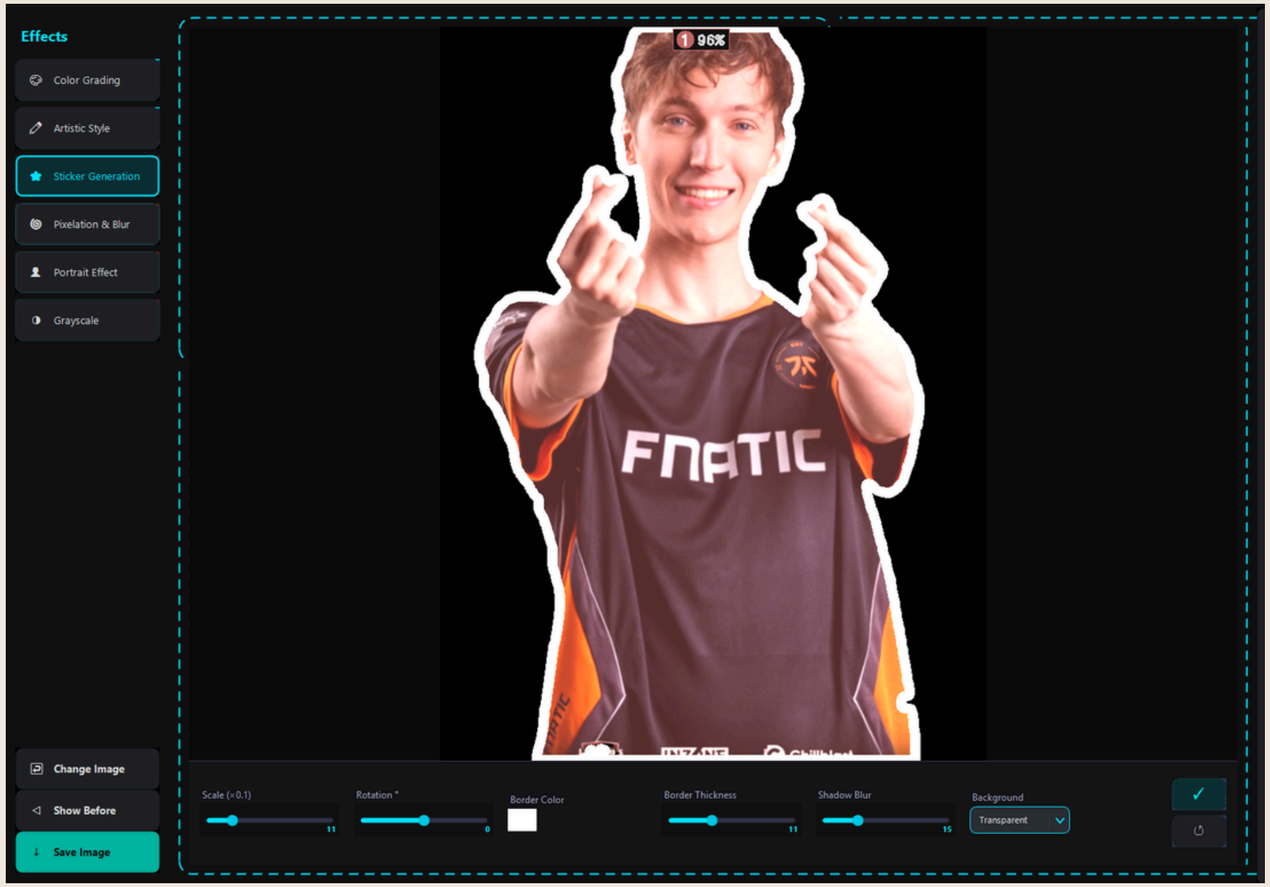
Users can select a 'Tint Color' and adjust the 'Strength,' 'Brightness,' and 'Contrast' sliders. Upon clicking 'Apply,' the software would then apply these effects specifically to the chosen 'Foreground' or 'Background' mask.

Artistic Style



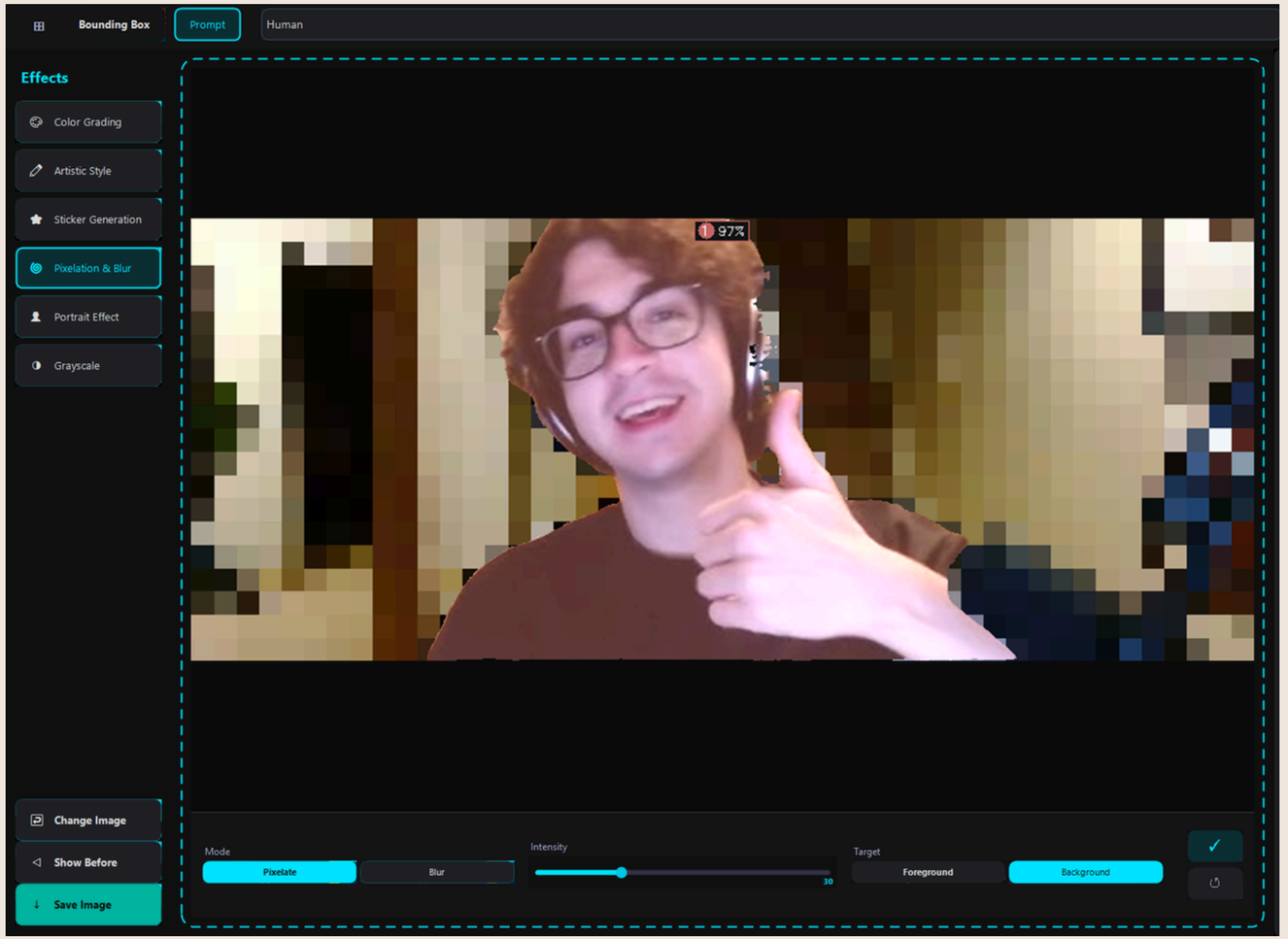
Users can select between 'Stylize' or 'Pencil' modes and adjust the 'Sigma S' slider to control the smoothing neighborhood size and the 'Sigma R' slider to manage color simplification. Upon clicking 'Apply,' the effects would be applied specifically to the chosen masks

Sticker



Users can select the different Backgrounds and adjust the 'Scale' and 'Rotation' sliders to adjust the cutout's size and orientation and the 'Border Thickness' and 'Shadow Blur' sliders for the sticker's outline. Upon clicking 'Apply,' the effects would be applied specifically to the chosen masks.

Pixel

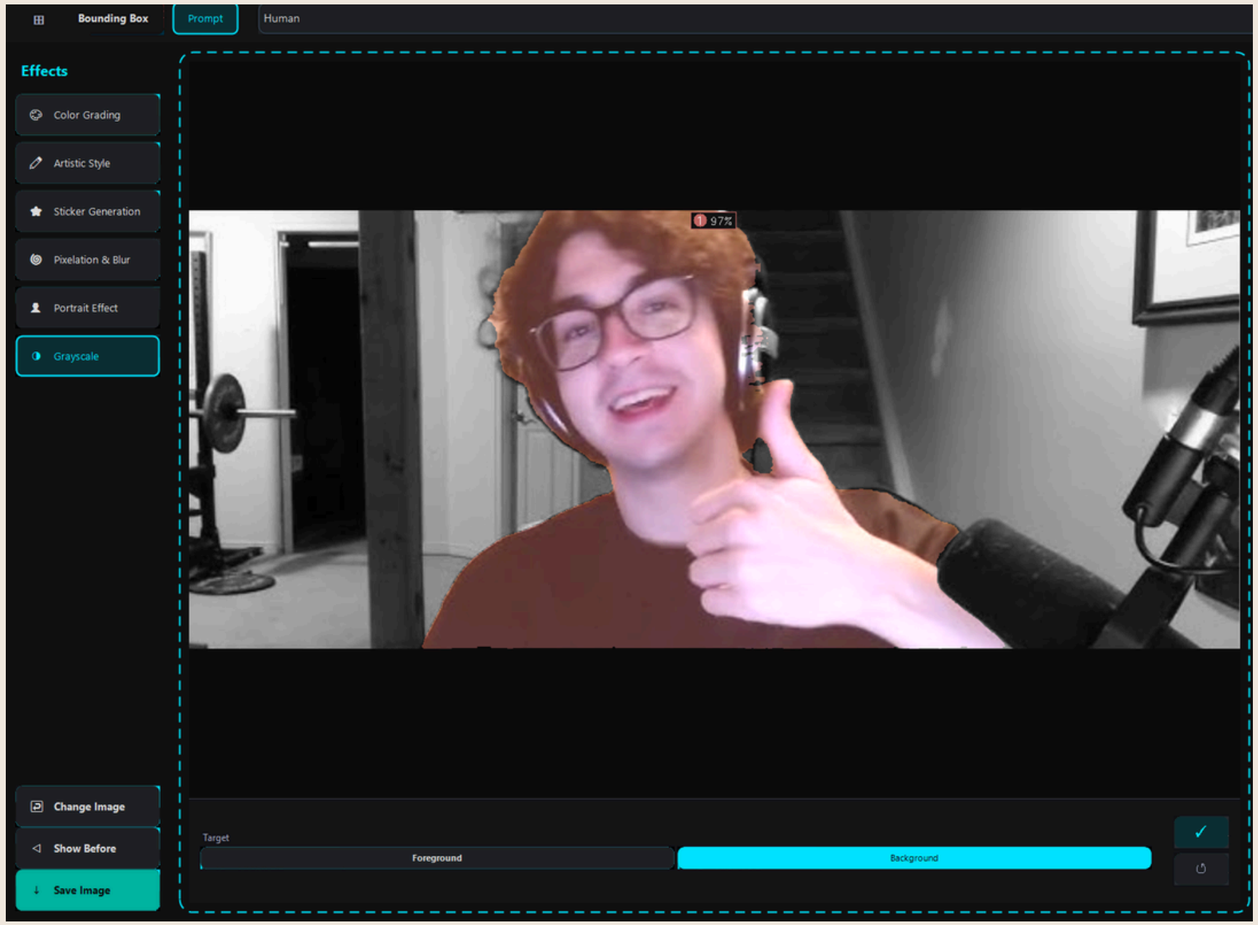


Users can select between 'Pixelate' or 'Blur' and adjust the 'Intensity' slider to control the level of pixelation or blur applied to the image. Upon clicking 'Apply,' the effects would be applied specifically to the chosen masks.

Portrait

Users can adjust the 'Blur Strength' to control the intensity and the 'Edge Feather' to smooth the transition between the mask and the rest of the image. Upon clicking 'Apply,' the effects would be applied specifically to the chosen masks."

Grayscale

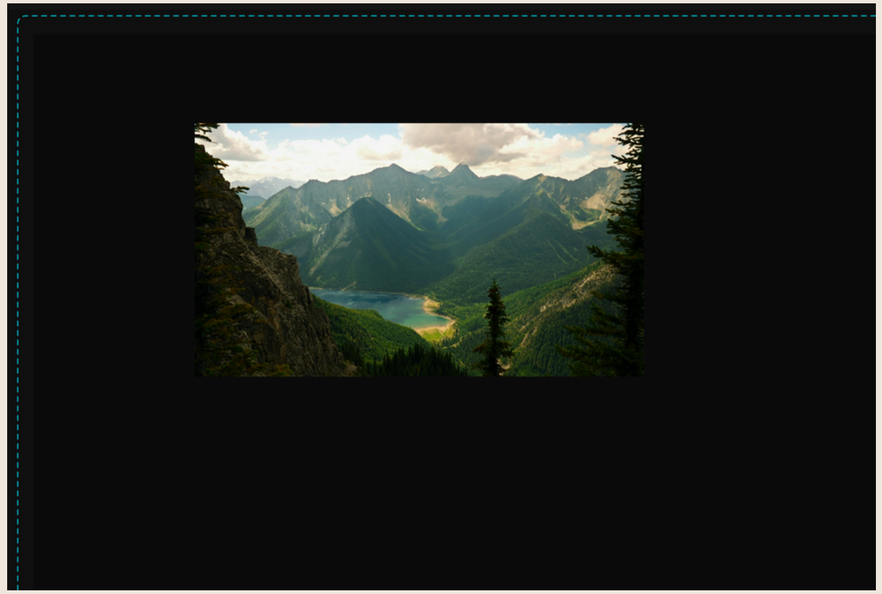


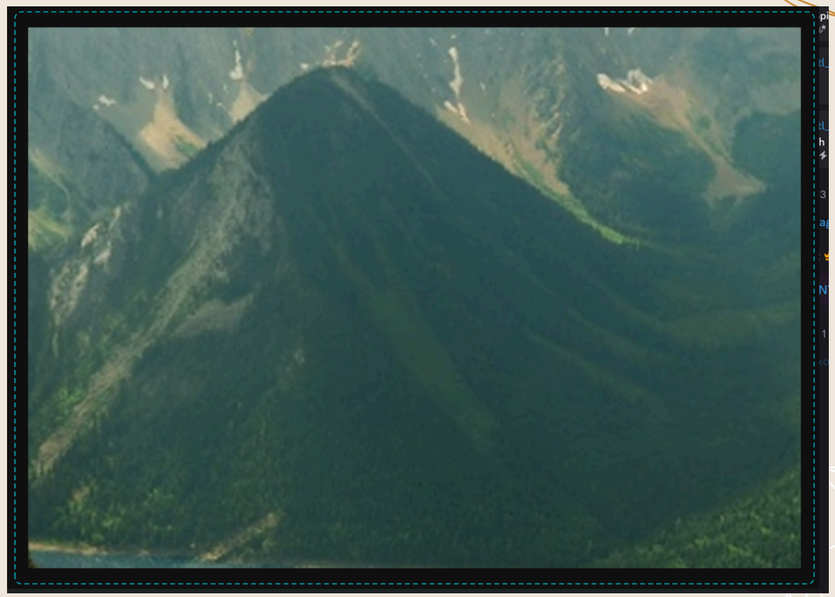
Users can select the target mask and, upon clicking 'Apply,' it will apply the grayscale specifically to the chosen mask to remove color information while preserving brightness.

Additional Features

Users can compare the edited image to the old one by clicking the “Show Before” button. While also being able to confirm to keep the effect after the effect is applied. They can additonally save the image and change it to edit another one.

Additional Features





Users can zoom in on the image to view details more clearly. They can also pan around the image using the scroll button, allowing them to move across different areas of the image while zoomed in.

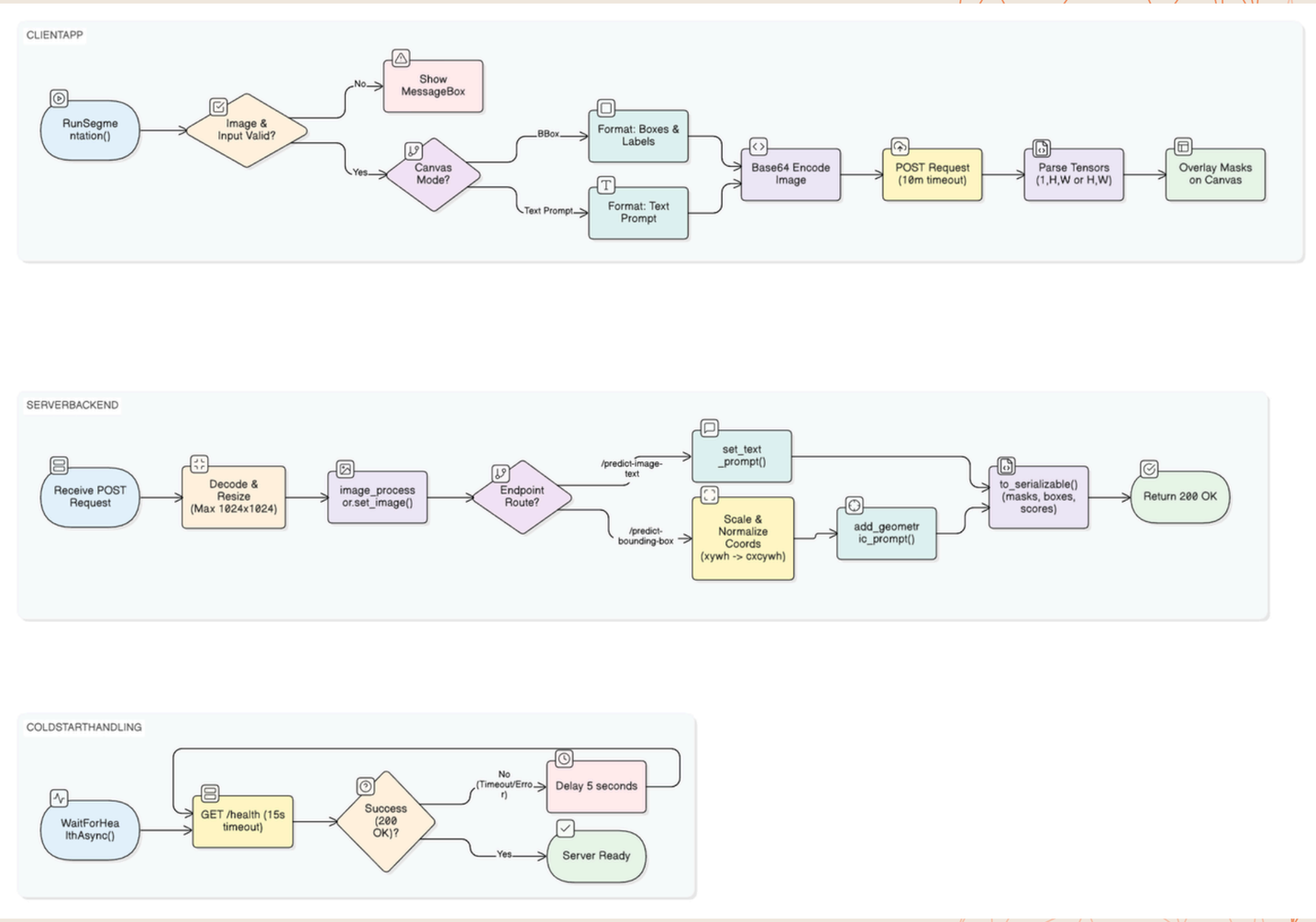
Methodology: SAM3 Implementation

- Server Hosting: Since SAM3 is a very large model and cannot run efficiently on most laptops, it is hosted on Lightning.ai servers using an NVIDIA T4 GPU for faster processing.
- The server uses a FastAPI backend with two main API endpoints:
- /predict-image-text – performs segmentation using a text prompt.
- /predict-bounding-box – performs segmentation using a bounding box input.
- The server experiences a cold start delay of about 5–6 minutes when it is first activated, as the model needs time to load into GPU memory and has a timeout to 10 minutes.

FLOWCHARTS

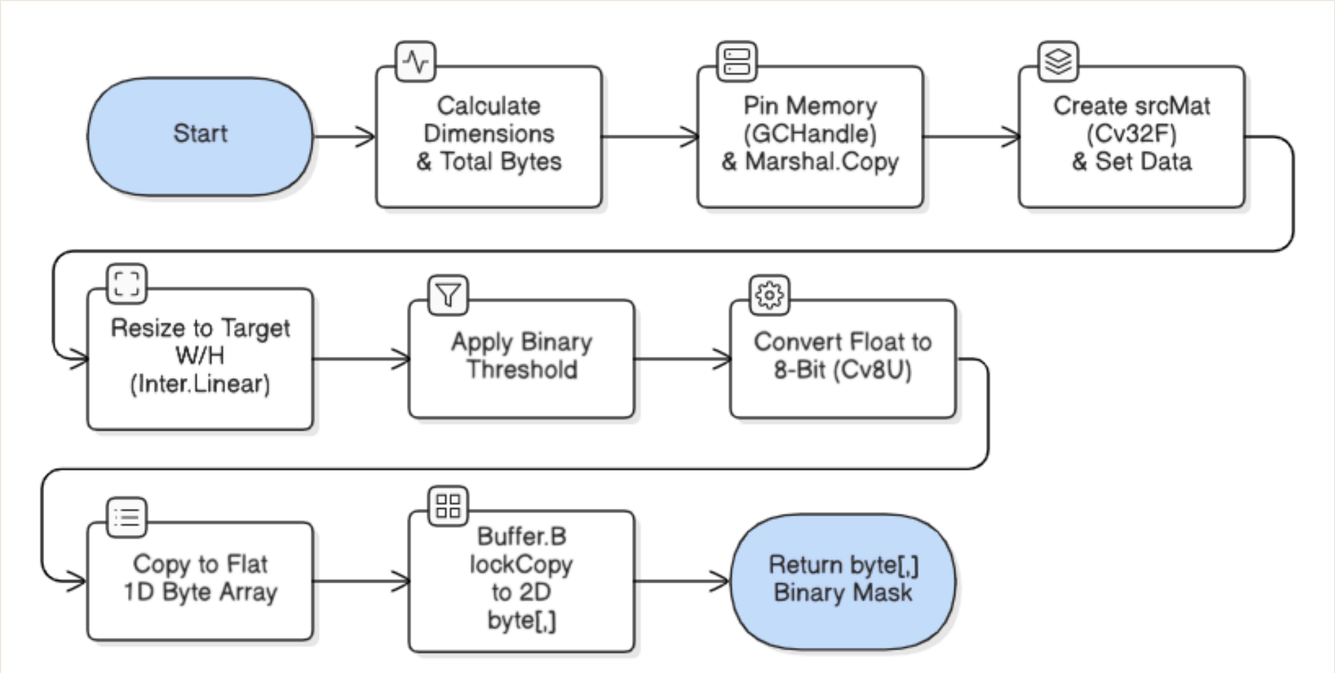
API Integration & Client Architecture

- On the frontend, when Segment button is clicked, it checks whether the image selected is valid
- The image is converted into Base64 string to send to the server and a POST request is made
- The server receives the POST request and decodes and resizes the Base64 image
- It then generates the mask and packages it in the JSON format
- The ColdStartHandling ensures that the server is ready before the client can send data to it by pinging the GET /health endpoint until it receives a 200 OK



FLOWCHARTS

Resize and Threshold Mask

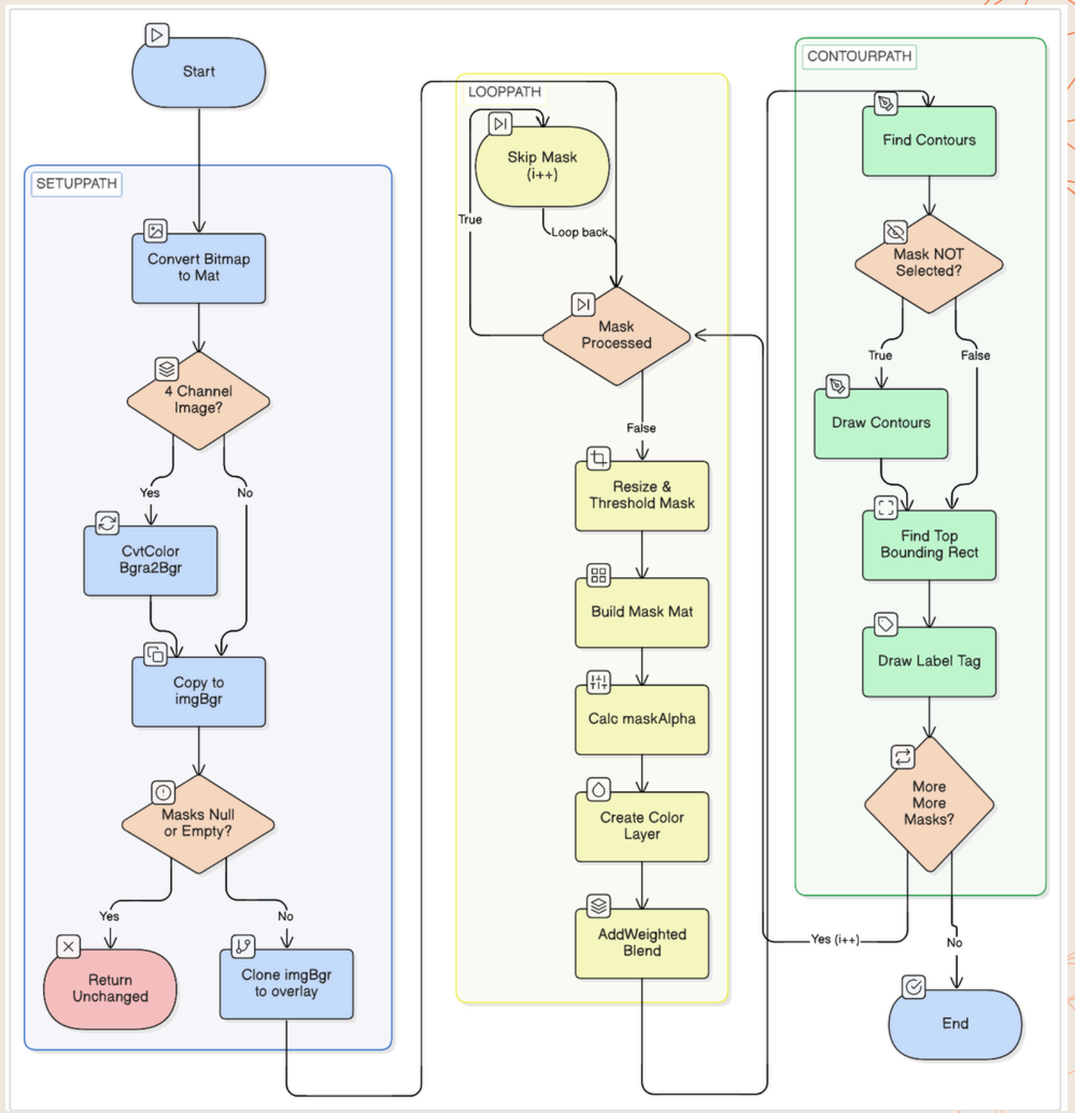


- After the app receives the masks, it calculates the number of bytes in the array and pins it in place using Marshal.copy
- The array is wrapped in a Cv32F mat and cv.resize is used to adjust the mask to the input image size
- A binary threshold is then used to convert the mask from a probability mask to a binary one
- The mask is converted into Cv8U and is returned as a byte [H, W]

FLOWCHARTS

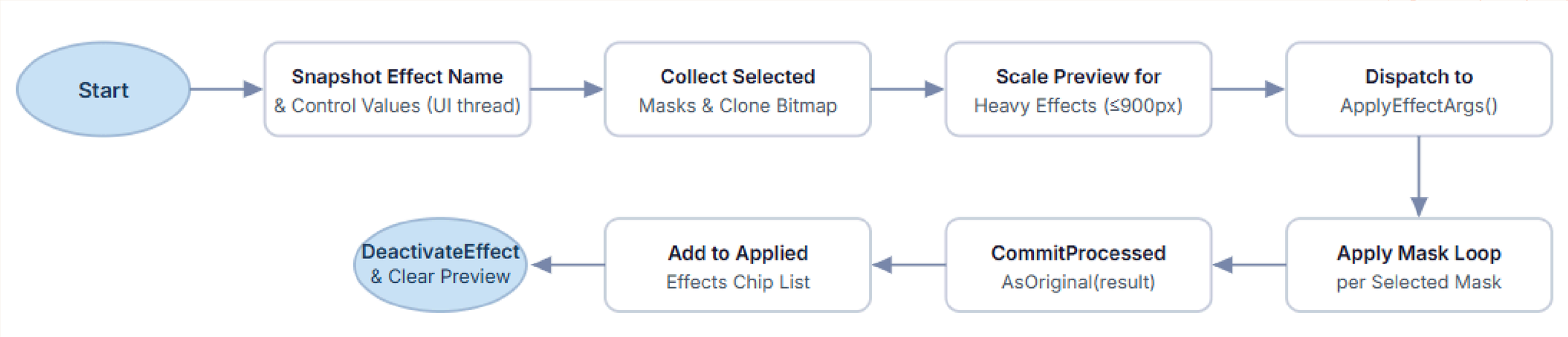
Overlay Masks

- The image is converted to a BGR Mat and cloned into an overlay; if no masks are provided, the original is returned unchanged
- For each mask, ResizeAndThresholdMask() and BuildMaskMat() prepare a binary mask Mat, then the mask color is blended onto the overlay using AddWeighted() with reduced alpha for unselected masks
- FindContours() extracts the mask border; unselected masks get a colored outline drawn via DrawContours()
- A label tag is positioned above each mask showing the mask number in a colored circle and a confidence score percentage via PutText()
- The final overlay Mat is converted back to a Bitmap and returned



FLOWCHARTS

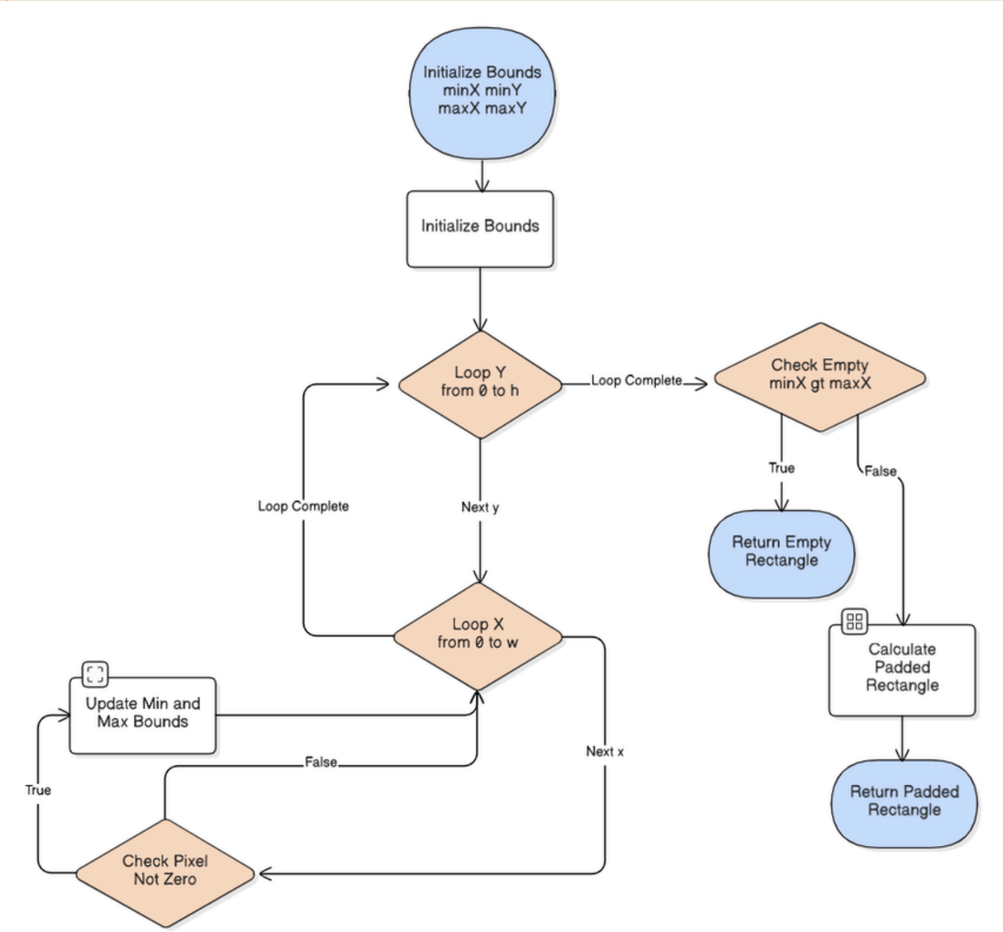
Applying Effects



- The active effect name and all control values (sliders, toggles, colors) are snapshotted from the UI thread before any async work begins
- Selected masks are collected from the canvas and the source bitmap is cloned to avoid mutating the displayed image
- For heavy effects (Artistic, Portrait, PixelBlur), the source is downscaled to a max of 900px on the longest side for preview, with kernel/block sizes scaled proportionally to match the full-resolution look

FLOWCHARTS

MaskBoundingRect

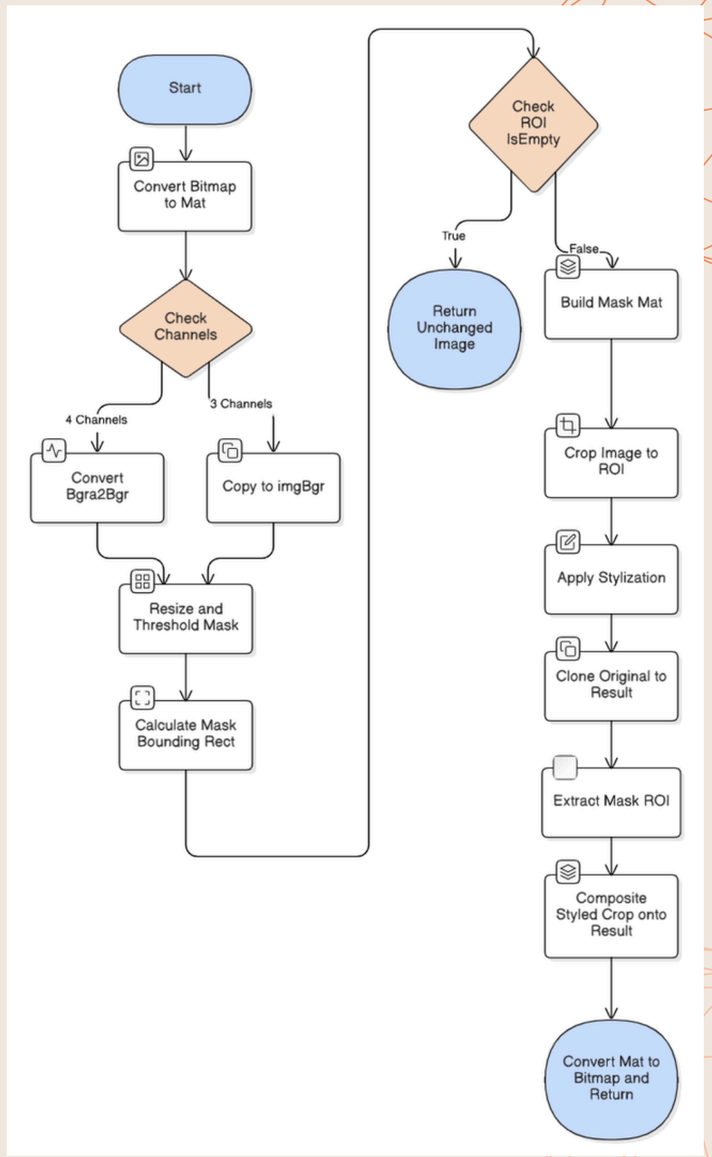


- The minimum x,y values start at the maximum possible values
- The maximum x,y values start at 0
- The nested loops search for pixels that are part of the masked selection
- When an active pixel is found, if the pixel is at a point further than min X or max Y, it will become the new min X and max Y
- A buffer of around 8 pixels is added to the bounding box and is added to the current x and y values
- The coordinates are then returned

FLOWCHARTS

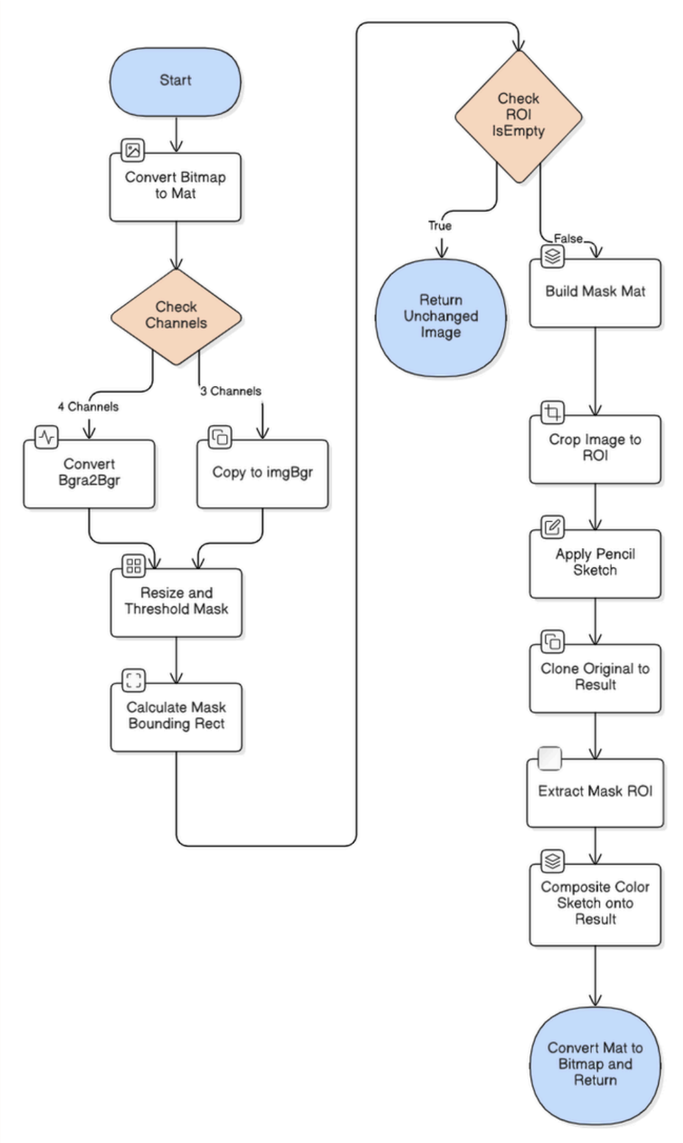
Stylization Effect

- The user inputs an image that is then converted into 3 channel image
- The mask is converted into the OpenCV format
- A clone of the original image is made
- A region of interest is calculated and the image is cropped to that specific area
- cv2.Stylization is applied on the cropped area
- The processed image is combined on top of the original image



FLOWCHARTS

Pencil Sketch Effect

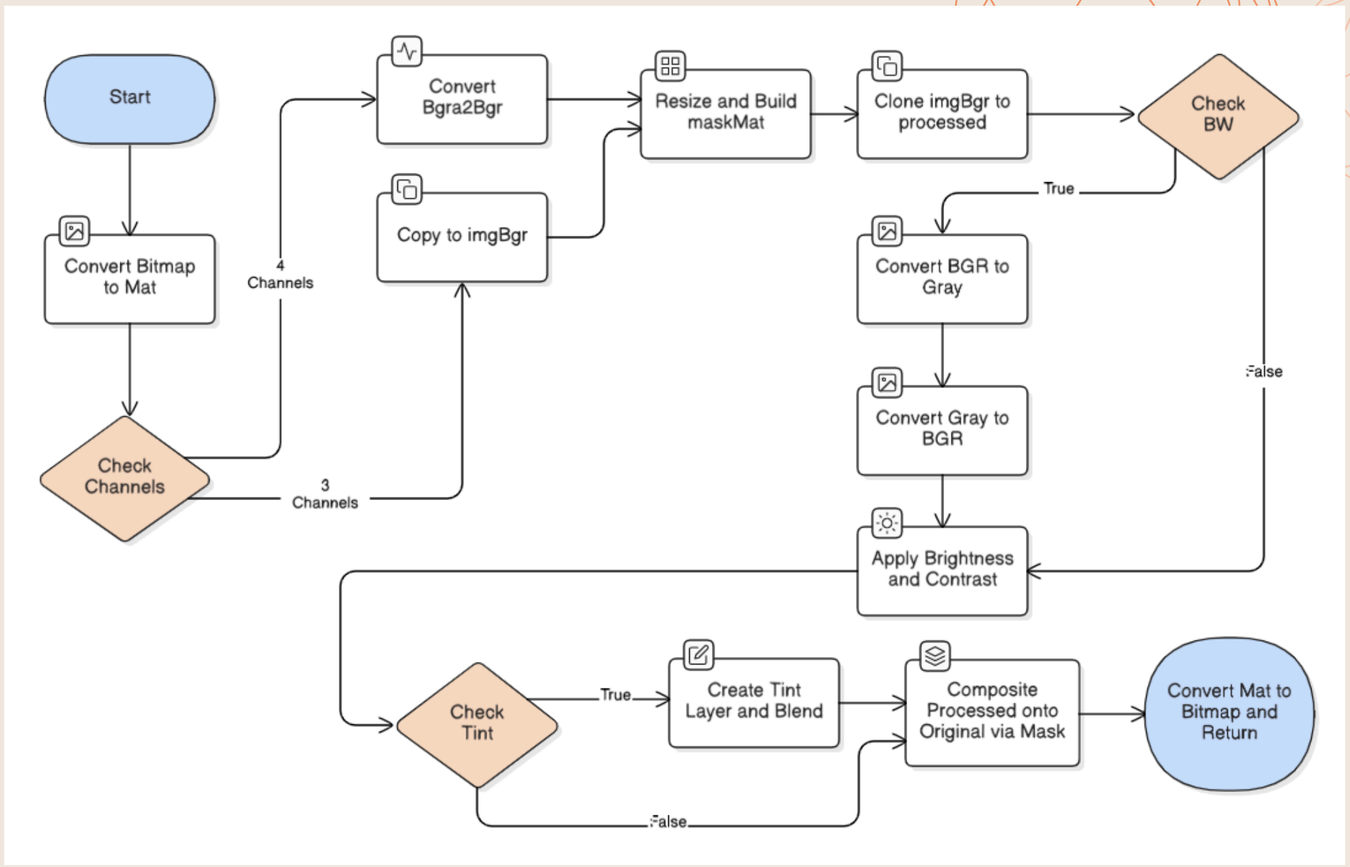


- The user inputs an image that is then converted into 3 channel image
- The mask is converted into the OpenCV format
- A clone of the original image is made
- A region of interest is calculated and the image is cropped to that specific area
- cv2.PencilSketch is applied on the cropped area and returns a pencil sketch image in black and white and color
- The color sketch version of the image is combined on top of the original image

FLOWCHARTS

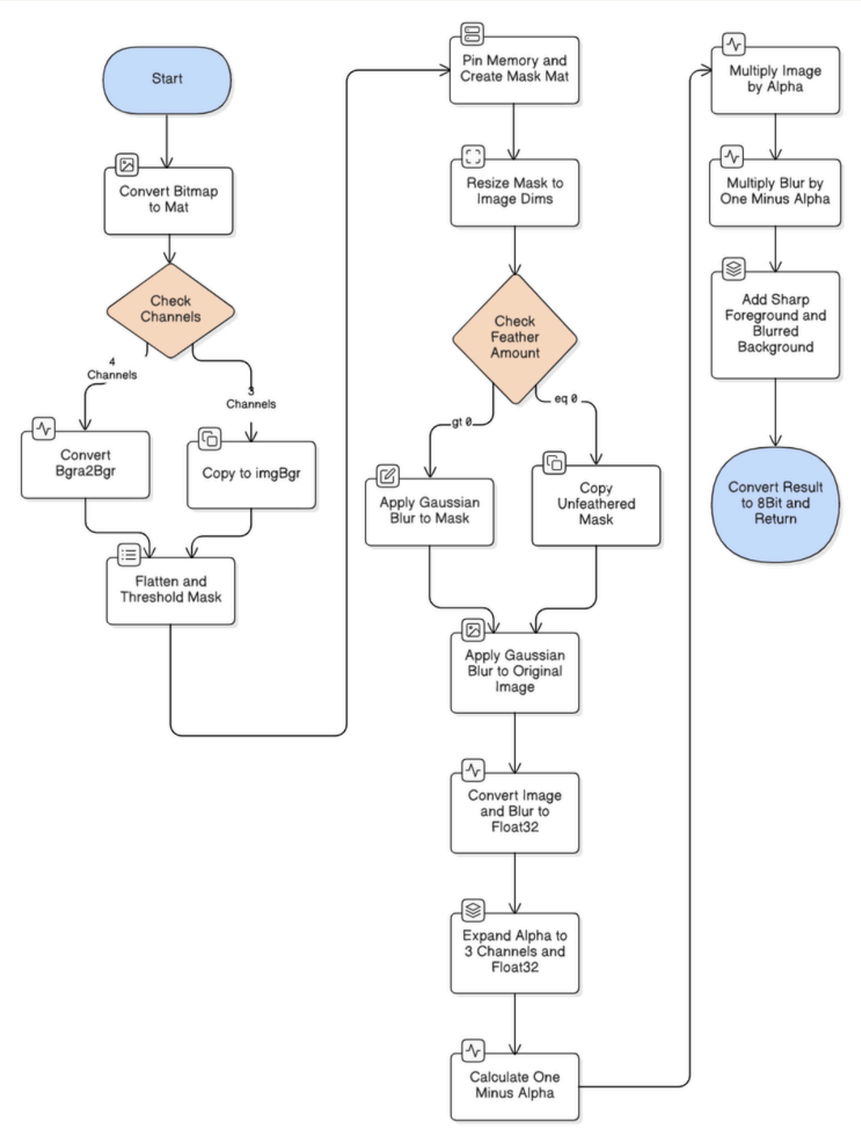
Color Grading

- User inputs an image which is converted into a 3 channel image
- The mask is converted into the OpenCV format
- Effects are applied on a clone of the image
- If black and white is selected, the image is turned to grayscale and back to RGB
- cv2.ConvertScaleAbs is used to adjust brightness and contrast
- If the user enables tinting, a solid layer of the color selected by the user
- cv2.AddWeighted is used combine the image with the tint
- cv2.CopyTo is used to apply the changes made on the cloned image to the mask of the original image



FLOWCHARTS

Portrait Effect

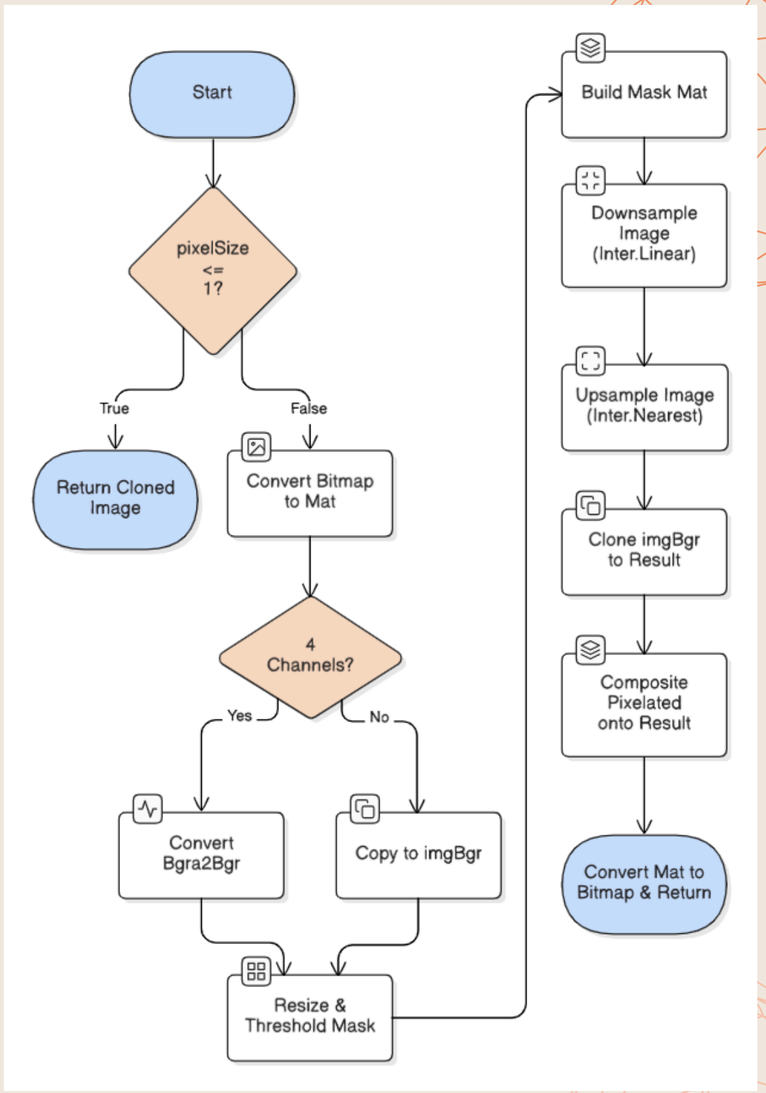


- The user inputs an image that is then converted into 3 channel image
- A float mask is created and resized using cv2.Resize to fit the resolution of the image
- If feathering is enabled, a gaussian blur is applied to the image to smooth out the edges of the selection
- A gaussian blur, with the blur strength selected by the user using a slider, is applied to a copy of the original image
- The original and blurred images are converted into float32 and converted into BGR
- Everything is combined together using the formula:
  - result = sharp * alpha + blurred * (1 - alpha)

FLOWCHARTS

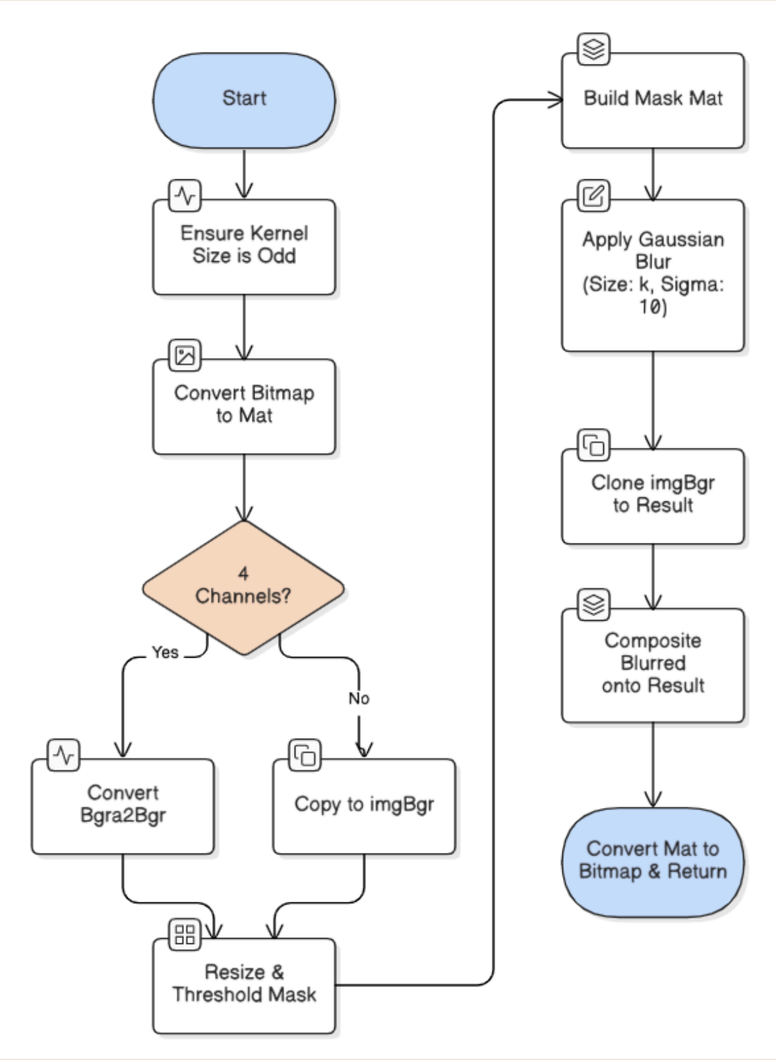
Pixelation Effect

- The user inputs an image that is then converted into 3 channel image
- The mask is converted into the OpenCV format
- A copy of the image is shrunk to the size of the amount the of the user's input
- That copy is then upscaled back to its original size but with the interpolation method set to nearest to emulate the pixelation effect
- The image and the copy are then combined together into the final result



FLOWCHARTS

Blur Effect

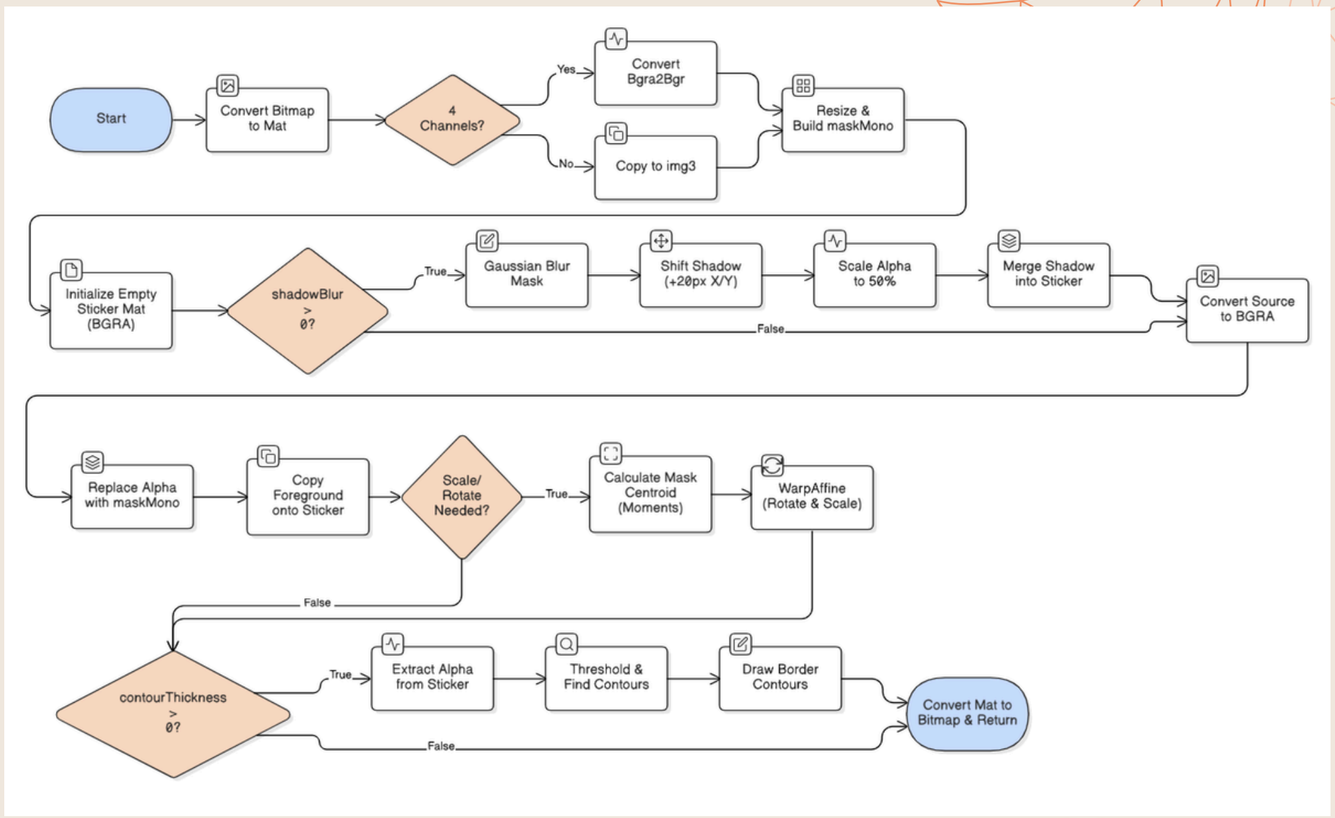


- The user inputs an image that is then converted into 3 channel image
- The mask is converted into the OpenCV format
- A gaussian blur is applied to a copy of the original image using cv2.gaussianblur
- The blurred image is combined with the original

FLOWCHARTS

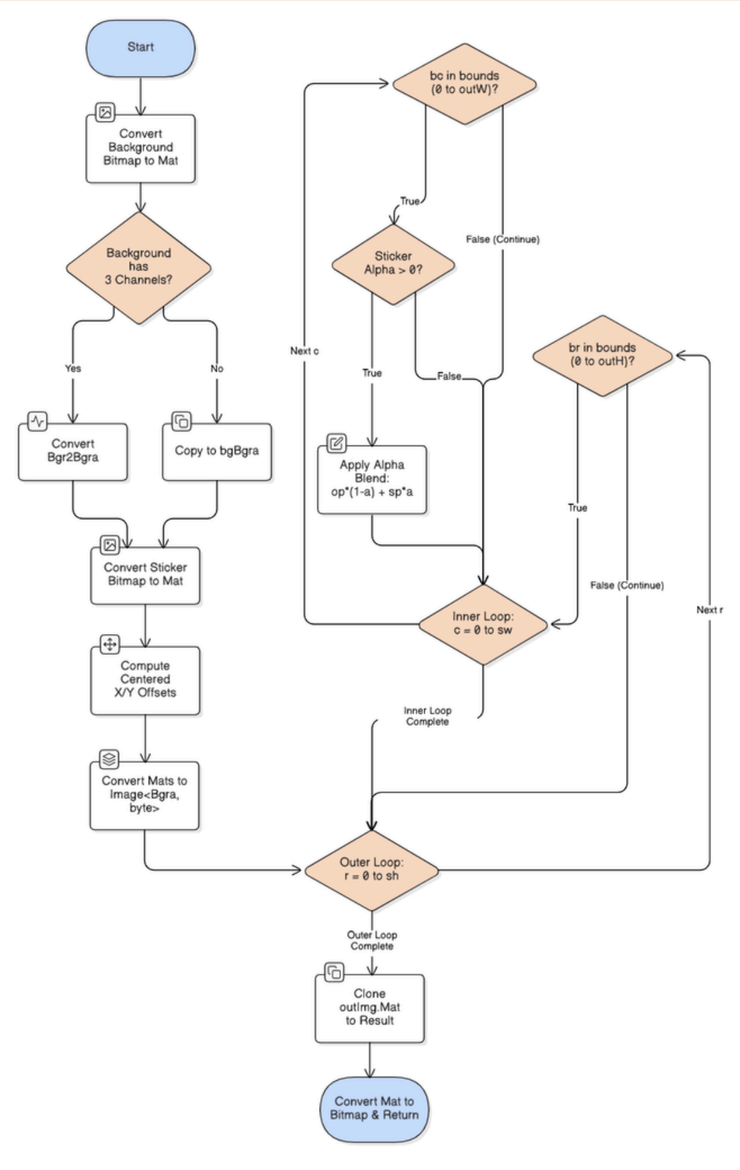
Extract Sticker

- The user inputs an image that is then converted into 3 channel image
- The binary black and white mask is converted into the OpenCV format
- A transparent sticker canvas of the same size as the image is created
- To apply a shadow, gaussian blurring is applied to the mask
- cv2.WarpAffine shifts the blurred mask to the right and the opacity is halved
- It is then merged together BGRA image
- The masked image is extracted and is merged with the shadow in the sticker layer
- If the user enables rotation, cv2.Moments create a rotation matrix
- To draw the contour, cv2.FindContours and cv2.DrawContours is used to draw the desired border thickness



FLOWCHARTS

Composite Sticker

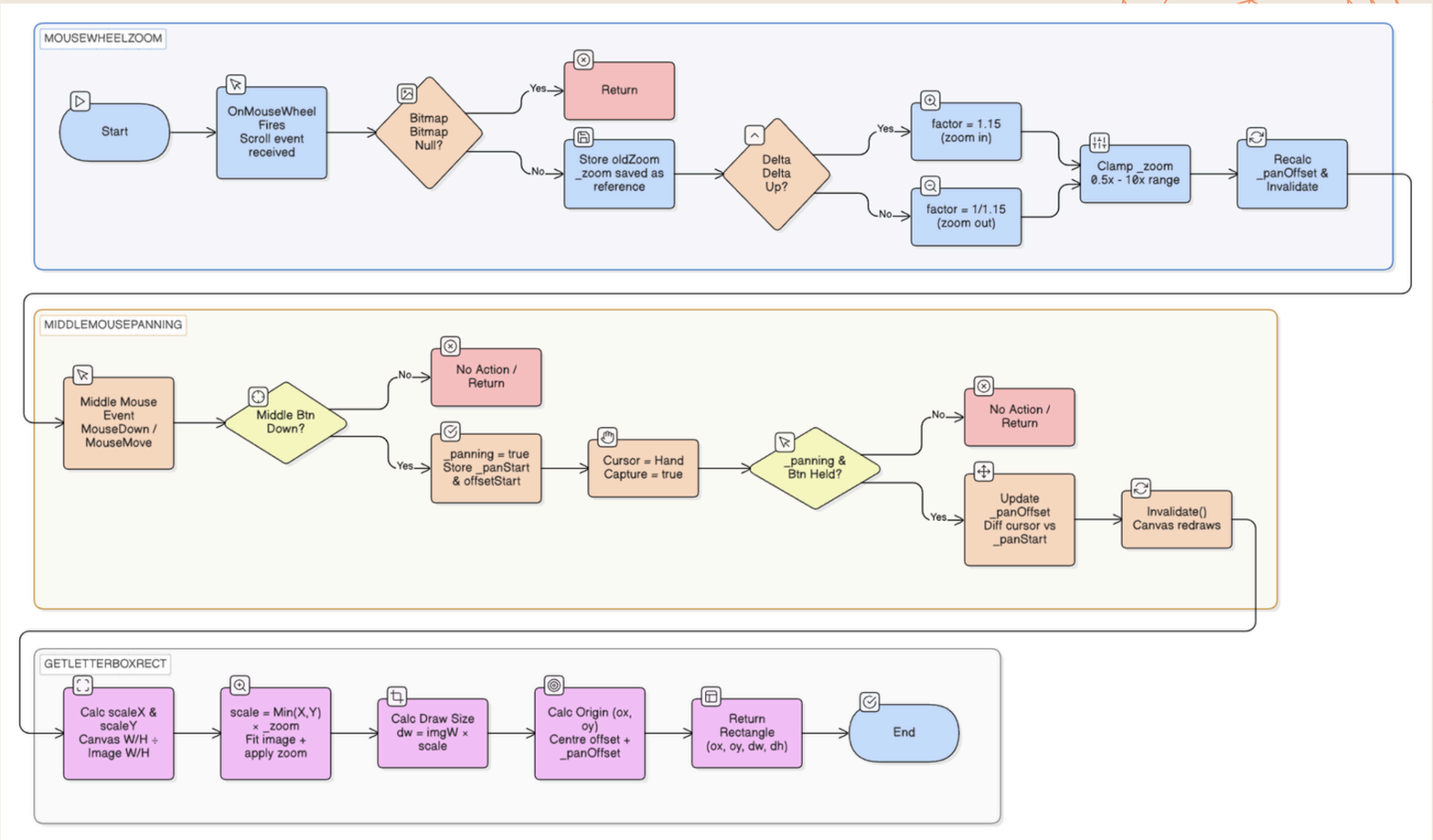


- Converts the background into a 4 channel BGRA image
- It takes the previously extracted sticker and places it in the middle of the background by subtracting the sticker's height and width by the background's height and width and dividing by 2
- It then checks each pixel to apply alpha blending to pixels that aren't transparent using the formula:
    - Output = Background x (1 - alpha) + Foreground x alpha
- The output is cloned and returned in the Bitmap format

FLOWCHARTS

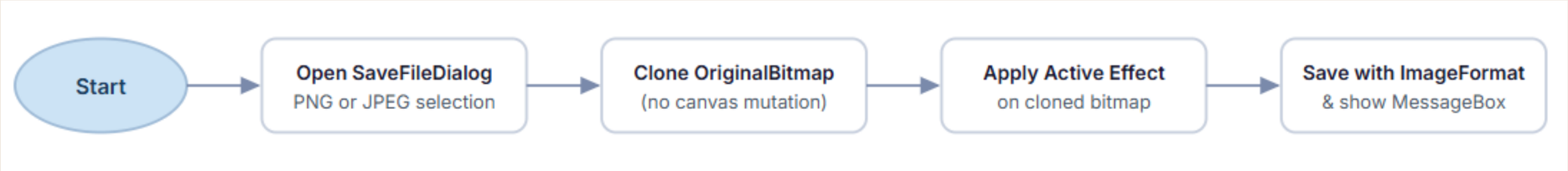
Zooming/Panning

- Zoom and pan state are tracked using a _zoom float (default 1.0), a _panOffset PointF, and a _panning bool
- Scrolling the mouse wheel scales the zoom by a factor of 1.15x in or out, clamped between 0.5x and 10x to prevent over-zooming
- The zoom is cursor-anchored — the pan offset is recalculated on each scroll so the image scales toward the mouse position rather than the canvas center
- Holding and dragging the middle mouse button activates panning, storing the initial cursor and offset positions as a reference and setting the cursor to a hand icon
- While panning, the offset is updated each MouseMove by diffing the current cursor position against the stored drag start point, then the canvas is invalidated to redraw
- GetLetterboxRect() calculates the final image draw rectangle by fitting the image to the canvas using the smaller of the X or Y scale factors, then applying _zoom and _panOffset to position it correctly



FLOWCHARTS

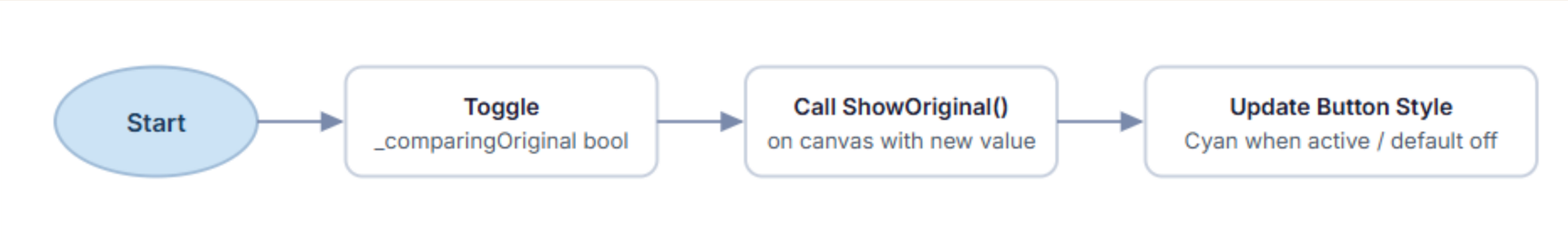
Saving Images



- A SaveFileDialog is opened letting the user choose between PNG and JPEG output formats
- The current OriginalBitmap is cloned so the saved output does not mutate the working canvas
- If there is an active effect with selected masks, the effect is applied on top of the cloned bitmap before saving
- The bitmap is saved using System.Drawing.Imaging.ImageFormat and a confirmation MessageBox is shown on success, or an error message on failure

FLOWCHARTS

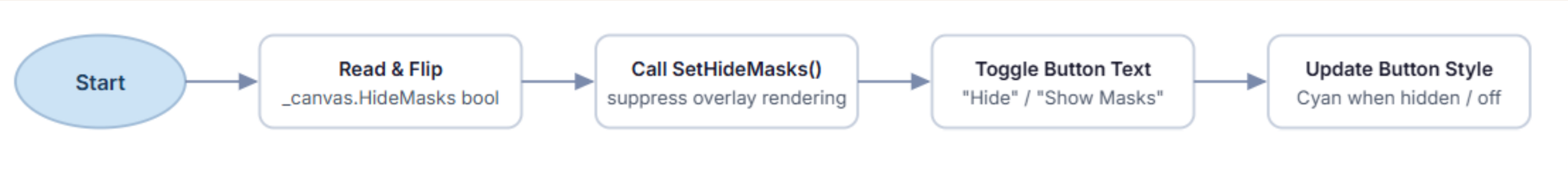
Comparing to Original Image



- A boolean _comparingOriginal is toggled on each button click
- _canvas.ShowOriginal() is called with the toggled value, instructing the canvas to swap its display between the processed bitmap and the original
- The Compare button turns cyan with a dark foreground when active, and reverts to the default dark button style when inactive

FLOWCHARTS

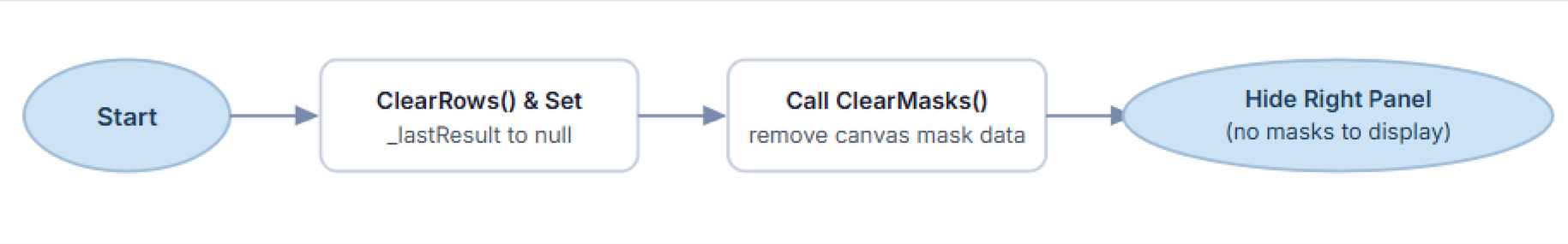
Hiding masks



- A boolean is read from _canvas.HideMasks and flipped on each button click
- _canvas.SetHideMasks() is called with the new value, instructing the canvas to suppress mask overlay rendering
- The button text toggles between ”Hide Masks” and ”Show Masks”, and the button turns cyan when masks are hidden to signal the active state

FLOWCHARTS

Clearing masks



- The mask list control is cleared via _maskList.ClearRows() and _lastResult is set to null, discarding the last segmentation result
- _canvas.ClearMasks() is called to remove all mask data from the canvas
- The right panel is hidden since it has no masks left to display

Tools Used

TECH STACK

Visual Studio Code, Python - Used for prototyping openCV functions

C# .NET 8 and Visual Studio - Used for the client application

EmguCV - a C# wrapper for openCV used for image processing

Meta SAM3 - used to generate clean masks to be used for further image editing

Lighting.ai - Used for the server

Limitations

1. As a large model, SAM3 requires significant computational resources to run effectively.
2. The model can struggle with highly complex user prompts and small areas.
3. Segmentation may not be 100% accurate in all scenarios.
4. The application is dependent on a stable internet connection to function.
5. The server takes around 5-6 minutes to cold start and can take 1-2 minutes to segment.

Conclusion

- Successfully built a photo editing app that incorporates SAM3 to output a mask
- Ensured that the masks created are accurate and robust
- Created a user friendly UI that allows the user to freely select target objects and apply a variety of effects onto it
- Successfully applied OpenCV functions to edit the segmented images