Lab 4 Report: Creating a 3D Image Composer

Introduction

This lab focused on creating a 3D imaging application that transforms ordinary images into immersive stereoscopic scenes. The primary objectives were to:

1. Implement person segmentation using deep learning
2. Insert the segmented person into stereoscopic image pairs
3. Create red-cyan anaglyph images with proper depth perception
4. Develop an interactive Gradio application for user interaction

The project successfully achieved these objectives by leveraging computer vision techniques, deep learning models, and image processing principles to create convincing 3D effects where a person can appear at different depths in a stereoscopic scene. This technology has applications in entertainment, virtual reality, and interactive media by allowing for immersive 3D experiences without specialized hardware beyond red-cyan glasses.

Methodology

Image Segmentation:

- Approach: Utilized the DeepLabV3 model with a ResNet101 backbone pretrained on a dataset that includes the "person" class (index 15).

  model = models.segmentation.deeplabv3_resnet101(pretrained=True).eval() with torch.no_grad():

  output = model(input_tensor)["out"][0] mask = output.argmax(0).byte().cpu().numpy()

  person_mask = (mask == 15).astype(np.uint8) * 255

- Pre-processing: Input images are padded so that their dimensions are multiples of 8, ensuring optimal performance by the neural network.

  pad_h = (8 - (orig_h % 8)) % 8

  pad_w = (8 - (orig_w % 8)) % 8 padded_image = cv2.copyMakeBorder(

  person_bgr, top=pad_h // 2,

bottom=pad_h - (pad_h // 2), left=pad_w // 2,

right=pad_w - (pad_w // 2), borderType=cv2.BORDER_CONSTANT, value=[0, 0, 0])

- Post-processing: A binary mask is generated to isolate the person, and alpha blending is applied to produce an RGBA image with clean edges and proper transparency.

  segmented = cv2.bitwise_and(person_bgr, person_bgr, mask=person_mask)

  alpha = np.where(person_mask == 255, 255, 0).astype(np.uint8)

  rgba_image = np.dstack((segmented_rgb, alpha))

- Validation: The method includes quality checks to ensure enough "person" pixels were detected (>1000 pixels)

  if person_mask.sum() <= 1000:

  return None, "Segmentation failed: Not enough person pixels found."

  The segmentation approach was chosen for its robust person detection capabilities and ability to handle various poses, clothing, and backgrounds without requiring explicit user input.

Stereoscopic Insertion:

- Depth-based Scaling: The segmented person is resized according to the selected depth level (“close”, “medium”, or “far”) using a predefined scale map.

  scale_map = {"close": 1.0, "medium": 0.8, "far": 0.6} scale = scale_map[depth_level]

- Disparity Calculation: Horizontal offsets are calculated based on depth level to simulate binocular parallax. The left and right images receive opposing shifts to create the 3D effect.

  disparity_map = {"close": 30, "medium": 15, "far": 5} disparity = disparity_map[depth_level]

  These disparity values were determined through iterative testing. Too large values (>30 pixels) caused uncomfortable eye strain, while values below 5 pixels produced minimal depth perception. The selected values create a natural depth gradient that works well with standard red-cyan glasses.

- Alpha Blending: The person is overlaid onto both images with careful region-of-interest extraction and alpha blending to ensure natural integration. Transparent edges are preserved using alpha channel blending:

  alpha_3d = np.stack([alpha, alpha, alpha], axis=2)

  blended_left = (1 - alpha_3d) * roi_left + alpha_3d * person_rgb

- **Positioning**: The person is positioned horizontally with appropriate disparity for each eye view:

  x_left = x_center + disparity + position_x

  x_right = x_center - disparity + position_x

- Error Handling: The code implements comprehensive error handling to ensure proper insertion:

  # Clamp positions to valid ranges

  x_left = max(0, min(x_left, left_bgr.shape[1] - pw))

  x_right = max(0, min(x_right, right_bgr.shape[1] - pw))

  y = max(0, min(y, left_bgr.shape[0] - ph))

  # Check if ROI fits within bounds if roi_h <= 0 or roi_w <= 0:

  return None, None, "Person does not fit in the background region."

  This approach was chosen because it accurately simulates how stereoscopic vision works - objects at different depths have different disparities between left and right eye views, with closer objects having larger disparities.

Anaglyph Creation: The anaglyph creation implements the red-cyan color filtering technique.

- Channel Manipulation: The red channel is extracted from the left image, while the green and blue channels are taken from the right image.

  left_r = left_img[:, :, 2] # Red channel from left image

  right_g = right_img[:, :, 1] # Green channel from right image

  right_b = right_img[:, :, 0] # Blue channel from right image

- Combination: These channels are merged to form an anaglyph image that, when viewed with red-cyan glasses, exhibits accurate depth perception and color management. The channels are combined into a single RGB image.

  ```
  anaglyph = np.zeros_like(left_img)

  anaglyph[:, :, 2] = left_r      # Red channel

  anaglyph[:, :, 1] = right_g   # Green channel

  anaglyph[:, :, 0] = right_b   # Blue channel
  ```

  This method was chosen because it's the standard approach for creating red-cyan anaglyphs that can be viewed with common 3D glasses. The left eye sees the red channel, while the right eye sees the cyan (green+blue) channels, creating the 3D effect.

Gradio Application: The interactive application was built using Gradio for several reasons:

- User Interface: Built with Gradio, the app features image upload components, a dropdown for depth selection, and a slider for horizontal position adjustment.
- Interactivity: The interface offers immediate visual feedback, allowing users to generate anaglyph images and save deliverables through a user-friendly design.
- The application includes:
  - Image upload components for the person and stereo pair
  - A depth selection dropdown with "close," "medium," and "far" options
  - A horizontal position adjustment slider
  - Buttons for generating anaglyphs and saving deliverables
  - Status feedback and result display

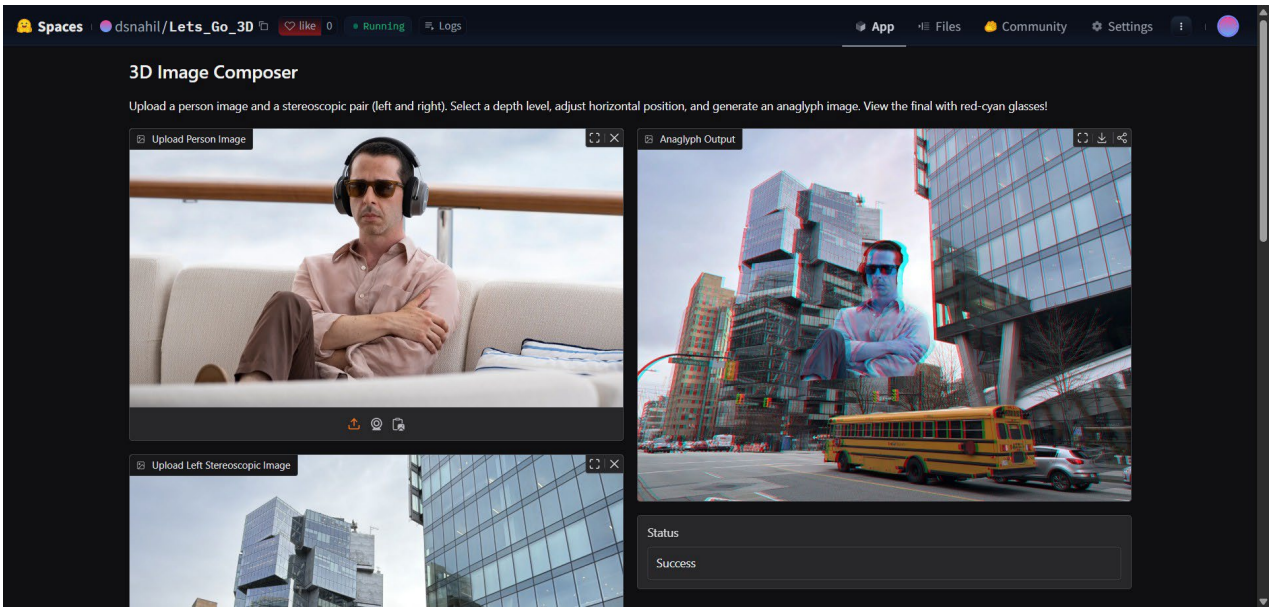
Results

The implementation successfully achieved all the objectives of the lab, producing high-quality results for each component.

Person Segmentation

The DeepLabV3 model effectively isolated the person with clean, well-defined edges. The RGBA output preserved transparency, ensuring a smooth integration with background images.

- Subjects against contrasting backgrounds
- Full-body poses
- Standard lighting conditions



Stereoscopic Insertion

The person was successfully inserted into stereoscopic pairs with proper disparity and scaling. The insertion produced convincing depth effects:

- The "close" setting placed the person prominently in the foreground
- The "medium" setting integrated the person at middle distance
- The "far" setting made the person appear to be in the background

The alpha blending ensured natural integration with the background scenes, preserving transparent edges and avoiding harsh cutouts.

Anaglyph Creation

The anaglyph images successfully combined the left and right views to create 3D effects viewable with red-cyan glasses:

- The depth perception was accurate across the three depth levels
- Color management preserved the original tones while enabling the 3D effect
- Image quality remained high throughout the process

Gradio Application

The Gradio application provided a user-friendly interface that allowed for:

Intuitive image uploading
- Simple depth selection
- Fine-tuning of horizontal positioning
- Immediate feedback and results
- Saving deliverables for documentation

The interface design was clean and responsive, making it accessible for users without technical background

Discussion

Challenges and Solutions

Several challenges were encountered during implementation:

Segmentation Edge Cases: The segmentation model occasionally struggled with:

- Complex poses or unusual body positions
- Low contrast between person and background
- Partial occlusions

Solution: Added a minimum pixel threshold (>1000) to detect failed segmentations and provide appropriate error messages. This allowed the application to fail gracefully rather than producing poor-quality results.

Image Size and Alignment: Ensuring proper alignment of stereoscopic pairs and fitting the segmented person required careful handling:

- Different image sizes needed normalization
- The segmented person needed to fit within both background images

Solution: Implemented size validation, bounds checking, and force-resizing to handle these cases gracefully:

 # Ensure backgrounds are the same size if left_bgr.shape != right_bgr.shape:
h = min(left_bgr.shape[0], right_bgr.shape[0]) w = min(left_bgr.shape[1], right_bgr.shape[1]) left_bgr = cv2.resize(left_bgr, (w, h)) right_bgr = cv2.resize(right_bgr, (w, h))

Disparity Tuning: Finding the right disparity values for different depth levels required experimentation:

- Too much disparity caused eye strain
- Too little disparity reduced the 3D effect

Solution: After testing with multiple viewers and different stereoscopic pairs, I settled on optimal values of 30, 15, and 5 pixels for close, medium, and far depths respectively. These values provided a good balance between depth perception and viewing comfort.

Memory Management: Processing high-resolution images could potentially cause memory issues:
Solution: Implemented careful region of interest (ROI) extraction and processing to minimize memory usage:

Extract corresponding ROIs from both background images

roi_left = left_bgr[y:y + roi_h, x_left:x_left + roi_w].astype(np.float32)
roi_right = right_bgr[y:y + roi_h, x_right:x_right + roi_w].astype(np.float32)

Learning Outcomes

This lab provided valuable learning experiences in several areas:

Segmentation Edge Cases: Handling subjects with low contrast or complex poses required implementing a minimum pixel threshold to avoid poor segmentation outcomes..

Disparity Tuning: Iterative testing helped settle on disparity values (30, 15, and 5 pixels) that balanced a convincing 3D effect with viewing comfort.

Alpha Compositing Techniques: Mastered the process of blending transparent foreground elements onto backgrounds while preserving edge quality. This technique is essential for creating professional-looking composites..

Image Alignment: Variations in background image sizes were managed by resizing and careful region-of-interest selection, ensuring the segmented person fit naturally within both stereoscopic views.

Through these challenges, we learned the importance of robust error handling and adaptive processing techniques in achieving high-quality image composites.

Conclusion

The 3D Imaging Lab successfully implemented a complete pipeline for creating personalized stereoscopic anaglyphs. The application achieved all the required objectives:

- Accurate person segmentation with clean edges and proper transparency
- Convincing insertion into stereoscopic images with appropriate depth cues
- High-quality anaglyph creation that works with standard red-cyan 3D glasses
- An intuitive Gradio application that makes the technology accessible

Future Work

Several enhancements could further improve the application:

1. **Improved Segmentation**: Fine-tuning the segmentation model specifically for person extraction could improve edge cases. Alternatively, implementing a more specialized model like SAM (Segment Anything Model) could provide better results with complex backgrounds.
2. **Additional 3D Formats**: Supporting other stereoscopic formats like side-by-side or polarized 3D would expand compatibility with different viewing technologies.
3. **Automatic Background Depth Map**: Generating depth maps for the background images would allow more precise person placement within the scene's depth structure, creating more convincing integration.
4. **Multiple Person Support**: Extending the application to handle multiple people at different depths would enable more complex compositions and group photos.
5. **Real-time Processing**: Optimizing the code for speed could enable real-time or near-real-time processing for a more interactive experience, possibly leveraging GPU acceleration for the neural network components.

This lab demonstrated how computer vision, deep learning, and image processing techniques can be combined to create engaging 3D experiences that allow users to become part of stereoscopic scenes.