DOI: 10.1111/cgf.15176

ACM SIGGRAPH / Eurographics Symposium on Computer Animation 2024
M. Skouras and H. Wang
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 43 (2024), Number 8

SketchAnim: Real-time sketch animation transfer from videos

Gaurav Rai[†], Shreyas Gupta[†] and Ojaswa Sharma[†]

†Graphics Research Group, Indraprastha Institute of Information Technology Delhi

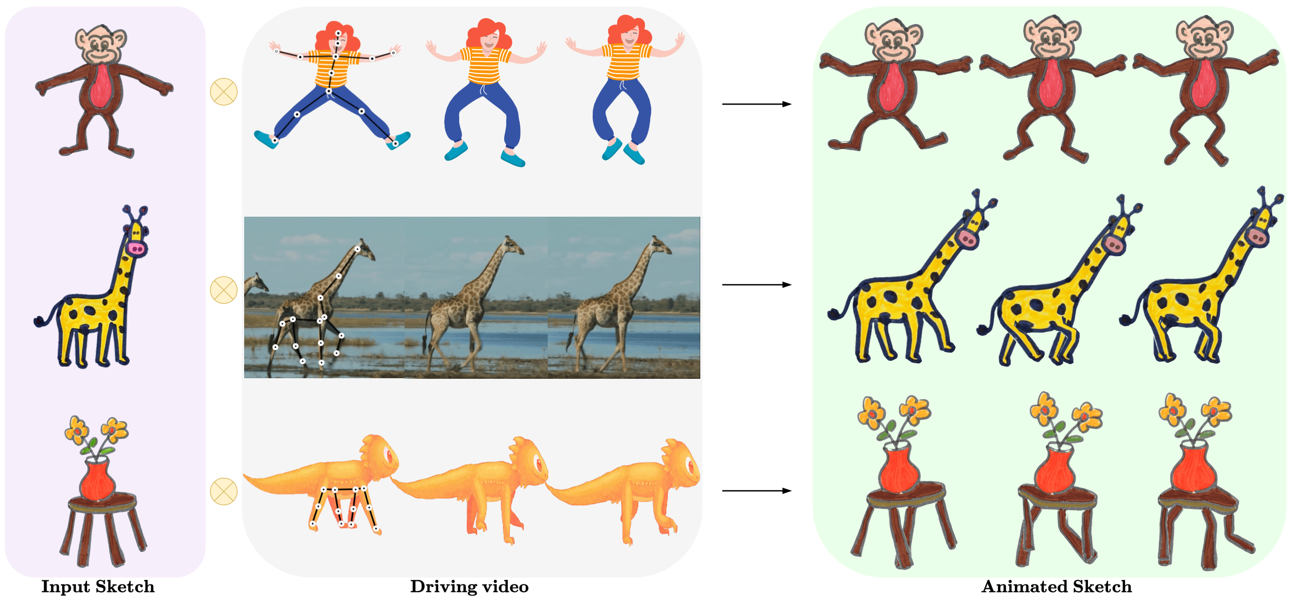

Figure 1: *Sketch animation using motion from exemplar video.*

Abstract

Animation of hand-drawn sketches is an adorable art. It allows the animator to generate animations with expressive freedom and requires significant expertise. In this work, we introduce a novel sketch animation framework designed to address inherent challenges, such as motion extraction, motion transfer, and occlusion. The framework takes an exemplar video input featuring a moving object and utilizes a robust motion transfer technique to animate the input sketch. We show comparative evaluations that demonstrate the superior performance of our method over existing sketch animation techniques. Notably, our approach exhibits a higher level of user accessibility in contrast to conventional sketch-based animation systems, positioning it as a promising contributor to the field of sketch animation. https://graphics-research-group.github.io/SketchAnim/

Keywords: Sketch animation, Skeleton mapping, Pose matching, Motion transfer

CCS Concepts
• **Computing methodologies** → *Animation;*

1. Introduction

Sketches serve as a great medium for visual communication and animating sketches can add dynamism and interactivity. Traditional tools for animating sketches require a unique set of skills to bring hand-drawn characters to life. Artists often refer to some existing videos and try to replicate a similar video motion to the sketch character when attempting to animate it. However, this process is often time-consuming, and an inaccurate replication of motion may make the character's movement look non-fluent or unnatural, especially for those with little to no experience in animation.

Character animation has been an area of extensive research in recent years. Initial work includes that of Xing et al. [XWSY15] who proposed a method that autocompletes a set of continuously repeated strokes during sketching and performs a keyframe-based animation. TraceMove [PGC16] provides a data-assistant user interface that enables novice animators in sketching and 2D character animation. It takes skeleton points as user input and utilizes a motion capture dataset to predict the skeleton in the subsequent frames. Live-Sketch [SBF*18] extracts dynamic deformation from videos as moving control points and transfers the motion to sketches while addressing certain challenges such as extracting motion from videos, video-to-sketch alignment, and motion transfer.

Recent advancements in sketch animation utilize neural network-based techniques like CharacterGAN [HFW*22], that relies on keypoints added manually by the user. Another approach, AnimationDrawing [SZL*23], uses video motion and applies it to the sketch. However, it has limitations, supporting only the animation of biped characters and generating artifacts in instances of inaccurate joint estimation and occlusion.

In this work, we propose a method for animating sketches by applying the motion from a driving video to the sketch. Our algorithm takes as input a sketch to be animated, a driving video from which the motion is to be applied, and a skeleton of the character from the sketch. It then generates an animated sketch with the motion of the video applied to it. Our method offers multiple advantages at various stages. It supports animation of sketch characters with general structures, and not just restricted to biped or quadruped characters. For instance, our method seamlessly handles inanimate object sketches. Our approach generates high quality animations compared to other recent techniques. The main contributions of our work are as follows

- We provide SketchAnim, an automatic, user-friendly, and efficient sketch animation framework capable of handling different types of motions such as biped, quadruped and inanimate. Our framework offers a two-stage approach to animating sketches from videos: *motion extraction*, and *motion transfer*.
- A novel motion transfer approach to map video skeleton motion to sketch skeleton. We introduce a depth-order based approach to handle self-occlusion during sketch deformation.
- Our proposed method shows superior performance quantitatively and qualitatively compared to state-of-the-art methods.

2. Related work

2.1. Sketch-based animation

Significant work has been done in sketch animation in recent years. The major difficulties in handling geometric deformations, partial occlusions, in-plane and out-of-plane rotations, and complex backgrounds make this task highly challenging—recent improvements with the sketch animation attempt to create interactive animations by adding dynamic motion. The approach of Bregler et al. [BLCD02] is a keyframe-based technique that tracks the cartoon motion and maps to the sketch or line drawing. Sketch-based animation methods [PGC16, XWSY15] not only animate the static object but also assist in sketching. Xing et al. [XWSY15] proposed a method that autocompletes the continuously repeated strokes during the sketching and performs keyframe-based animation. Once the user draws the sketch outlines for each frame and fills in the details in the first frame, the detail in the first frame will reflect in all the frames. The idea extends the local similarity methods and proposes the global similarity that contains high-level structure across multiple frames. Global similarity captures the object contour as global context and individual strokes as local context. However, they require manual user input at multiple stages and are purely data-driven.

Few sketch animation methods [KCGF14, KCG*14] highly depend on user interaction since these tools' performances are good, but they need lots of effort for novice animators. Xing & Kazi et al. [XKG*16] add predefined motion effects such as fire, smoke, and water to add the animation effects, but these are only for a limited animation style. Few methods [AHSS04, WXSC04] predict the style of the next keyframe, but they require manual inputs. Several methods [BJS*08, HGCA12, SBF*18, PGC16] for sketch animation that uses motion from a reference video. Such methods require manual inputs from the user at multiple stages. Tracemove [PGC16] is a purely data-driven method; the reference video is used to assist in sketching and a motion capture dataset for skeleton matching in order to make the subsequent keyframes. Ben-Zvi et al. [BZBM*16] is a data-driven method that changes the style instead of extracting the video's motion trajectories and transferring them to the input sketch. In other methods, Okabe et al. [OAIS09] animate the pictures of only fluids using the exemplar videos. The drawback of this method is that it is limited to fluid animation and requires more computation time. Santosa et al. [SCBS13] used optical flow for the animation, but it requires a similar structure of input sketch as in the exemplar video. Bregler et al. [BLCD02] is a keyframe-based technique that tracks the cartoon motion and maps to the sketch or line drawing. Since it needed manual user input at many stages, other sketch animation methods [CHZ14, TZS*16] use facial landmarks but are restricted to facial animation. On the other hand, [DAC*06, WRKS16] use MoCap [KW20] for the animation, but it is restricted to the human body rather than the arbitrary object.

Recent text-to-sketch animation methods, such as Gal et al. [GVA*23] take vector sketch input (represented as a cubic Bézier curve), use a text-to-video diffusion model, and optimize a Score Distillation Sampling (SDS) loss to generate the animated sketch. Similarly, AniClipart [WSML24] uses a Bézier curve motion trajectory of sketch keypoint and utilizes Video Score Distillation Sampling (VSDS) loss with a text-to-video prior for generating the sketch animation. Although the methods provide promising performance, they incorporate text input which is a different modality than a video input.

Live-Sketch [SBF*18] extracts the sparse motion trajectories from the video and transfers them to the sketch using control point tracking. It uses mesh deformation during the motion transfer in order to prevent distortion. However, this method also needs manual user input and cannot handle the motion for smooth regions. The hand-drawn character animation method [SZL*23] performs hand-drawn character animation using video motion but suffers from motion estimation in smooth regions and only handles biped motion.

Our proposed method, SketchAnim, is semi-automatic and needs only sketch input and a reference video with the skeleton of the video rest pose. Our method does need to give additional manual input, and it handles the topology structure even though the same category object structure is different in the video from the sketch.

2.2. Motion transfer

The two primary categories of deep learning techniques for motion transfer are model-based and model-free approaches. Model-based approaches primarily focus on pose-guided human image generation [CGZE19, GCT19]. The other model-based methods [HKK*20, WKZ18] use facial landmark detector to extract the pose of the driving image as guidance information. These methods require prior supervision of the moving object, such as facial landmarks and human pose. On the other hand, model-free approaches do not require any supervision or labeled data. Since the video generation problem is close to the future frame prediction problem, X2Face [WKZ18] propose a deep learning method that uses a dense motion field to deform the input face and generate the deformed image output. However, this is a data-driven method and is limited to faces only. In the past years, many deep learning methods [SLT*19a, SLT*19b] have been proposed that are unrestricted to a specific object and do not require labeled data. MonkeyNet [SLT*19a] is a self-supervised method that uses a learned unsupervised keypoint-based dense motion field using which it transfers the motion pattern of the driving video to the image. This method fails when the driving video poses drastic pose changes. FOMM [SLT*19b] adds local affine transformation over the learned keypoints to generate the animated image which improves the motion transfer quality. However, it fails for the sketch modality since it is a keypoint transformation-based motion model only suitable for image animation. Our proposed method, SketchAnim, is user-friendly and needs only sketch input and a reference video with a skeleton on the first frame of the video. It does not require additional manual user inputs, and handles the topology structure even though the object structure is different in the video from the sketch.

2.3. Video tracking

Object tracking is one of the essential problems in computer vision and computer graphics; object tracking has a variety of applications, including remote surveillance, human-computer interaction, and autonomous driving. The difficulties in handling geometric deformations, partial occlusions, and complex backgrounds make this task difficult even after much work has been put into it. Traditional energy minimization-based optical flow methods [HS81, S*94, BBM09, MB08, CWL*14, HGS*15, BF06, ČKL12, AIK11] estimate the pixel-level flow. These methods are limited to tracking each feature point individually; these methods do not consider the topological relationship of the object. It doesn't track the object's semantic-meaning part, which results in a large distortion and artifacts even with a small tracking failure. Buchanan et al. [BF06] proposed an interactive feature tracking method that tracks accurate and meaningful features from the 2D video. It uses a dynamic tracking approach and a k-d tree for efficient computation. DP-Track does not find the global minimum of its energy function because the occlusion is not correctly formulated. GraphTrack [AV11] proposed an enhancement of the DP-Track [BF06] method and is more reliable on complex videos and more stable when users add additional patches. Because GraphTrack provides a full-frame appearance rather than only a patch appearance. SIFT Flow [LYT10] has been proposed to find the pixel-to-pixel correspondence between two frames inspired by Optical flow. SIFT flow matches SIFT descriptor rather than raw pixels. The major drawback is when there are significant changes in appearance. Cai et al. [CWL*14] proposed a tracking approach based on graph learning to address deformation and occlusion challenges. The graph learning architecture incorporates a variety of inputs, including appearance and geometric location, to enhance tracking performance. In recent years, there are several object-tracking methods [IMS*17, HSZ*22, SYLK18, KRG*23, DFI*15] have been proposed. While Deep learning methods are effective for short-term tracking, pairwise optical flow techniques cannot handle extended temporal contexts, making them unsuitable for long-term tracking. Long-term optical flow methods [SHL*23, JGR*18] address this limitation by integrating multiple frames to extend pairwise flow. However, they struggle if occlusion occurs in the video's multiple frames. Point tracking methods [ST08, NŠM24, DGM*22, DYV*23, WCC*23, HFF22] overcome the limitation by estimating long-range particle motion throughout the video and handling occlusion at multiple-frames. PIPs [HFF22] use fixed temporal windows to predict a point's motion. It disregards the context information when occlusion exceeds the temporal window since it tracks the point individually. At the same time, TAPIR [DGM*22] contained the temporal window with temporal depth order. Co-tracker [WCC*23] achieves state-of-the-art performance by harnessing spatial correlation among multiple points and effectively handling the occlusion across multiple frames.

3. Methodology

We propose *SketchAnim*, an easy to use and robust framework for sketch animation using motion from a driving video. The algorithm begins by obtaining user-provided input, including a sketch $\mathcal{S}$, a driving video $\mathcal{V}$ containing a moving object, and a user-provided skeleton $\mathcal{K}_v^1$ of the object in the first frame of the driving video. The driving video may be another animated or a real-world video that provides a realistic character motion for the sketch. The video skeleton $\mathcal{K}_v^1$ is tracked across all frames of the video. Subsequently, bone-joint angles are estimated for the motion skeleton. The video skeleton is mapped to the sketch to compute a sketch skeleton $\mathcal{K}_s$ and skinning weights are computed using bounded biharmonic weights [JBPS11]. Our algorithm then computes the bone transformations required for sketch animation through Linear Blend Skinning (LBS). Finally, the transformations are applied to the sketch to generate an animated output video $\mathcal{V}_s$, effectively portraying the given sketch in accordance with the dynamics of the driving video sequence. The output video is presented to the user as the animated result, encapsulating the essence of the original sketch within the context of the driving video. Algorithm 1 outlines the primary steps of our approach and Figure 2 shows the overall sketch animation framework.

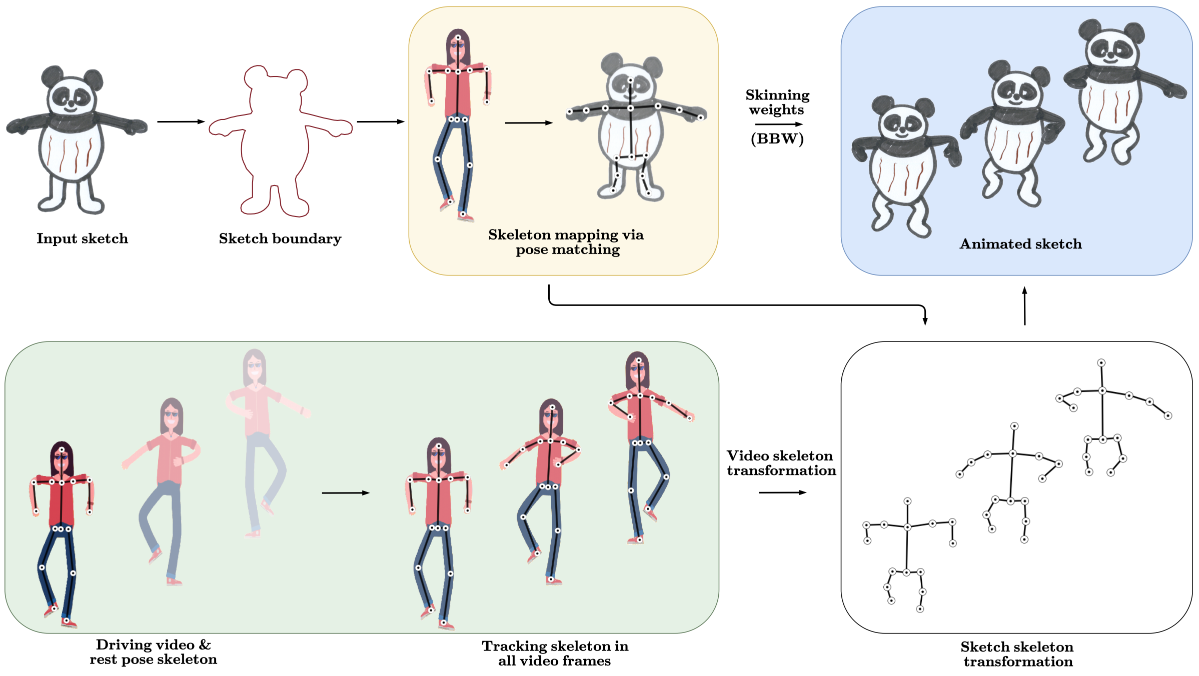
Input sketch
Sketch boundary
Skeleton mapping via pose matching
Skinning weights
(BBW)
Animated sketch
Driving video & rest pose skeleton
Tracking skeleton in all video frames
Video skeleton transformation
Sketch skeleton transformation

Figure 2: *Overview of our SketchAnim framework for sketch-based animation. The method involves taking a sketch and a video as input. The user then draws a skeleton on the initial frame of the video. Mapping of this skeleton to the input sketch is done by shape matching with MVC. Thereafter, the transformation of skeleton bones is estimated by tracking the skeleton joints across the video frames. Finally, the sketch is animated using linear blend skinning with BBW weights.*

ALGORITHM 1: SketchAnim: Video-to-Sketch motion transfer

```
Input: Sketch $\mathcal{S}$, driving video $\mathcal{V}$, and video skeleton $\mathcal{K}_v^1$
Output: Sketch video $\mathcal{V}_s$
Segment sketch boundary polygon $\mathcal{B}_s$
Compute triangulation $\mathcal{T}_s$ (CDT) of $\mathcal{B}_s$
Segment the first frame of the video $\mathcal{B}_v$
Compute sketch skeleton $\mathcal{K}_s$ from $\mathcal{K}_v^1$ using MVC (Sec. 3.1)
Tracking video skeleton $\mathcal{K}_v^1$ across all frames
Motion transfer between video and sketch skeleton(Sec. 3.2)
Compute BBW weights for $\mathcal{T}_s$ and $\mathcal{K}_s$ (Sec. 3.3)
Deform $\mathcal{S}$ using LBS to produce $\mathcal{V}_s$
```

3.1. Skeleton mapping

We use the Segment Anything Model (SAM) [KMR*23] to perform segmentation of the characters from the sketch and the first video frame. The boundaries of these segmentation regions are resampled with points spaced d distance apart as polygons $\mathcal{B}_s$ and $\mathcal{B}_v$ respectively for the sketch image and the video frame. At this stage, the character poses represented by $\mathcal{B}_s$ and $\mathcal{B}_v$ may not exactly match.

The skeleton $\mathcal{K}_v^1$ of the first frame of the video is taken as an input from the user. We automatically map the vertices of this skeleton to create a sketch skeleton $\mathcal{K}_s$ with the topology of the former.

Our skeleton mapping approach is based on shape matching between boundaries $\mathcal{B}_v$ and $\mathcal{B}_s$. To start with, $\mathcal{B}_s$ and $\mathcal{B}_v$ do not have much correlation as they were constructed independently of each other. We use the neural deformation pyramid approach of Li and Harada [LH22] to deform the video boundary $\mathcal{B}_v$ and get a close approximation of $\mathcal{B}_s$. This results in a new sketch boundary $\tilde{\mathcal{B}}_s$ whose points are in correspondence with $\mathcal{B}_v$ and $|\tilde{\mathcal{B}}_s| = |\mathcal{B}_v|$. Once the shape correspondence between the two boundaries is established, we transform the vertices of the video skeleton $\mathcal{K}_v^1$ to those of $\mathcal{K}_s$ via barycentric coordinates. To perform the same, we use mean value coordinates (MVC) [Flo03] to calculate the barycentric coordinate $\alpha_v \in \mathbb{R}^{|\mathcal{B}_v|}$ of a vertex $b_v \in \mathcal{K}_v^1$ with respect to the boundary polygon $\mathcal{B}_v$. The corresponding skeleton point p_s (see Figure 3) in the sketch skeleton $\mathcal{K}_s$ w.r.t. $\tilde{\mathcal{B}}_s$ can be calculated as

$$p_s = \sum_{j=1}^{|\tilde{\mathcal{B}}_s|} b_{s,j}\,\alpha_{v,j}, \forall b_{s,j} \in \tilde{\mathcal{B}}_s. \tag{1}$$

In order to transfer the video motion to the sketch, we also need to capture the motion of $\mathcal{K}_v^1$ across frames of the video. This can be achieved with point tracking in the video, however the primary challenge here comes from occlusions and error propagation across frames. We require a sparse tracking method that is reasonably resilient to occlusions and propagated errors. We use the Co-Tracker [KRG*23] neural network model to track vertices of $\mathcal{K}_v^1$ across frames keeping the skeleton topology fixed. This results in a sequence of video skeletons $\{\mathcal{K}_v^i\}$ that we use for motion mapping to the sketch.

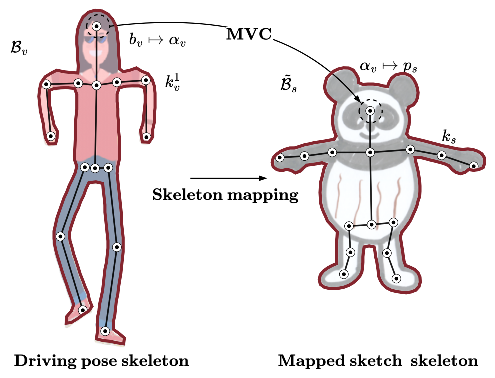

Figure 3: *Video skeleton mapping to sketch via mean-value coordinates.*

3.2. Motion extraction and motion transfer

We would like to derive a sequence of sketch skeletons $\{\mathcal{K}_s^i \mid i = 1,\ldots,n\}$ from the sequence of video skeletons $\{\mathcal{K}_v^i\}$ computed previously. While there are many ways to do so, for best results one should ensure that the skeleton motion of the video is captured by the sketch accurately and that the sketch proportions do not get deformed unnaturally.

3.2.1. Skeleton tree node hierarchy

Consider the video skeleton $\mathcal{K}_v^1$ as a tree, with the *deepest vertex* designated as the root. The deepest vertex is determined by considering distances from leaf nodes in the skeleton graph with edges weights taken as the edge lengths. We determine the root node as the non-leaf node that minimizes the sum of all shortest-path lengths to all the reachable leaves (as shown in Figure 4). In situations where there is a tie, the vertex with the highest degree is selected to be the root node. We maintain the same tree topology across the video frames. Since the video and sketch skeletons are isomorphic, we choose the corresponding vertex in the sketch skeleton to be the root node.

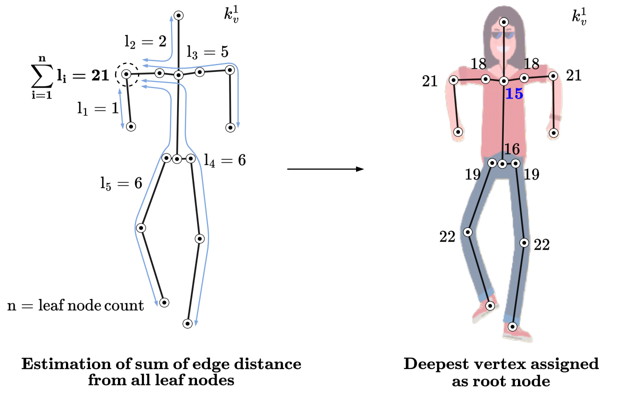
k_v^1
$l_2 = 2$
$l_3 = 5$
$\sum_{i=1}^{n} l_i = 21$
$l_1 = 1$
$l_4 = 6$
$l_5 = 6$
n = leaf node count
k_v^1
21
18
18
21
15
16
19
19
22
22
Estimation of sum of edge distance from all leaf nodes
Deepest vertex assigned as root node

Figure 4: *Estimation of the root node as deepest vertex. The blue color number denotes the smallest sum from the leaf nodes and the corresponding node as the root.*

3.2.2. Initial pose matching

The skeleton of the sketch $\mathcal{K}_s$ thus obtained may not have the same pose as that of the first frame of the video skeleton $\mathcal{K}_v^1$, therefore we first match the pose of the former to the later. We achieve this by aligning the edges of $\mathcal{K}_s$ to those of $\mathcal{K}_v^1$, without changing the position of the root node.

Our pose matching is performed in a breadth-first search (BFS) manner starting from the root node. For an edge connected to a node under consideration, we reorient it in the direction of the corresponding edge in $\mathcal{K}_v^1$. More specifically, for a skeleton node p_v under consideration in $\mathcal{K}_v^1$, let a connected edge be (p_v, q_v). The corresponding node q_s connected to p_s in $\mathcal{K}_s$ is modified as

$$q_s' = p_s + \frac{\|q_s - p_s\|}{\|q_v - p_v\|}(q_v - p_v). \tag{2}$$

We get the pose-matched sketch skeleton $\mathcal{K}_s^1$ after performing the entire BFS pass of this modification (see Figure 5).

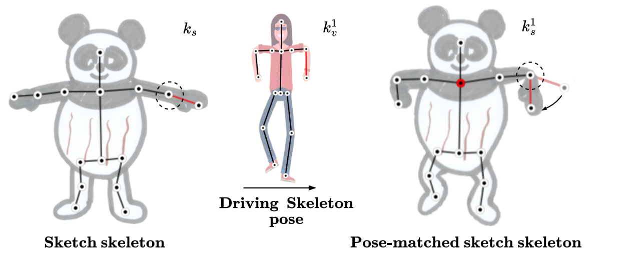

Figure 5: *Pose matching of the sketch skeleton to the video skeleton. The skeleton is traversed in a BFS manner starting from the root node (shown in red). At a current node, transformations are applied to edges in its 1-ring to match the orientation of the corresponding source edges.*

3.2.3. Motion mapping

We map motion of the sequence of video skeletons $\{\mathcal{K}_v^i\}$ to the sketch skeleton $\mathcal{K}_s^1$ sequentially. Our approach is to compute transformations between skeletons of two successive video frames and apply that to the starting skeleton in the sketch skeleton sequence, starting from the first frame. Here again we proceed in a breadth first fashion to calculate the transformation parameters for each edge and transfer the same to the sketch skeleton under consideration.

During motion mapping at timestep t the root node r_v^t undergoes only translation, which we apply to the corresponding sketch skeleton as $r_s^{t+1} = r_s^t + (r_v^{t+1} - r_v^t)$. For a skeleton node p_v^t under consideration, let a connected edge be (p_v^t, q_v^t). We calculate the rotation and scaling for the corresponding sketch edge as

$$\theta = \cos^{-1}\left(\frac{(q_v^{t+1} - p_v^{t+1}) \cdot (q_v^t - p_v^t)}{\|(q_v^{t+1} - p_v^{t+1})\|\|(q_v^t - p_v^t)\|}\right), \tag{3}$$
$$s = \frac{(q_v^{t+1} - p_v^{t+1})}{(q_v^t - p_v^t)}.$$

The entire BFS pass gives us the sketch skeleton $\mathcal{K}_s^{t+1}$ at the next timestep. We also keep these transformation parameters as bone transformations for linear blend skinning in the next stage.

3.3. Resolving self-occlusions

Self-occlusions in 2D animation are difficult to handle if the depth ordering of different parts of the object are not known. We resolve occlusions by creating depth order for the skeleton vertices first and then propagating these to the mesh during skinning. To resolve occlusions, we add a discrete depth value to each vertex in the skeleton starting with a zero value at the root vertex. Every other vertex is given a value equal to the number of edges it is away from the root. We assign a negative sign to all the vertices for which the corresponding vertices in the video skeleton were found to be occluded in one or more frames during tracking. This gives us the correct depth order of skeleton vertices in the sketch. During skinning animation, we transfer the depth values from skeleton vertices to mesh vertices based on the skinning weights in a fashion similar to linear blend skinning and use these depth values as the z-coordinate of mesh vertices in rendering.

After the video motion is transferred to sketch skeleton, we animate the sketch. To do so, we first triangulate the sketch boundary polygon $\mathcal{B}_s$ and compute a constrained Delaunay triangulation (CDT). Additional points are added to the triangulation by Poisson disk sampling the interior of the boundary. Smooth skinning weights for this triangulation and the skeleton $\mathcal{K}_s^1$ are computed using the Bounded Biharmonic Weights (BBW) [JBPS11] method. The triangulation is then deformed using linear blend skinning (LBS) and texture mapped with the sketch image to generate each frame of the animated sketch. Algorithm 2 outlines our entire approach to motion extraction and motion mapping from video to sketch.

ALGORITHM 2: Motion transfer between video and sketch skeletons

```
Input: Sketch skeleton $\mathcal{K}_s$ and sequence of video skeletons $\{\mathcal{K}_v^i \mid i = 1, \ldots, n\}$
Output: Sequence of sketch skeletons $\{\mathcal{K}_s^i \mid i = 1, \ldots, n\}$
Define the deepest vertex (skeleton joint) as root
foreach skeleton $\mathcal{K}_v^i$ in the sequence do
    foreach joint in $\mathcal{K}_v^1$ do
        Initial pose matching $\mathcal{K}_v^1$ to $\mathcal{K}_s^1$ (Eqn. 2)
        Traverse the skeleton joint in BFS manner
        Estimate the sketch skeleton transformation using $\mathcal{K}_v^i$ (Eqn. 3)
        Assign depth order to each joint (root = 0)
    end
end
Compute BBW weights for $\mathcal{T}_s$ and $\mathcal{K}_s$ and deform using LBS
```

4. Results and comparisons

We evaluated the performance of SketchAnim on a system equipped with an Intel Xeon CPU and an Nvidia Quadro P6000 GPU featuring 24 GB of memory. For the skeleton tracking, we use a co-tracker [KRG*23], and shape matching is performed using the deformation pyramid method of Li and Harada [LH22]. Our approach incorporates a novel techniques for motion extraction, motion transfer, and occlusion handling that enhance the quality of our animation. The framework is implemented in Python as an interactive application. The source code for our implementation will be made available for research purposes.

In terms of computational time, our method takes approximately 30 seconds to generate an animation sequence of 24 frames with a skeleton consisting of 12-15 vertices. This duration may vary depending on the number of skeleton vertices and the frames in the driving video. In our method, the motion transfer phase is the most time-consuming, primarily at mesh deformation and animated video rendering stages. On the other hand, in the motion extraction phase, the time taken by skeleton mapping depends upon the mesh configuration.

We have used a set of colored sketches to illustrate sketch animation capabilities of SketchAnim. These include about 50 hand-drawn sketch samples with different categories - biped, quadruped, and inanimate. We further collected samples of driving videos with the Creative Commons license available on the internet. The selected cartoon and natural videos contain a central character/object visible in all the frames. Some quadruped samples are also selected from the MGif [SLT*19b] dataset. Our sketch dataset is available publicly for research purposes and can be accessed at https://github.com/graphics-research-group/SketchAnim/.

4.1. Evaluation matrices

We use the following metrics to evaluate quantitative results and compare SketchAnim with state-of-the-art methods.

- **FID:** Frechet Inception distance (FID) [HRU*17] is used to estimate the overall quality of the generated frame. It measures the similarity between the distribution of real images and generated images.
- **AED:** Average Euclidian distance (AED) estimates the identity similarity of the ground truth and generated frame. This implies that the generated frame shares the same identity as the ground truth image. We use a person re-identification [HBL17] network similar to TPS [ZZ22] to measure the identity features of the human body.

We use the input sketch as the real sample and the animated sketch sequence as the generated sample to evaluate FID. For estimating AED, the driving video is considered the real sample and an animated sketch sequence as the generated sample.

4.2. Comparisons with the state-of-the-art

4.3. Qualitative evaluation

We perform animation on 50 sketch samples across different categories and assess the motion transfer and appearance consistency within the animated sequences for the qualitative evaluation. Figures 6 and 7 show animated sequences for three different categories of sketches: biped, quadruped, and inanimate. In the biped example (see Figure 6(a)), the hand movements of the sketch character

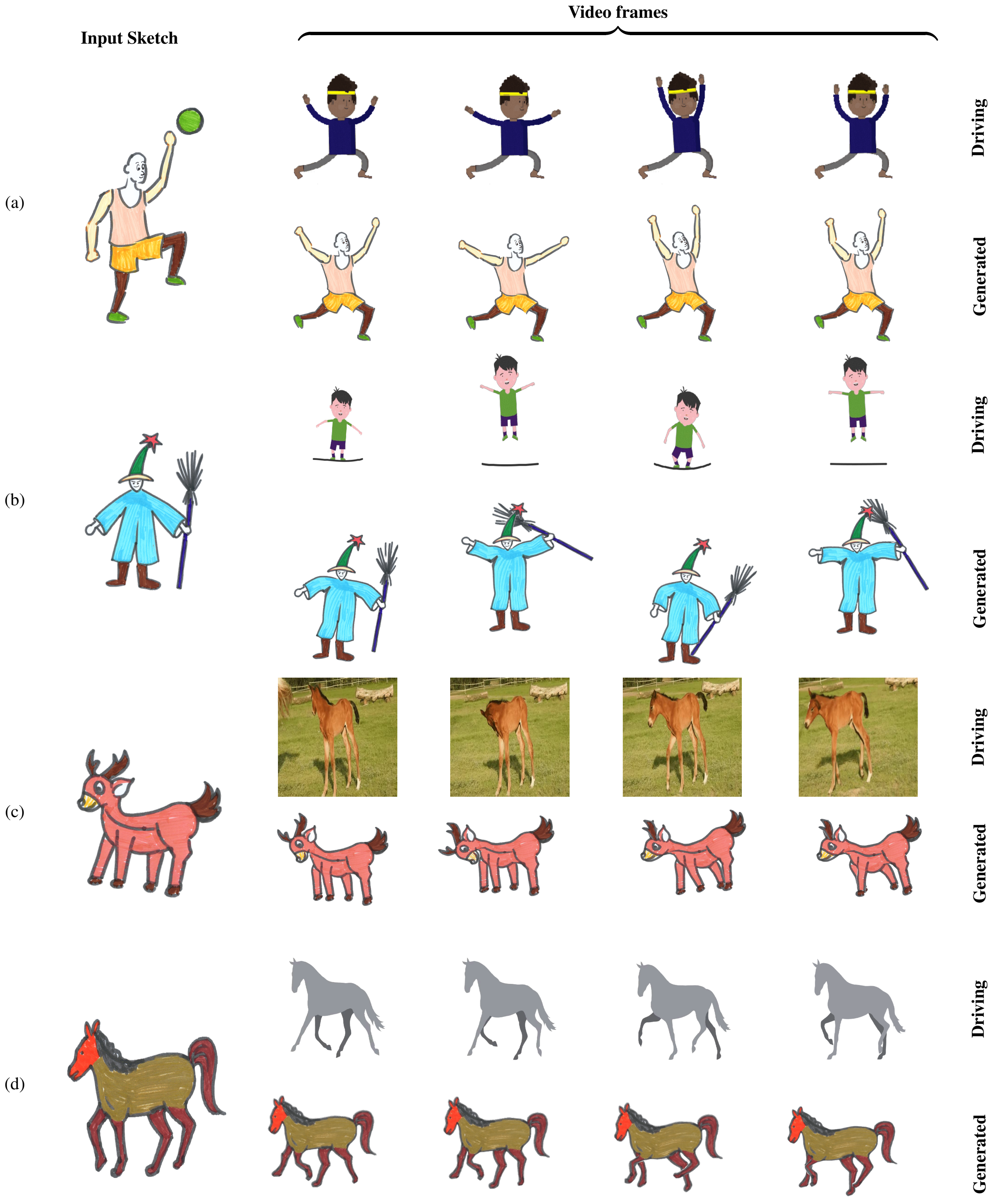

Figure 6: *SketchAnim results on biped and quadruped sketches.*

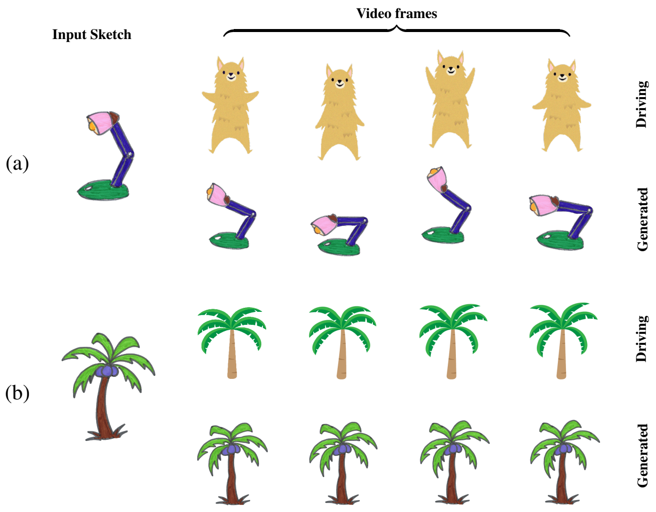

Figure 7: *SketchAnim results on sketches of inanimate objects.*

precisely correspond to the same in the driving videos. Figure 6(b) illustrates the appropriate handling of self-occlusion as the broom passes behind the character's head without any distortion or artifact. In the quadruped sample, the deer's motion aligns with the exemplar horse's movement in Figure 6(c). Additionally, illustrated in Figure 6(d), the motion of a horse's legs is depicted, showcasing its running movement. In non-living sketches, Figure 7(a), the lamp exhibits jumping and bending motions in accordance with the hand movements of a bear. In another scenario, (see Figure 7(b)), the motion of tree leaves is mirroring the similar motion style of tree in the driving video.

The method of Siarohin et al. [SLT*19b] uses keypoint transformation for animation that fails in cross-domain motion transfer and generates artifacts during sketch animation. In Figure 8, the hand motion distorted during the deformation, and the motion consistency is not stable. Similarly, the leg motion of the quadruped character distorts and collapses during animation, as depicted in Figure 9. The approach of Smith et al. [SZL*23] is close to our method and generates promising results, but it is limited to some kind of motion (biped motion), whereas our method seamlessly handles other motion categories very well. In Figure 8, the bat moves behind the character's body; it generates the artifact during deformation. On the other hand, our method maintains appearance consistency and smooth deformation. Also, Figures 8 and 9 show qualitative comparison with these methods. It is clear that the subtle motion of body parts in the characters are very well captured by our method whereas the other two fail to model these. The cases of occlusions are also well handled with our method.

4.3.1. Quantitative evaluation

We performed a quantitative evaluation on 15 bipeds, 15 quadrupeds, and 10 inanimate sketch samples. Our method demonstrates higher efficiency and accuracy compared to other state-of-the-art techniques. Table 1 shows a quantitative comparison between these method and ours. Our method achieves a better FID score (lower is better), indicating higher quality in the generated sketches. Additionally, our proposed method demonstrates superior AED metric, reflecting its robustness and capability. AnimationDrawing [SZL*23] is limited to biped motion only, therefore we cannot evaluate it for quadruped and inanimate classes. For these two motion categories we compare our approach against FOMM [SLT*19b]. Table 1 presents the comparison results, demonstrating that our method outperforms FOMM. Our approach is more robust and accurate in terms of motion transfer, reduced distortion, and precise motion estimation, especially in handling occlusion cases.

Table 1: Quantitative comparison with state-of-the-art methods.

Method	Biped		Quadruped	Inanimate
	FID↓	AED↓	FID↓	FID↓
Siarohin et al. [SLT*19b]	186.37	0.4826	160.01	–
Smith et al. [SZL*23]	201.65	0.4775	–	–
SketchAnim (Ours)	**150.25**	**0.4430**	**140.00**	**176.89**

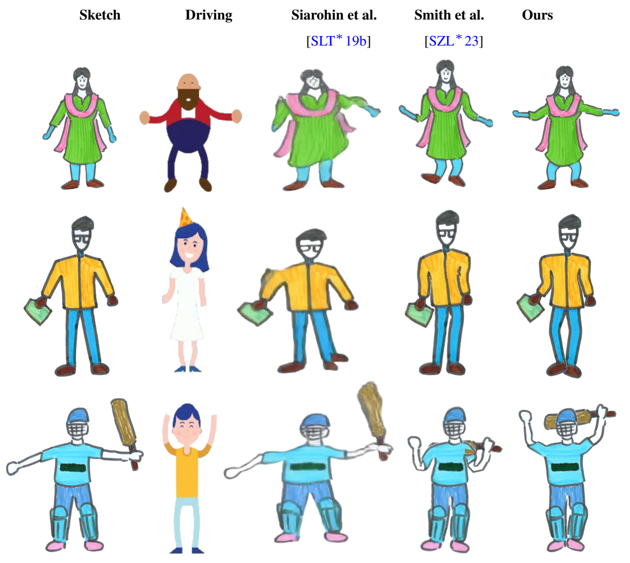

Figure 8: *Comparison of biped samples with the state-of-the-art methods.*

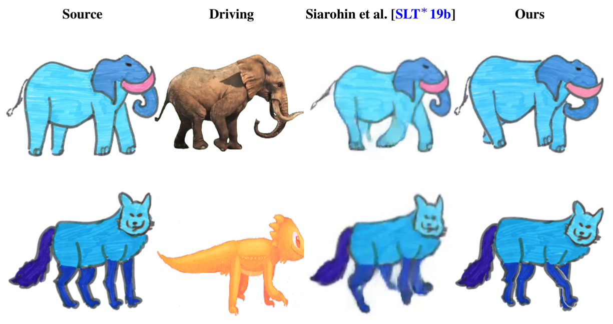

Figure 9: *Comparison of quadruped animation with the state-of-the-art methods.*

5. Limitations and future work

Our proposed framework has limitations, such as it fails to handle motion without correspondence between the video object and sketch. If the skeleton topologies differ, we cannot transfer human motion to an animal or vice-versa. Additionally, if missing details are in the input sketch, our method will not autocomplete the missing details during deformation.

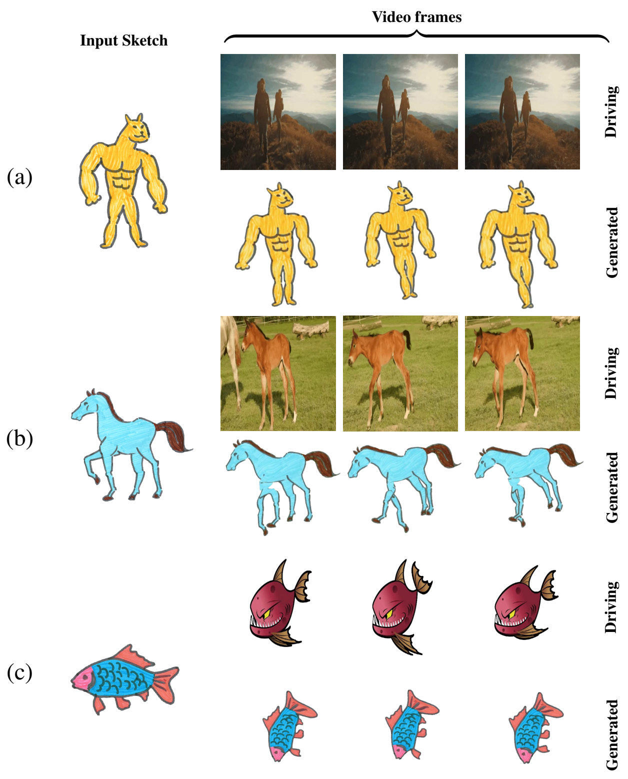

Figure 10: *Failure cases such as (a) frontal motion, (b) self-occlusion in rest pose, and (c) error propagation in shape mapping.*

Our method struggles to handle frontal motion, such as a human moving toward the camera's direction (see Figure 10(a)). In the rest pose, the body part should not be self-occluded because the tracking method cannot handle the occlusion in such cases. In some cases where the body parts may be occluded in the rest pose, the generated output tends to have distortion as shown in Figure 10(b) where the legs of the horse are occluded in the rest pose, which leads to the artifacts in the animated frames. Error propagation presents a challenge in our method; if tracking failure or shape mismatch occurs in the initial frames, errors will accumulate in the final animated sequence. Additionally, incorrect skeleton mapping might reorient the animated object incorrectly. In Figure 10(c), the tracking point of the fish's mouth shifts toward the wing, distorting the final animated sketch.

In future work, we aim to improve the motion extraction method by incorporating 3D context to address above issues. A keypoint-based tracking approach can be developed to estimate the motion details (such as co-part motion) to enhance performance. Furthermore, depth cues and camera parameters can be integrated to enable 3D handling capabilities. We would also like to work on an automatic skeleton extraction and matching approach to enhance our method for efficiency and robustness.

6. Conclusion

In this paper, we introduce a method for animating sketches based on a static sketch and a reference video. The animation process involves applying the motion from the reference video to animate the sketch. Users have the flexibility to choose a specific animation for the sketch, providing a direct reference video input. Furthermore, we take minimal user input in the form of drawing a skeleton on a video frame. This approach enables the animation of sketches with arbitrary shapes, removing restrictions to specific categories such as biped or quadruped. Our method successfully captures subtle motion from the given video and performs better than similar methods in terms of quality of animation. While our proposed framework exhibits notable strengths, it does have certain limitations. For instance, it may encounter challenges in handling motion when there is no direct correspondence between the video object and the sketch. Tracking failures can arise in complex motion scenarios within the video, and instances of occlusion. In such cases, errors may propagate in skeleton mapping and motion transfer. Although our method effectively manages self-occlusion cases, it does not autonomously complete missing sketch details in specific scenarios when there is overlap between parts of the body. We wish to handle more complex scenarios by building upon this work.

7. Acknowledgement

This work was supported by the iHub Anubhuti IIITD Foundation. We would like to thank Anshul Goswami for creating the hand-drawn sketches and allow to be used for research purposes.

References

[AHSS04] Agarwala A., Hertzmann A., Salesin D. H., Seitz S. M.: Keyframe-based tracking for rotoscoping and animation. *ACM Transactions on Graphics (ToG) 23*, 3 (2004), 584–591. 2

[AIK11] Artner N. M., Ion A., Kropatsch W. G.: Reprint of: Multi-scale 2D tracking of articulated objects using hierarchical spring systems. *Pattern recognition 44*, 9 (2011), 1969–1979. 3

[AV11] Amberg B., Vetter T.: GraphTrack: Fast and globally optimal tracking in videos. In *CVPR 2011* (2011), IEEE, pp. 1209–1216. 3

[BBM09] Brox T., Bregler C., Malik J.: Large displacement optical flow. In *2009 IEEE Conference on Computer Vision and Pattern Recognition* (2009), IEEE, pp. 41–48. 3

[BF06] Buchanan A., Fitzgibbon A.: Interactive feature tracking using kd trees and dynamic programming. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)* (2006), vol. 1, IEEE, pp. 626–633. 3

[BJS*08] Barnes C., Jacobs D. E., Sanders J., Goldman D. B., Rusinkiewicz S., Finkelstein A., Agrawala M.: Video puppetry: a performative interface for cutout animation. In *ACM SIGGRAPH Asia 2008 papers*. 2008, pp. 1–9. 2

[BLCD02] Bregler C., Loeb L., Chuang E., Deshpande H.: Turning to the masters: Motion capturing cartoons. *ACM Transactions on Graphics (TOG) 21*, 3 (2002), 399–407. 2

[BZBM*16] Ben-Zvi N., Bento J., Mahler M., Hodgins J., Shamir A.: Line-drawing video stylization. In *Computer Graphics Forum* (2016), vol. 35, Wiley Online Library, pp. 18–32. 2

[CGZE19] Chan C., Ginosar S., Zhou T., Efros A. A.: Everybody dance now. In *Proceedings of the IEEE/CVF international conference on computer vision* (2019), pp. 5933–5942. 3

[CHZ14] Cao C., Hou Q., Zhou K.: Displaced dynamic expression regression for real-time facial tracking and animation. *ACM Transactions on graphics (TOG) 33*, 4 (2014), 1–10. 2

[ČKL12] Čehovin L., Kristan M., Leonardis A.: Robust visual tracking using an adaptive coupled-layer visual model. *IEEE Transactions on Pattern Analysis and Machine Intelligence 35*, 4 (2012), 941–953. 3

[CWL*14] Cai Z., Wen L., Lei Z., Vasconcelos N., Li S. Z.: Robust deformable and occluded object tracking with dynamic graph. *IEEE Transactions on Image Processing 23*, 12 (2014), 5497–5509. 3

[DAC*06] Davis J., Agrawala M., Chuang E., Popović Z., Salesin D.: A sketching interface for articulated figure animation. In *Acm siggraph 2006 courses*. 2006, pp. 15–es. 2

[DFI*15] Dosovitskiy A., Fischer P., Ilg E., Hausser P., Hazirbas C., Golkov V., Van Der Smagt P., Cremers D., Brox T.: Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision* (2015), pp. 2758–2766. 3

[DGM*22] Doersch C., Gupta A., Markeeva L., Recasens A., Smaira L., Aytar Y., Carreira J., Zisserman A., Yang Y.: Tap-vid: A benchmark for tracking any point in a video. *Advances in Neural Information Processing Systems 35* (2022), 13610–13626. 3

[DYV*23] Doersch C., Yang Y., Vecerik M., Gokay D., Gupta A., Aytar Y., Carreira J., Zisserman A.: Tapir: Tracking any point with per-frame initialization and temporal refinement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2023), pp. 10061–10072. 3

[Flo03] Floater M. S.: Mean value coordinates. *Computer aided geometric design 20*, 1 (2003), 19–27. 4

[GCT19] Geng Z., Cao C., Tulyakov S.: 3d guided fine-grained face manipulation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2019), pp. 9821–9830. 3

[GVA*23] Gal R., Vinker Y., Alaluf Y., Bermano A. H., Cohen-Or D., Shamir A., Chechik G.: Breathing life into sketches using text-to-video priors. *arXiv preprint arXiv:2311.13608* (2023). 2

[HBL17] Hermans A., Beyer L., Leibe B.: In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737* (2017). 6

[HFF22] Harley A. W., Fang Z., Fragkiadaki K.: Particle Video Revisited: Tracking through occlusions using point trajectories. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXII* (2022), Springer, pp. 59–75. 3

[HFW*22] Hinz T., Fisher M., Wang O., Shechtman E., Wermter S.: Charactergan: Few-shot keypoint character animation and reposing. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision* (2022), pp. 1988–1997. 2

[HGCA12] Held R., Gupta A., Curless B., Agrawala M.: 3D puppetry: a kinect-based interface for 3D animation. In *UIST* (2012), vol. 12, Citeseer, pp. 423–434. 2

[HGS*15] Hare S., Golodetz S., Saffari A., Vineet V., Cheng M.-M., Hicks S. L., Torr P. H.: Struck: Structured output tracking with kernels. *IEEE transactions on pattern analysis and machine intelligence 38*, 10 (2015), 2096–2109. 3

[HKK*20] Ha S., Kersner M., Kim B., Seo S., Kim D.: Marionette: Few-shot face reenactment preserving identity of unseen targets. In *Proceedings of the AAAI conference on artificial intelligence* (2020), vol. 34, pp. 10893–10900. 3

[HRU*17] Heusel M., Ramsauer H., Unterthiner T., Nessler B., Hochreiter S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems 30* (2017). 6

[HS81] Horn B. K., Schunck B. G.: Determining optical flow. *Artificial intelligence 17*, 1-3 (1981), 185–203. 3

[HSZ*22] Huang Z., Shi X., Zhang C., Wang Q., Cheung K. C., Qin H., Dai J., Li H.: Flowformer: A transformer architecture for optical flow. In *European conference on computer vision* (2022), Springer, pp. 668–685. 3

[IMS*17] Ilg E., Mayer N., Saikia T., Keuper M., Dosovitskiy A., Brox T.: Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), pp. 2462–2470. 3

[JBPS11] Jacobson A., Baran I., Popovic J., Sorkine O.: Bounded biharmonic weights for real-time deformation. *ACM Trans. Graph. 30*, 4 (2011), 78. 3, 6

[JGR*18] Janai J., Guney F., Ranjan A., Black M., Geiger A.: Unsupervised learning of multi-frame optical flow with occlusions. In *Proceedings of the European conference on computer vision (ECCV)* (2018), pp. 690–706. 3

[KCG*14] Kazi R. H., Chevalier F., Grossman T., Zhao S., Fitzmaurice G.: Draco: bringing life to illustrations with kinetic textures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2014), pp. 351–360. 2

[KCGF14] Kazi R. H., Chevalier F., Grossman T., Fitzmaurice G.: Kitty: sketching dynamic and interactive illustrations. In *Proceedings of the 27th annual ACM symposium on User interface software and technology* (2014), pp. 395–405. 2

[KMR*23] Kirillov A., Mintun E., Ravi N., Mao H., Rolland C., Gustafson L., Xiao T., Whitehead S., Berg A. C., Lo W.-Y., et al.: Segment anything. *arXiv preprint arXiv:2304.02643* (2023). 4

[KRG*23] Karaev N., Rocco I., Graham B., Neverova N., Vedaldi A., Rupprecht C.: Cotracker: It is better to track together. *arXiv preprint arXiv:2307.07635* (2023). 3, 4, 6

[KW20] Kitagawa M., Windsor B.: *MoCap for artists: workflow and techniques for motion capture*. Routledge, 2020. 2

[LH22] Li Y., Harada T.: Non-rigid point cloud registration with neural deformation pyramid. *Advances in Neural Information Processing Systems 35* (2022), 27757–27768. 4, 6

[LYT10] Liu C., Yuen J., Torralba A.: Sift flow: Dense correspondence across scenes and its applications. *IEEE transactions on pattern analysis and machine intelligence 33*, 5 (2010), 978–994. 3

[MB08] Martinez B., Binefa X.: Piecewise affine kernel tracking for non-planar targets. *Pattern Recognition 41*, 12 (2008), 3682–3691. 3

[NŠM24] Neoral M., Šerỳch J., Matas J.: Mft: Long-term tracking of every pixel. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision* (2024), pp. 6837–6847. 3

[OAIS09] Okabe M., Anjyo K., Igarashi T., Seidel H.-P.: Animating pictures of fluid using video examples. In *Computer Graphics Forum* (2009), vol. 28, Wiley Online Library, pp. 677–686. 2

[PGC16] Patel P., Gupta H., Chaudhuri P.: Tracemove: A data-assisted interface for sketching 2d character animation. In *Proceedings of the 11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications: Volume 1: GRAPP* (2016), pp. 191–199. 2

[S*94] Shi J., et al.: Good features to track. In *1994 Proceedings of IEEE conference on computer vision and pattern recognition* (1994), IEEE, pp. 593–600. 3

[SBF*18] SU Q., BAI X., FU H., TAI C.-L., WANG J.: Live sketch: Video-driven dynamic deformation of static drawings. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (2018), pp. 1–12. 2

[SCBS13] SANTOSA S., CHEVALIER F., BALAKRISHNAN R., SINGH K.: Direct space-time trajectory control for visual media editing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2013), pp. 1149–1158. 2

[SHL*23] SHI X., HUANG Z., LI D., ZHANG M., CHEUNG K. C., SEE S., QIN H., DAI J., LI H.: Flowformer++: Masked cost volume autoencoding for pretraining optical flow estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023), pp. 1599–1610. 3

[SLT*19a] SIAROHIN A., LATHUILIÈRE S., TULYAKOV S., RICCI E., SEBE N.: Animating arbitrary objects via deep motion transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 2377–2386. 3

[SLT*19b] SIAROHIN A., LATHUILIÈRE S., TULYAKOV S., RICCI E., SEBE N.: First order motion model for image animation. *Advances in Neural Information Processing Systems 32* (2019). 3, 6, 8

[ST08] SAND P., TELLER S.: Particle video: Long-range motion estimation using point trajectories. *International journal of computer vision 80* (2008), 72–91. 3

[SYLK18] SUN D., YANG X., LIU M.-Y., KAUTZ J.: Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 8934–8943. 3

[SZL*23] SMITH H. J., ZHENG Q., LI Y., JAIN S., HODGINS J. K.: A method for animating children's drawings of the human figure. *ACM Transactions on Graphics 42*, 3 (2023), 1–15. 2, 8

[TZS*16] THIES J., ZOLLHOFER M., STAMMINGER M., THEOBALT C., NIESSNER M.: Face2face: Real-time face capture and reenactment of rgb videos. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 2387–2395. 2

[WCC*23] WANG Q., CHANG Y.-Y., CAI R., LI Z., HARIHARAN B., HOLYNSKI A., SNAVELY N.: Tracking everything everywhere all at once. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2023), pp. 19795–19806. 3

[WKZ18] WILES O., KOEPKE A., ZISSERMAN A.: X2face: A network for controlling face generation using images, audio, and pose codes. In *Proceedings of the European conference on computer vision (ECCV)* (2018), pp. 670–686. 3

[WRKS16] WEI S.-E., RAMAKRISHNA V., KANADE T., SHEIKH Y.: Convolutional pose machines. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition* (2016), pp. 4724–4732. 2

[WSML24] WU R., SU W., MA K., LIAO J.: AniClipart: Clipart animation with text-to-video priors. *arXiv preprint arXiv:2404.12347* (2024). 2

[WXSC04] WANG J., XU Y., SHUM H.-Y., COHEN M. F.: Video tooning. In *ACM SIGGRAPH 2004 Papers*. 2004, pp. 574–583. 2

[XKG*16] XING J., KAZI R. H., GROSSMAN T., WEI L.-Y., STAM J., FITZMAURICE G.: Energy-brushes: Interactive tools for illustrating stylized elemental dynamics. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (2016), pp. 755–766. 2

[XWSY15] XING J., WEI L.-Y., SHIRATORI T., YATANI K.: Autocomplete hand-drawn animations. *ACM Transactions on Graphics (TOG) 34*, 6 (2015), 1–11. 2

[ZZ22] ZHAO J., ZHANG H.: Thin-plate spline motion model for image animation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 3657–3666. 6