Computer Vision for Interactive Animated Drawing

Sky Cho and Anh Thach

***Abstract*— Computer Vision offers a wide range of applications, from face swapping to the development of augmented reality games. One notable application is the creation of interactive animated drawings. This application uses a trained model to identify and extract hand-drawn objects from photographs of paper sketches. This paper explores and describes how to utilize computer vision skills for animated drawings.**

I. Introduction

Everyone has, at some point as a child, imagined how their drawings might come to life in reality. With advancements in technology, this childhood dream has become a reality. The system utilized automatically animates children's drawings from photos, offering a fast, easy-to-use, and reliable solution for handling the unique variations in these artworks. It works through four main stages: figure detection, segmentation masking, pose estimation and rigging, and animation. By fine-tuning existing computer vision models, the system adapts well to the abstract nature of children's drawings. It also introduces a new motion re-targeting method that uses the "twisted perspective" often seen in kids' art to bring their characters to life in a fun and engaging way [1].

Fig. 1. Examples of Children drawings [1].

II. Background

Children's drawings have been studied for over a century to explore their thought processes, intellectual growth, and perceptions. These drawings often feature varied schemas, asymmetries, and proportions. They may include twisted perspectives, where body parts face different directions for clarity. Additionally, some parts might be omitted, or non-human features added, showcasing unique creativity. This uniqueness posed significant challenges for previous methods attempting to animate such drawings. Many of the previous methods required additional user input, such as annotated joint locations, segmentation masks, or motion specifications, while some required users to draw the same character in multiple poses—an ability beyond most children. Certain approaches were tailored to specific forms, such as anime characters, coloring-book figures, or human-like proportions, making them unsuitable for abstract children's art. Tools like Photo Wake-Up focused on creating 3D models for realistic figures, disregarding the unique and imperfect proportions of children's drawings. Similarly, frameworks like Monster Mash and Neural Puppet required consistent inputs or specialized training images, limiting their accessibility for amateur users. The model created by Smith et al. tries to address these limitations and focuses on 2D plane, aligning with the characteristics of children's drawings [1].

III. Related Work

There are also many relevant works that are needed for this system, focusing on some key areas [2].

- Art and Cognition:
  - Research highlights that drawing order reveals the organization and hierarchical representation of a scene.
  - Studies by Novick and Tversky propose that drawing order reflects schematization and conceptualization of objects.
  - Automated systems like AARON focus on final drawings rather than the temporal process of drawing, which remains underexplored.
- Writing Order Recovery:
  - Recovering writing order from handwriting images is well-studied, with writing often having well-defined rules.
  - Sketches are more complex due to dependencies across different parts, making the problem significantly harder.
- Non-Photorealistic Rendering (NPR):
  - Techniques for synthesizing line drawings from images and 3D models exist.
  - Some NPR methods attempt to mimic hand-drawing processes and can benefit from meaningful drawing order frameworks.
  - Challenges include stroke layering and overlapping details.
- Line Drawing Simplification:
  - Simplification methods reduce clutter by line omission or perceptual grouping.
  - Techniques assess line importance based on density, proximity, or parallelism and support applications like progressive drawing and level-of-detail representation.
- Sketch Understanding and Recognition:
  - Focus on recognition and beautification of line drawings using domain-specific knowledge (e.g., mathe-

matical symbols, mechanical designs).
  - These approaches are not designed for reconstructing the temporal order of strokes.
- Distinct Focus of the Work:
  - This work emphasizes reconstructing the temporal aspect of drawing strokes, which is a largely unexplored problem.
  - It distinguishes itself from prior efforts that optimize final images or explore shape representation.

IV. Methods

The methods employed in this study are adapted from the work of Smith et al. from the Meta Research Team. Their approach provides a robust framework for automating the animation of hand-drawn figures.

Starting with the photograph of the image, the animation process of humanoid drawings involves a structured pipeline composed of three key steps: human figure detection, segmentation, pose estimation, and animation [1]. This pipeline ensures that hand-drawn figures can be animated automatically without extensive manual annotation.

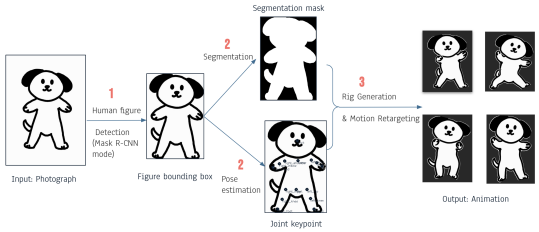

Fig. 2. Overview of the animation pipeline used for automating hand-drawn figure animations. The pipeline consists of four main stages: figure detection, segmentation, pose estimation, and animation.

A. Figure detection

The first step in the animation process involves detecting the human figure within the drawing and establishing a bounding box around it [1]. This bounding box serves as the foundation for subsequent processing by isolating the figure from the background and any additional elements present in the image, such as decorations or overlapping objects. By narrowing the focus to the region of interest, this step ensures that the system analyzes only the relevant parts of the image, improving the accuracy of the subsequent stages [3].

To detect human figures within drawings, we employ Mask R-CNN, a state-of-the-art object detection model, with a ResNet-50+FPN backbone [1]. The model is initialized with weights pretrained on the MS-COCO dataset, a large-scale semantic segmentation dataset [1], [4]. However, MS-COCO primarily consists of real-world photographs and lacks a category specifically for human figure drawings [1]. To address this limitation, the model is fine-tuned by freezing the backbone weights and attaching a new classification head designed to predict a single class: human figure [1]. This fine-tuning approach enables the model to adapt its object detection and segmentation capabilities from real-world images to artistic renderings, effectively bridging the domain gap between the pretrained dataset and the target task.

B. Segmentation

After obtaining the figure's bounding box, the image undergoes a segmentation process to generate the segmentation mask. This process employs an image processing-based approach to extract the mask (Fig. 3) [1].

1) **Starting Point: Bounding Box Detection**
   The segmentation process begins with the detection of a bounding box that encapsulates the raw image of the figure. This bounding box serves as the initial input for the segmentation process, ensuring that the subsequent steps focus on the relevant region of interest.
2) **Grayscale Conversion and Thresholding**
   Following the detection of the bounding box, the image within it is converted to grayscale. This step simplifies the image data by reducing it to a single color channel, making it easier to analyze. After grayscale conversion, adaptive thresholding is applied to enhance the contrast between different regions of the image. This converts the grayscale image into a binary image, where pixels are either black or white, based on their intensity values relative to a threshold. This step helps in separating the figure from its background and prepares the image for further processing [5].
3) **Dilating**
   To reinforce connections and solidify the shape of the figure, a dilating operation is performed using a 3×3 kernel. Dilation expands the edges of the figure slightly, ensuring that any minor gaps or breaks are filled. This step is essential for maintaining the structural integrity of the figure and preventing fragmentation during subsequent processing steps [6].
4) **Flood Filling**
   Flood filling is a technique used to fill in any enclosed areas within the figure, making them solid. This process involves filling the edges of the image inward to remove holes or gaps inside the figure. By doing so, it ensures that the figure is continuous and free from internal voids, which is crucial for accurate segmentation [6].
5) **Morphological Closing**
   Flood filling is a technique used to fill in any enclosed areas within the figure, making them solid. This process involves filling the edges of the image inward to remove holes or gaps inside the figure. By doing so, it ensures that the figure is continuous and free from internal voids, which is crucial for accurate segmentation [7].
6) **Largest Polygon Selection**
   After applying the above steps, the algorithm calculates the size of all detected shapes within the image. It then selects only the largest shape, which is most likely to be the figure of interest. This step helps in filtering out noise or smaller irrelevant shapes that may have been detected during the segmentation process. By focusing on the largest polygon, the algorithm ensures that it captures the primary object within the image [8].

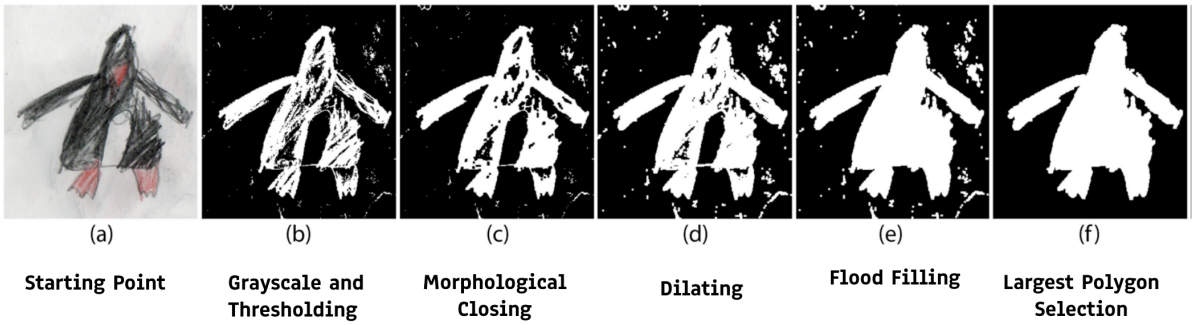

Fig. 3. Illustration of the segmentation process, showing the generated segmentation mask for a hand-drawn figure. This mask is extracted using an image processing-based approach after detecting the figure's bounding box [1].

C. Pose estimation

In order to animate the drawing figure with complex motions, we need to understand its proportions and pose [1]. Determining the exact nature of each stroke in a drawing can be challenging, even for humans. To simplify this task, Smith et al. focus on a limited set of keypoints that can serve as joints during the animation process. They adopt the 17 keypoints defined by the MS-COCO dataset and utilize a pose estimation model to predict their locations (Fig. 4) [1].

Fig. 4. Keypoints detected for pose estimation in the drawing, based on the 17 keypoints defined by the MS-COCO dataset.

D. Animation

To create a rigged character suitable for animation, researchers generated a 2D mesh from the segmentation mask using Delaunay triangulation and textured it with the original drawing, while constructing a skeleton from predicted joint locations, including the shoulders, elbows, wrists, hips, knees, ankles, and nose. They defined the root joint as the average position of the hips and the chest joint as the average position of the shoulders. The joints were connected to form a skeletal rig, and each mesh triangle was assigned to a body part (e.g., arms, legs, or trunk) based on the closest bone, allowing body parts to be rendered in different orders to simulate depth. Animation was driven by preselected motion clips, which the researchers re targeted from 3D motion data by projecting joint positions onto a 2D plane determined through principal component analysis to ensure the plane preserved maximum motion variance. Bones were oriented to match their global directions in the 2D plane, and the character mesh was reposed using as-rigid-as-possible (ARAP) shape manipulation, which preserved local geometry while allowing flexible repositioning. To maintain the stylistic coherence of children's drawings, they avoided foreshortening and applied root motion by scaling the actor's motion offset using the leg-length ratio between the actor and character. For complex motions, they computed separate 2D projection planes for upper and lower limbs, which enhanced recognizability even when motions did not align with a single plane. This technique embraced the artistic "twisted perspectives" common in children's drawings, aligning the animation style with the source art, and increased appeal, as demonstrated in user studies [1].

V. Concept to Code

A. Overview

The project leverages the **Amateur Drawings Dataset**, introduced by the Facebook Research team. The dataset includes over 178,000 annotated drawings of human figures, enabling the development of algorithms to animate hand-drawn characters. Key dataset features include:

- **Annotations**: Bounding boxes, segmentation masks, and joint location data for characters in the drawings.
- **Format**: JSON files for annotations and image files for the drawings.

B. Pipeline from Concept to Code

The following steps outline the process of using the dataset to achieve project objectives:

1) Step 1: Environment Setup: First, set up the development environment using the following commands:

```
conda create --name animated_drawings
python=3.8.13
conda activate animated_drawings
```

2) Step 2: Clone and Install Repository: Clone the `AnimatedDrawings` repository and install it using `pip`:

```
git clone https://github.com/
facebookresearch/AnimatedDrawings.git
cd AnimatedDrawings
pip install -e .
```

3) Step 3: Running the Code: Depending on your use case, you can either test the provided examples or animate your own pictures:

a) Option 1: Using Animated Drawings Examples: Run the following Python script to test with the pre-configured examples:

```
from animated_drawings import render
# Load and animate a pre-configured example
render.start('./examples/config/mvc/
interactive_window_example.yaml')
```

This will open an interactive window showcasing the animation. Use the spacebar to play/pause and arrow keys to navigate the animation.

b) Option 2: Using Your Own Pictures: To animate your own drawings, follow these steps:

1. Prepare Your Picture Ensure the picture contains a humanoid figure.

2. Set Up Docker and TorchServe Use Docker to run TorchServe for image processing. Perform the following:

- Install Docker Desktop and ensure it is running.
- Build the Docker image:
  ```
  cd torchserve
  docker build -t docker_torchserve .
  ```
- Start the Docker container and expose necessary ports:
  ```
  docker run -d --name docker_torchserve -p
  8080:8080 -p 8081:8081 docker_torchserve
  ```
- Verify TorchServe is running:
  ```
  curl http://localhost:8080/ping
  ```

  The response should be:
  ```
  {
      "status": "Healthy"
  }
  ```

  If there are issues:
  - Wait 10 seconds if TorchServe has not initialized.
  - Increase Docker memory to 16GB if RAM is insufficient.

With Docker and TorchServe set up, the process automatically detects and segments the character in your drawing, rigs it with BVH motion data, and outputs an animated file.

3. Generate Animations Run the following command to generate annotations and animations:

```
python image_to_animation.py drawings/
your_image.png output_folder
```

Replace `your_image.png` with your image path and `output_folder` with the desired output directory.

4) Step 4: View Result: The generated animation will be saved in the specified output directory as either a `.gif` or `.mp4` file.

To locate and view the results:

```
# Example of locating the output file
ls output_folder/

# Example of viewing the animation
open output_folder/video.mp4    # On macOS
xdg-open output_folder/video.mp4  # On Linux
start output_folder\video.mp4  # On Windows
```

Notes:

- The `.gif` format is ideal for embedding in presentations or sharing online.
- The `.mp4` format provides higher quality for video playback.

VI. Testing and Evaluation

To evaluate the model's performance in segmenting images, we tested it on a set of 5 images with generated segmentation masks. The evaluation metrics used were Intersection over Union (IoU) and Dice Coefficient, which are commonly used to measure the accuracy of segmentation models.

A. Metrics Definition

Intersection over Union (IoU): This metric measures the overlap between the generated segmentation mask and the corresponding ground truth mask. It is calculated as:

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

Dice Coefficient: The Dice Coefficient evaluates the similarity between the generated and ground truth masks. It is defined as:

$$\text{Dice} = \frac{2 \times \text{Area of Overlap}}{\text{Total Pixels in Ground Truth} + \text{Total Pixels in Prediction}}$$

The Python implementation used for calculating these metrics is available at: `https://github.com/anhthach375/AnimatedDrawings_341CV`

B. Results

The model was evaluated on 5 test images. Table I summarizes the IoU and Dice Coefficient for each image and provides the average values.

TABLE I
Evaluation Results: IoU and Dice Coefficient for Tested Images

Image	IoU	Dice Coefficient
Image 1	0.1295	0.2292
Image 2	0.1623	0.2792
Image 3	0.3086	0.4717
Image 4	0.3654	0.5352
Image 5	0.3577	0.5269
Average	**0.2647**	**0.4084**

C. Analysis

The evaluation results indicate variability in the model's performance across the test images. The IoU and Dice Coefficient values range from 0.1295 to 0.3654 and 0.2292 to 0.5352, respectively. Images 1 and 2 show relatively low values for both metrics, suggesting that the model struggled with segmentation in these cases, possibly due to challenges like unclear object boundaries or significant occlusions.

On the other hand, higher IoU and Dice values for Images 4 and 5 demonstrate the model's capability to segment images with well-defined features. The average IoU (0.2647) and Dice Coefficient (0.4084) reflect moderate overall performance but highlight room for improvement, particularly

in capturing finer details and ensuring better overlap between predictions and ground truth masks.

D. Visualization

Figure 5 provides a visual comparison of the generated mask and the original image in grayscale for Image 5. This example illustrates the model's ability to capture the general shape of the object while missing some finer details.

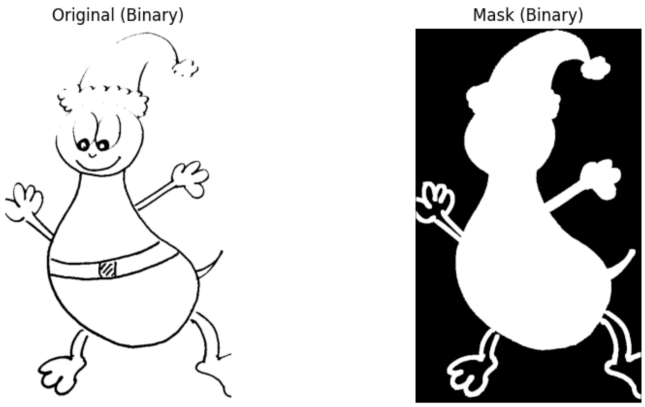

Fig. 5. Generated Mask vs. Original Image for Image 3

VII. Challenges

Smith et al. tackled several challenges in their work, including adapting standard computer vision models to better handle children's drawings. Their method adapts existing computer vision models originally designed to detect human figures and joints in photographs by fine-tuning them to work specifically with children's drawings. This adaptation accounts for the significant differences between photographic images and the abstract, non-photorealistic features of children's drawings. Furthermore, while traditional methods for detection, segmentation, and pose estimation on non-photorealistic images exist, their adoption has been slower due to the lack of easily available datasets tailored for such tasks. They also conducted experiments to explore the relationship between the size of the fine-tuning training set and the success rates of the models. This investigation was necessary to optimize the system's performance when applied to the unique characteristics of children's drawings. The Amateur Drawings Dataset, consisting of in-the-wild photographs of drawings submitted by the public, complements existing datasets and allows for further exploration and analysis in this domain.

Meanwhile, we encountered practical challenges, including a Docker memory limitation that prevented image hosting, which they resolved by increasing the memory limit to 16GB. They are also working on configuring YAML files to enable the setup of multiple pictures in a single scene, though this remains a work in progress [1].

VIII. Conclusion

This project demonstrates the potential of computer vision techniques in bringing children's drawings to life through interactive animations. By leveraging state-of-the-art models like Mask R-CNN for figure detection, segmentation methods for mask generation, and pose estimation models based on MS-COCO keypoints, we were able to develop a pipeline that successfully animates abstract and non-standard figures. The inclusion of artistic features such as "twisted perspectives" further enriches the animations, making them faithful to the creative styles of children's art.

Through this project, we learned the importance of adapting existing models to domain-specific challenges. The unique nature of children's drawings required modifications such as fine-tuning pretrained models and handling abstract features that are uncommon in real-world datasets. We also encountered practical obstacles, such as computational resource limitations and configuration issues, which emphasized the need for careful infrastructure planning and iterative debugging.

A. Future Work

Several areas of improvement and exploration remain:

- **Model Enhancements:** Future work could explore more advanced architectures, such as transformer-based models, to improve segmentation accuracy and pose estimation performance.
- **Dataset Expansion:** Expanding the dataset to include more diverse drawings could improve the model's generalizability to a wider variety of artistic styles.
- **Real-time Interaction:** Developing a real-time system that allows users to see their drawings animated instantly would increase the accessibility and practicality of this application.
- **3D Animation:** Extending the framework to generate 3D animations could open up new creative possibilities for users.

B. Real-world Implications

The ability to animate children's drawings has broad implications:

- **Educational Applications:** The system can be used as a teaching tool to engage children in learning through creativity and technology.
- **Entertainment:** Animated drawings could find applications in games, interactive storytelling, and personalized content creation.
- **Therapeutic Use:** This tool could also serve as a medium for children to express themselves in therapeutic settings, providing insights into their thoughts and emotions.

In conclusion, this project bridges the gap between abstract artistic expressions and modern animation technologies, offering a glimpse into the possibilities of integrating computer vision with creativity. While challenges remain, the results provide a strong foundation for future exploration and innovation in this space.

REFERENCES

[1] Y. L. S. J. Harrison Jesse Smith, Qingyuan Zheng and J. K. Hodgins, "A method for animating children's drawings of the human figure," *ACM Transactions on Graphics*, vol. 42, no. 32, pp. 1–15, 2023.

[2] L. L. Hongbo Fu, Shizhe Zhou and N. J. Mitra, "Animated construction of line drawings," *ACM Transactions on Graphics*, vol. 30, no. 6, pp. 1–10, 2011.

[3] S. Agarwal, "Bounding box annotation services: Best practices & tips," https://annotationbox.com/bounding-box-annotation-services-best-practices-and-tips/, June 2024, accessed: 2024-12-4.

[4] "TorchVision object detection finetuning tutorial — PyTorch tutorials 2.5.0+cu124 documentation," https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html, accessed: 2024-12-4.

[5] "Digital Image Processing — Pearson eLibrary — elibrary.pearson.de," https://elibrary.pearson.de/book/99.150005/9781292223070, [Accessed 05-12-2024].

[6] R. M. Haralick, S. R. Sternberg, and X. Zhuang, "Image analysis using mathematical morphology," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-9, no. 4, pp. 532–550, 1987.

[7] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, 1986.

[8] S. Nayak, S. Sarkar, and B. Loeding, "Automated extraction of signs from continuous sign language sentences using iterated conditional modes," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 2583–2590.