

VideoDoodles: Hand-Drawn Animations on Videos with Scene-Aware Canvases

EMILIE YU, Centre Inria d'Université Côte d'Azur, France

KEVIN BLACKBURN-MATZEN, Adobe, USA

CUONG NGUYEN, Adobe, USA

OLIVER WANG, Adobe, USA

RUBAIAT HABIB KAZI, Adobe, USA

ADRIEN BOUSSEAU, Centre Inria d'Université Côte d'Azur and TU Delft, France



Fig. 1. *Video doodles* combine hand-drawn animations with video footage. Our interactive system eases the creation of this mixed media art by letting users place planar canvases in the scene which are then tracked in 3D. In this example, the inserted rainbow bridge exhibits correct perspective and occlusions, and the character's face and arms follow the tram as it runs towards the camera.

We present an interactive system to ease the creation of so-called *video doodles* – videos on which artists insert hand-drawn animations for entertainment or educational purposes. Video doodles are challenging to create because to be convincing, the inserted drawings must appear as if they were part of the captured scene. In particular, the drawings should undergo tracking, perspective deformations and occlusions as they move with respect to the camera and to other objects in the scene – visual effects that are difficult to reproduce with existing 2D video editing software. Our system supports these effects by relying on planar canvases that users position in a 3D scene reconstructed from the video. Furthermore, we present a custom tracking algorithm that allows users to anchor canvases to static or dynamic objects in the scene, such that the canvases move and rotate to follow the position and direction of these objects. When testing our system, novices could create a variety of short animated clips in a dozen of minutes, while professionals praised its speed and ease of use compared to existing tools.

CCS Concepts: • **Computing methodologies** → **Graphics systems and interfaces**; *Computational photography*.

Additional Key Words and Phrases: video editing, motion design, interface, video depth

Authors' addresses: Emilie Yu, Centre Inria d'Université Côte d'Azur, France, emilie.yu@inria.fr; Kevin Blackburn-Matzen, Adobe, USA, matzen@adobe.com; Cuong Nguyen, Adobe, USA, cunghuyen@adobe.com; Oliver Wang, Adobe, USA, olwang@google.com; Rubaiat Habib Kazi, Adobe, USA, rhahib@adobe.com; Adrien Bousseau, Centre Inria d'Université Côte d'Azur and TU Delft, France, adrien.bousseau@inria.fr.

© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in *ACM Transactions on Graphics*, <https://doi.org/10.1145/3592413>.

ACM Reference Format:

Emilie Yu, Kevin Blackburn-Matzen, Cuong Nguyen, Oliver Wang, Rubaiat Habib Kazi, and Adrien Bousseau. 2023. VideoDoodles: Hand-Drawn Animations on Videos with Scene-Aware Canvases. *ACM Trans. Graph.* 42, 4 (August 2023), 12 pages. <https://doi.org/10.1145/3592413>

1 INTRODUCTION

Video doodles are an emerging mixed media art that combines video content with hand-drawn animations to produce unique and memorable video clips. Adding animated drawings and annotations to videos is an effective way to emphasize actions and motions of characters in the scene, to create visual explanatory material for educational purposes, or simply to make mundane videos more fun, personalized and attractive.

Artists typically create video doodles by drawing 2D animations frame-by-frame, using the video as an underlay. The main difficulty faced by artists following this manual workflow is to make the drawn content interact convincingly with the video content. Let us consider the example of adding a few animated “doodles” to a tramway video (see Fig. 1). When drawing a face over the tram, the artist must make sure that the eyes and mouth *follow* the tram as it moves across the frame. However, simply translating the corresponding doodles may not be enough, as their 2D scale, skew and orientation need to change to reflect the deformations caused by *perspective projection* as the tram comes closer to the camera and makes a turn. Even when doodling a static background element, such as the rainbow bridge across the rails, camera motion can yield non-trivial trajectories in the image plane, which the doodles must follow to appear fixed to the scene. Other difficulties arise in the presence of *occlusions*, which greatly contribute to the perception of 3D layout. For example, the tram should occlude part of the bridge to give the impression that it



Fig. 2. At the core of our interactive system is a novel tracking algorithm that deduces the 3D position and orientation of a planar canvas over an RGBD video given a few keyframes (green denotes a position keyframe, red denotes a position and orientation keyframe). Note how the canvas rotates to align with the direction of the trajectory and gets occluded by the body and the poles. Users create scene-aware doodles by drawing over the canvas in a simple 2D interface.

passes below it. To summarize, effective video doodles require the hand-drawn content to be *scene-aware*, meaning that they appear as if they were embedded in and synchronized with the content of the 3D scene captured in the video.

While computer vision methods can assist in tackling this challenge, existing solutions fall short in terms of accuracy, control, and accessibility to novice users. Professional video editing software offers a plethora of tracking algorithms to insert content that follows video motion [Adobe 2022; BorisFX 2022; Foundry 2022]. But these algorithms come with numerous parameters to tweak, and are prone to errors and drift in the presence of occlusion or in areas that lack reliable image features. Furthermore, tracking algorithms offer limited support for 3D motion estimation and are not sufficient to simulate occlusions between the inserted drawings and the video content. Artists reproduce such effects by decomposing the video into several layers using keyframed binary masks and rotoscoping. As an alternative to this 2D workflow, estimating the camera poses and the geometry of the captured scene can serve to insert virtual objects and re-render the video using 3D modeling software [Blender 2022b; Kopf et al. 2021; Schönberger and Frahm 2016; Zhang et al. 2021], resulting in accurate perspective and occlusions. But this 3D video editing workflow is still in its infancy as 3D UIs are cumbersome to use with 2D displays and input devices.

We propose a novel interactive system that combines the ease of 2D drawing with the strength of 3D computer vision to enable amateurs to create video doodles with a wide range of 3D effects. Our system takes as input casually-captured videos, which we preprocess with recent computer vision methods to obtain per-frame cameras [Schönberger and Frahm 2016], optical flow [Teed and Deng 2020], and dense depth maps [Kopf et al. 2021]. We leverage this geometric information to introduce *planar 3D canvases* on which users draw their doodles (Fig. 2). By anchoring these canvases to 3D points in the scene, our system automatically renders the doodles with convincing *perspective* and *occlusion* effects. We further augment the canvases with a dedicated tracking algorithm, such that the 3D position *and* orientation of the canvases follow the arbitrary moving objects they are anchored to. Importantly, we let users control this algorithm by keyframing the canvas position and orientation in image-space, effectively hiding most of the complexity of the underlying 3D representation. Our optimization then solves for the canvas 3D trajectory and orientation that best follows the scene motion while being constrained by the keyframes. Combined together, our technical and user interface contributions enable

even novices to turn their videos into convincing *video doodles* for a variety of applications, ranging from fun posts on social media to engaging illustrative tutorials.

2 RELATED WORK

We first discuss professional software for video editing as well as research work that aims at augmenting video editing by leveraging 3D geometry and motion estimation. We then discuss methods for point tracking, along with the user control they offer, as such tracking is at the core of our *VideoDoodles* system.

Professional video editing software. VFX artists and professional motion designers can choose from a rich array of powerful tools to insert animated virtual objects in captured scenes [Adobe 2022; Blender 2022b; BorisFX 2022; Foundry 2022; Runway 2022]. In particular, 3D camera estimation [Adobe 2022; Blender 2022b] and object tracking [Foundry 2022; KenTools 2022] are now mature technologies that accommodate many special effects, although these professional tools have a steep learning curve and sometimes require planning the shot at the time of capture – e.g. by placing markers for better tracking. Planar tracking [BorisFX 2022] is closest to our goal but is limited to tracking planes present in the scene, while our scene-aware canvases can move and orient differently from the surface of objects they are anchored to.

Leveraging 3D geometry. Automatic 3D computer vision methods hold the potential of bringing advanced editing tools to casual users, including re-rendering photos and videos with different camera properties [Klose et al. 2015; Liu et al. 2022a], stabilizing complex camera trajectories [Kopf et al. 2014], creating 2.5D parallax effects [Kopf et al. 2020]. Closer to our target application are systems that leverage 3D pose and geometry estimation for stylizing moving objects [Snavely et al. 2006] or for drawing over their surface [Kasten et al. 2021; Rav-Acha et al. 2008]. But these systems do not allow drawing *away* from existing surfaces, which is important to produce a variety of effects, as in Fig. 1 where the bridge is drawn above the ground, and the arms are drawn around the tram.

A large body of multi-view 3D reconstruction algorithms, including Structure-From-Motion [Schönberger and Frahm 2016] and SLAM [Mur-Artal et al. 2015], can be used to find reliable sparse 3D scene points and camera poses, but these methods are designed only to reconstruct static components of videos. Furthermore, sparse reconstruction methods could support some features of our approach, but occlusion requires per-pixel geometry, which these

methods do not produce. Recent work has focused on estimating a consistent *dense* depth representation from videos with dynamic elements [Kopf et al. 2021; Luo et al. 2020; Zhang et al. 2021]. These methods were applied to virtual object insertion, but their demos were created by loading the 3D reconstruction in professional 3D rendering and compositing software such as *Blender* or *Nuke*. Our approach uses the video depth reconstructed by these methods to track, orient and render drawing canvases in the scene, allowing users to insert scene-aware doodles with a simple 2D user interface.

Inserting virtual objects into captured scenes is also key to augmented reality applications [Apple 2022; Du et al. 2020; Valentin et al. 2018]. In such applications, users typically place virtual objects in the scene through in-situ direct interactions [Leiva et al. 2020], by placing tracking targets on objects [Liao et al. 2022], or by selecting pre-defined targets – e.g. detected planar surfaces [Apple 2022]. Our system is designed to be used as a post-processing video editing tool rather than as an in-situ augmented reality tool. This positioning allows us to offer more precise and expressive, keyframe-controlled trajectories for the inserted doodles.

Leveraging scene motion. Our approach draws inspiration from prior work that leverages motion tracking to augment videos with hand-drawn annotations and with direct navigation along motion trajectories [Dragicevic et al. 2008; Goldman et al. 2006, 2008; Nguyen et al. 2013]. Subsequent work by Suzuki et al. [2020] allows the insertion of responsive sketches that follow or react to tracked motion. We extend this family of work by enabling more direct user control over the tracking results through keyframing of both position and orientation of drawing canvases, and by accounting for the 3D geometry of the scene over which the doodles are drawn.

In human-computer interaction, many approaches leverage human pose estimation and tracking [Cao et al. 2019] to offer dedicated visual effects and visualizations [Mayer et al. 2021; Saquib et al. 2019; Zhang et al. 2018]. *PoseTween* [Liu et al. 2020] is a system for animating drawings over human action videos which relies on human pose for 2D interpolation of user-provided keyframes. In contrast to these domain-specific solutions, our approach tracks 3D trajectories of arbitrary objects and supports 3D effects, including perspective transformations and occlusions of the inserted doodles.

Tracking arbitrary points in videos. While tracking bounding boxes [Henriques et al. 2014], segmentation masks [Oh et al. 2018], planar regions [Liang et al. 2018], or semantic keypoints for objects of a known class [Xiang et al. 2018] has been studied thoroughly, the more general problem of tracking arbitrary surface points across a video has received surprisingly little attention, as stressed by the recent *TAP-Vid* benchmark for long-range point tracking [Doersch et al. 2022]. Aiming at helping novices to create cartoon animations, *Live Sketch* [Su et al. 2018] guides a point tracker using keyframing to extract motion from a video clip and transfer that motion to a drawing. Their algorithm follows prior work that casts user-guided point tracking as an optimization where the point trajectory corresponds to the shortest path in a directed graph formed by the video pixels [Amberg and Vetter 2011; Buchanan and Fitzgibbon 2006]. This formulation was also used by Doersch et al. [2022] to annotate ground-truth trajectories for the *TAP-Vid* benchmark. We extend this formulation to the case of 3D point tracking by accounting for

the depth and camera pose at each frame. Furthermore, we leverage the resulting trajectory to also orient the 3D canvas according to the direction of the moving object. Finally, rotoscoping algorithms track curves along contours in a video, guided by user-provided keyframes [Agarwala et al. 2004; Li et al. 2016]. These algorithms rely heavily on the smoothness and contrast of image contours, while we focus on tracking points that may lie in feature-less regions.

3 CHALLENGES IN VIDEO DOODLES AUTHORING

We investigated current video doodling practice by surveying 20 online tutorials (T1-20, see complete list in supplemental materials) and by discussing the most common techniques with two professional motion designers (P1 and P2) having 17 and 15 years of experience with 2D motion graphics tools (*Adobe After Effects*, *Adobe Character Animator*), and animating with code (*Processing*, *CSS*).

2D animation workflow. The majority of tutorials we found [T1-13] describe a process akin to traditional 2D animation. The artist imports the video in an animation software and proceeds to draw on one or multiple overlaid layers, for every frame of the video. To streamline this workflow, artists can copy and transform the drawings from one frame to the next, or can rely on smooth interpolation of the drawn strokes between sparse keyframes [T11,12]. This is a difficult process that requires significant manual tweaking: “[It’d be] probably a lot of changing stuff on a per frame basis to make sure it catches up properly, you’d need to just be like, on this frame, you’re here, and change the drawing.” (P2)

Dealing with occlusions requires either erasing occluded parts of the doodles [T2], or creating binary masks that follow the occluding shape across the video [T12,13]. Another major challenge resides in synchronizing the drawings with events in the video. Artists achieve such synchronization by keeping the video visible as an underlay to provide visual context, and by taking notes on the precise timing of key events to plan the animation [T1].

Motion and camera tracking. Artists rely on experience and intuition to draw doodles such that they appear to have the correct motion and perspective in the scene, which can be challenging even for simple camera motions: “Honestly that’s a trial and error process. It’s me saying OK, let me try this position keyframe and see if [this] looks interesting like this and if not, I’ll just keep on doing it until I get something that I feel looks right.” (P1)

To ease this task, artists are faced with diverse tracking tools to pick from. One-point tracking [T14,20] is easy to set up but does not take orientation or perspective changes into account, whereas 3D camera tracking [T16] is well suited to place static elements in the scene but cannot track moving objects. When using these algorithms, artists often correct the inferred trajectories by deleting erroneous parts and replacing them with interpolated keyframes [T15], by providing corrective keyframes to the algorithm [T17-20], or by tuning tracking parameters [T17-19]. “If at a certain point I move too fast or something happens, it’s blurry and the tracking goes off and then it’s all over the place, this little tracker. So then I have to do some manual edits and put it back and just make sure it works OK. And yeah, it’s just a little bit cumbersome.” (P1)

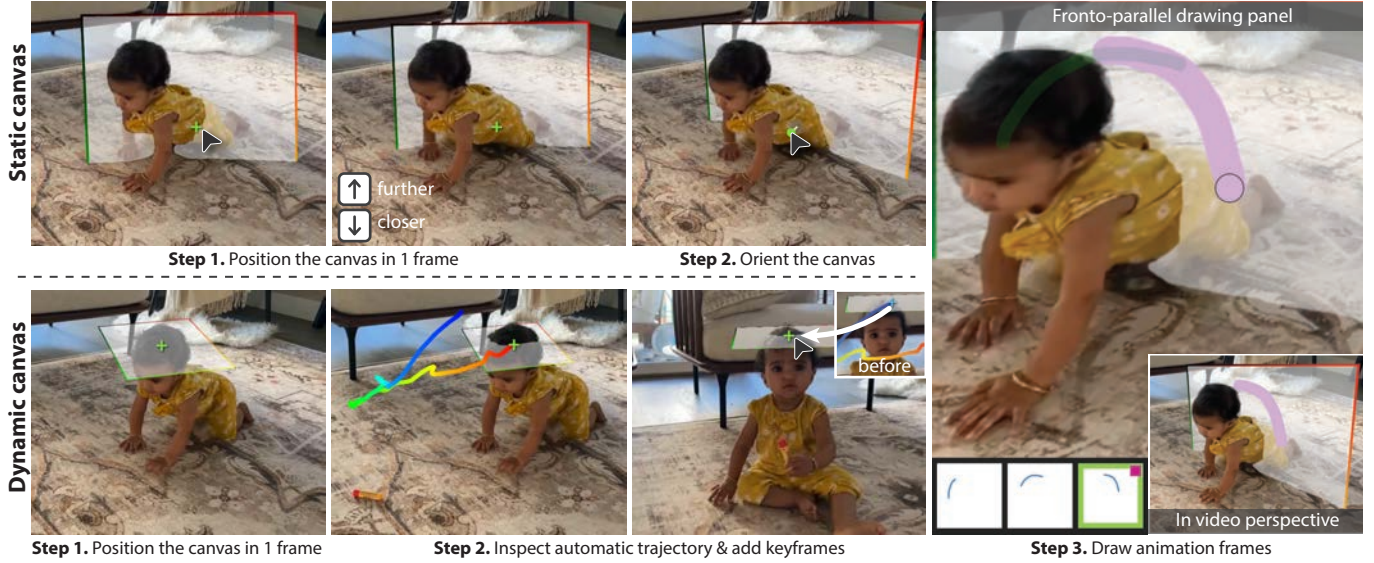


Fig. 3. Main features of our user interface. The user can place a static (top) and dynamic (bottom) canvases in the scene by dragging their centers (green cross). Canvases can be moved closer or further away from the camera, and can be oriented using a gauge figure. Dynamic canvases are placed in one or more keyframes, from which our system deduces a 3D trajectory (time is color-coded with a jet ramp). The user can then draw doodles in a dedicated panel where the canvas is rectified to be fronto-parallel, alleviating the need to draw in perspective.

Due to these challenges, both professional designers we interviewed judged that it would take them several hours to create a video doodle of a few seconds using the tools they are familiar with.

Design goals. Based on the above observations, we define the following design goals for our interactive *VideoDoodles* system:

G1. Scene-aware doodling: Maintain a 2D animation workflow, yet streamline the creation of perspective and occlusion effects.

G2. Flexible motion tracking: Support the tracking of both the moving camera and moving objects in the scene.

G3. User control: Offer control on the motion and timing of the doodles using the established interaction paradigm of keyframing.

4 USER WORKFLOW

Our system follows the principles of *mixed-initiative* user interfaces [Horvitz 1999] as it leverages 3D computer vision to offer significant value-added automation, combined with simple 2D interaction mechanisms to enable amateurs to efficiently guide and refine the end result. We first describe a typical interactive session with this system, illustrated in Fig. 3. We provide a recording of this session in the accompanying video, along with additional animated results. We detail the algorithms behind our system in Section 5.

Input. Our system takes as input a video clip representing a single camera shot. In a preprocess, we augment this input with per-frame camera pose, depth map and optical flow (Section 5.1).

Planar canvases. We fulfill our first design goal (G1) by embedding the doodles into *planar 3D canvases* that are placed in the scene via 3D rigid transformations. On the one hand, users can easily draw strokes on planar canvases using a 2D interface. On the other hand, we can render these canvases with correct perspective and

occlusions in all video frames thanks to the estimated camera poses and depth maps. While this simple mental model cannot represent non-planar curves, advanced animation effects can be achieved by animating the strokes within the canvas, as in traditional 2D animation. Many sketch-based modeling systems rely on similar canvases to lift 2D strokes to 3D [Bae et al. 2008; Blender 2022a; Canvas 2022; Dorsey et al. 2007; Leiva et al. 2020; Li et al. 2017].

Placing a canvas. Our interface allows users to place *static* or *dynamic* canvases in the video. Static canvases have a fixed position and orientation in the scene, and as such only react to camera motion (G2). Users place static canvases by dragging and rotating them over one of the video frames, and by optionally adjusting their scale relative to the scene. Our system automatically sets the center of the canvas such that it lies at the same depth as the underlying surface. Users can over-write this default depth using a slider.

Dynamic canvases follow moving objects in the scene (G2). To obtain such tracking, users position and orient the canvas in at least one keyframe. Our system then infers a 3D trajectory that follows the scene point under the center of the canvas across the video, and it orients the canvas relatively to the direction of that trajectory. If needed, users can refine the result by adjusting the position and orientation of the canvas in additional keyframes (G3). By default, users only need to specify the 2D position of the canvas in each keyframe, and our algorithm takes care of deducing the depth that yields the best tracking in 3D. As with static canvases, users can over-write the inferred depth using a depth slider. Since our system embeds the canvas in 3D, the canvas automatically appears bigger or smaller as it moves in depth, without requiring an explicit keyframing of scale by the user.

Drawing on a canvas. Canvases can undergo significant foreshortening depending on their 3D orientation, and prior studies have shown that even experienced artists struggle to draw accurately over slanted surfaces [Schmidt et al. 2009]. Our interface avoids drawing in perspective by providing a secondary drawing panel, where the canvas is displayed under an orthographic, fronto-parallel view (G1). However, drawing on a blank canvas would make it difficult for users to align and synchronize their doodles with the video content. Our interface provides the necessary contextual cues by rectifying the portion of the video covered by the canvas and by displaying it as an underlay in the drawing panel. We help users find a suitable frame to draw onto by reporting the amount of foreshortening of the canvas for each frame of the trajectory, less foreshortened canvases yielding less distortion of the video after rectification. Furthermore, our interface directly shows occlusion effects as strokes are drawn to help users assess the 3D insertion of their doodles. Finally, we provide a basic frame-by-frame 2D animation tool within the drawing panel, which allows users to create simple loops that are repeated along the video. This tool also includes a simple *onion skinning* feature – displaying previous and next frames as semi-transparent overlays – to facilitate drawing animated sequences.

5 ALGORITHMIC COMPONENTS

At the core of our system is a tracking algorithm that leverages camera and scene motion, depth estimation, and user-provided keyframes to find the 3D trajectory of a scene-aware canvas.

5.1 Pre-computing depth and motion

Our method is built upon 3D video reconstruction; given a set of input frames, per-frame camera poses and world-aligned depth maps are computed. We use our own re-implementation of *Robust Consistent Video Depth Estimation* [Kopf et al. 2021]. This approach first estimates camera pose and a projection matrix for every frame using COLMAP [Schönberger and Frahm 2016]. It then uses a deep single-image depth predictor to compute scale and shift invariant depth maps, and solves for a geometric optimization that aligns these depth maps into consistent world coordinates, yielding a dense, temporally consistent geometric reconstruction. This approach internally uses optical flow [Teed and Deng 2020] computed between consecutive frames, which we also save for later use.

Equipped with the camera matrix and depth map for each frame t , we compute for every pixel p_i^t its 3D position P_i^t by unprojection. Similarly, we lift the optical flow vectors v_i^t to 3D to obtain scene flow vectors V_i^t .

5.2 Keyframe-based tracking

Given one or more user-specified canvas keyframes (consisting of a position and orientation), our goal is to recover a trajectory that

- Makes the canvas follow the scene point it is attached to, such that users do not have to reproduce that 3D motion by hand.
- Interpolates between keyframes, such that users have full control and can (optionally) deviate from motion tracking if desired.

We cast this problem as a series of optimizations, where the variables are the 3D positions and orientations of the canvas in each frame, the keyframes are expressed as hard constraints, and the motion

tracking is expressed as soft objectives to allow for correction by the user. We first detail how our approach tracks 3D positions, and then explain how we extend it to additionally track 3D orientation.

Tracking 3D positions. Our method builds upon related keyframe-based 2D tracking algorithms that search for trajectories with coherent appearance and motion within the space-time video volume (Fig. 4a) [Amberg and Vetter 2011; Buchanan and Fitzgibbon 2006; Doersch et al. 2022; Su et al. 2018]. These methods build a directed graph that connects each pixel in frame t to every pixel in frame $t+1$, and assigns each edge a weight that is proportional to the difference in appearance and position between the two pixels [Amberg and Vetter 2011; Su et al. 2018], or to the agreement with pre-computed optical flow motion vectors [Doersch et al. 2022]. Frames with a keyframe have only one node in this graph, corresponding to the pixel specified by the user to be the center of the canvas, which forces the trajectory to adhere exactly to the constraints. All nodes at the first frame and last frame are connected to a super-source and sink node respectively, and the trajectory is computed as the shortest path from source to sink.

We extend this family of algorithms to leverage the 3D information we extracted during preprocessing, and to generate a 3D trajectory. Specifically, we encourage the *scene-space* trajectory to align with the scene flow by expressing the edge weight between pixels in consecutive frames as

$$w(p_i^t \rightarrow p_j^{t+1}) = \left\| (p_j^{t+1} - p_i^t) - V_i^t \right\|^2. \quad (1)$$

Computing this weight as a 3D distance prevents the trajectory to jump between objects that lie close together in image space yet are far apart in depth. We improve the robustness of this formulation by removing nodes (pixels) that are too close to motion or depth discontinuities, as detected by computing the agreement between forwards and backwards optical flow for the former, and by computing the gradient of the depth map for the latter. This safeguard further reduces the risk of crossing object boundaries.

Building the graph over the entire video volume would be prohibitive. We drastically reduce complexity in two ways. First, we trim a large part of the graph by augmenting each pixel with an appearance vector a_i^t – which we compute as a set of deep visual features [Jabri et al. 2020] – and by only retaining, for every frame, the 10% pixels most similar to the keyframes in appearance. This trimming also helps the tracking algorithm to focus on the region of interest. Second, we reduce the size of the graph by working at the resolution of the deep visual feature maps, where one pixel corresponds to a patch of 10×10 pixels in the original video. With these settings, building the graph and finding a shortest path with Dijkstra’s algorithm for 80 frames of a 1200×674 video takes around 3 seconds on a 2.4GHz Intel i5 MacBook Pro with 16GB of memory.

Recovering stable, high-resolution trajectories. Prior methods directly output the nodes of the shortest path as the tracking trajectory [Amberg and Vetter 2011; Buchanan and Fitzgibbon 2006; Doersch et al. 2022; Su et al. 2018]. However, this initial *discrete* trajectory often suffers from jitter and drift over featureless surfaces or in the presence of occlusions, which are exacerbated when working with a low-resolution graph (see Fig. 5). We correct for such instability by

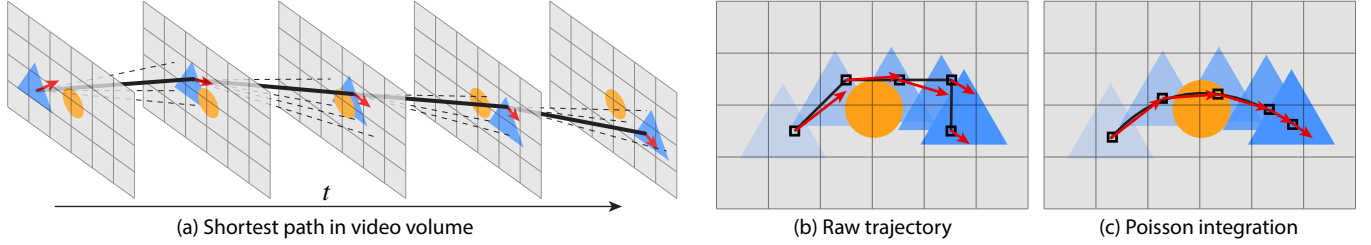


Fig. 4. Schematic illustration of our tracking algorithm. We extract an initial trajectory as the least-cost path in the directed graph connecting each keyframed pixel to similar pixels in consecutive frames (a). This trajectory often jitters over the object to track due to occlusions, lack of visual features, and our use of a low-resolution graph (b). We recover a stable trajectory by integrating the scene flow sampled along the initial trajectory at full resolution (c, red arrows).

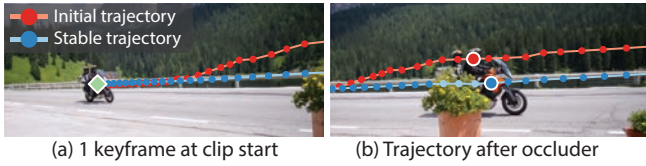


Fig. 5. Effect of Poisson integration. Given a single keyframe as input (a), the shortest-path algorithm yields an initial trajectory that drifts over the bicycle as it gets occluded by the foreground plant (b, red). Since the entire bicycle undergoes the same translation, integrating the scene flow vectors sampled along the initial trajectory removes the drift, resulting in a stable trajectory that runs behind the occluder (b, light blue).

observing that large portions of objects often move coherently, such that neighboring pixels have similar motion vectors. Following this intuition, we sample the scene flow along the initial trajectory to get an estimate of the trajectory's derivatives $\{V^1, \dots, V^T\}$. We then reconstruct a stable, high resolution trajectory by solving for the continuous 3D positions $\mathbf{P} = \{P^1, \dots, P^T\} \in \mathbb{R}^{3 \times T}$ of the canvas that satisfy these derivatives, which corresponds to a Poisson problem where the user-provided keyframed pixels $\{\tilde{p}^k\}$ act as boundary constraints (Fig. 4b,c). We formulate these constraints via the camera projection operator Π^k , such that the trajectory passes through 3D points P^k that reproject exactly on keyframed pixels. We further regularize the problem by encouraging the trajectory to run close to the unprojected keyframed pixels $\{\tilde{p}^k\}$, yielding:

$$\min_{\mathbf{P}} \sum_t \left\| (P^{t+1} - P^t) - V^t \right\|^2 + \lambda_{\text{depth}} \sum_k \left\| \tilde{p}^k - P^k \right\|^2, \quad (2)$$

such that $\tilde{p}^k = \Pi^k(P^k)$.

In cases where the user also specifies the depth of a keyframe via the depth slider, we directly enforce that the trajectory passes through the resulting 3D point by setting the constraint to $\tilde{p}^k = P^k$. Such constraints are particularly useful in scenarios where the user wants to anchor a canvas to the hidden side of an object, yet wants the canvas to follow the overall motion of that object (e.g. the left arm of the flamingo in Fig. 12). Our Poisson formulation reconciles the initial trajectory, which by definition of the graph nodes only passes through visible points of the object, with the user-specified depth values.

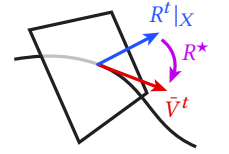
Tracking 3D orientations. Users can also control the orientation of a dynamic canvas along its trajectory by specifying a rotation matrix in one or more keyframes $\{\tilde{R}^k\} \in SO(3)^K$, from which our system deduces a sequence of rotation matrices $\{R^1, \dots, R^T\} \in SO(3)^T$, one for each frame of the trajectory.

Following our second design goal (G2), we want the orientation of the canvas to follow the orientation of the moving object it is tracking, as illustrated in Fig. 6. Let us first consider the restricted case where the user orients the canvas perpendicularly to the trajectory of the object. Let us further assume that the canvas normal is given by the first axis of the orthogonal frame encoded by the canvas rotation matrix, denoted as $R^t|_X$. Given the normalized scene flow vector \tilde{V}^t at each frame t , we can encourage the canvas' normal to align with the trajectory by minimizing $\|R^t|_X - \tilde{V}^t\|^2$.

However, we also want to give users the freedom to choose the *relative* orientation of the canvas with respect to the moving object, for instance to make the doodle parallel rather than perpendicular to the trajectory. We achieve this behavior by introducing an (unknown) rotation matrix R^* that transforms the canvas relatively to the motion trajectory, yielding:

$$\min_{\{R^t, R^*\}} \sum_t \|(R^t R^*)|_X - \tilde{V}^t\|^2 + \lambda_{\text{smooth}} \sum_t \|R^{t+1} - R^t\|^2, \quad (3)$$

such that $R^k = \tilde{R}^k$.



The first term encourages the canvas to align with the tangent of the trajectory, up to the relative orientation R^* . The second term prevents the canvas to twist unnecessarily as it travels along the trajectory [Stanko et al. 2017]. Finally, the constraint ensures that the canvas perfectly respects the keyframed orientations.

While the smoothness term brings robustness to noise in the scene flow, it tends to damp the rotation of the canvas around sharp turns of the trajectory. We address this issue by splitting the trajectory in the presence of abrupt turns (which we detect as local minima in velocity, i.e., when $|V^t| < 0.2 \times \max(|V^t|)$) and by assigning a different matrix R^* to each segment. A sudden change of direction in the trajectory is then captured by an instantaneous change of R^* rather than by progressive changes of R^t .

Our formulation bears resemblance with algorithms for interpolating orthogonal frames along 3D curves [Boumal 2013; Stanko et al. 2017], although such methods do not include the additional

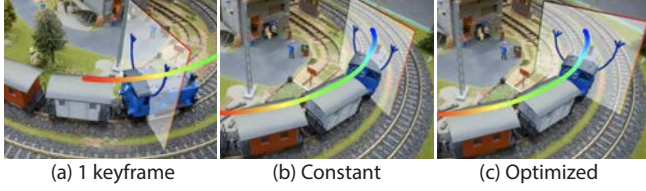


Fig. 6. In this example, the user orients the canvas to be perpendicular to the motion trajectory in one keyframe (a). Keeping this orientation constant in scene space produces an implausible result as the trajectory turns to follow the tracks (b). Our optimization rotates the canvas to preserve its orientation *relative* to the trajectory (c).

rotation matrix R^* , which is key to offer users control on canvas orientation in our context. Similarly to these prior methods, we solve our optimization over the manifold $SO(3)^T$ of rotation matrices using the Riemannian Trust Region algorithm implemented in the Pymanopt library [Townsend et al. 2016], with parameters described by Boumal [2013]. Following their recommendations, we initialize orientations with a spherical linear interpolation of the keyframe quaternions. While solving the small linear system in Equation 2 is very fast, finding a local minimum of Equation 3 can take up to a dozen of seconds, depending on the number of frames and segments (e.g., 1.5" for 1 segment of 80 frames in Fig. 6, vs. 9" for 3 segments and 143 frames in Fig. 3).

Parameter setting. We used a fixed set of parameters for all our results. A weak regularization on canvas depth $\lambda_{\text{depth}} = 0.01$ mainly serves when the user does not provide depth keyframes. In contrast, a high regularization on orientation smoothness $\lambda_{\text{smooth}} = 10$ prevents the canvas to align with every little turn in the trajectory.

6 RESULTS AND EVALUATION

Fig. 12 showcases a variety of video doodles we created with our system based on clips from the DAVIS datasets [Perazzi et al. 2016; Pont-Tuset et al. 2017], or captured by us. These results cover various application scenarios, ranging from humorous augmentations of casual videos (Flamingo cocktail party, BMX), to informative or instructive (Climbing, Tennis), and other forms of annotations and highlights (Travel vlog, Swinging). Importantly, these results exhibit numerous occlusions between real and drawn content (text in the Travel vlog, legs of the flamingo, frame of the swing), as well as complex trajectories, both in terms of position (Climbing, Comics parkour) and orientation (Swinging). We strongly encourage readers to look at the corresponding videos in supplemental materials.

We first evaluate our tracking algorithm quantitatively on a recent benchmark. We then evaluate our interface qualitatively by asking novice and professional users to create video doodles.

6.1 Tracking accuracy

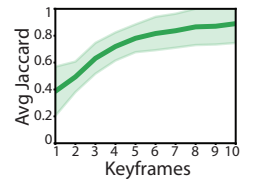
In the absence of a benchmark for 3D point tracking, we evaluate the accuracy of our algorithm on the recent TAP-Vid benchmark [Doersch et al. 2022], which provides a set of ground truth 2D trajectories (called *tracks*) and occlusion flags for the task of tracking arbitrary points in videos. Since our primary contribution is the

design of an interactive system for video doodling, the goal of this evaluation is not to claim improvement over fundamental tracking algorithms, but rather to demonstrate that the tailored algorithm we designed to support our user interface is on par with recent algorithms developed for a similar – albeit not identical – task.

Our approach requires a 3D reconstruction, but current algorithms for camera calibration from a single monocular video can fail when there is not enough camera motion, or too many dynamic objects. We therefore restrict our evaluation to a subset of the TAP-Vid-DAVIS dataset consisting only of those videos whose 3D reconstruction succeeded, yielding 24 (out of 30) annotated videos for a total of 575 tracks of 62 frames on average.

Single keyframe. First, we evaluate the performance of our algorithm in the case where only one keyframe is specified. Following the *strided* setting of TAP-Vid, we sampled each track every 5 frames to obtain a set of query keyframes, then ran our algorithm for each of these queries to obtain multiple predictions. The final metric is computed as an average over all queries for all tracks and all videos. Since our algorithm outputs a 3D trajectory, we project each point to 2D using the cameras and test for occlusion by checking if the point lies outside the image frame, or if the 3D point lies behind the depth map. Table 1 summarizes the outcome of this evaluation using the metrics recommended by Doersch et al. [2022] – average Jaccard, average position accuracy of visible points, and binary occlusion accuracy (higher is better). Our method outperforms the state-of-the-art TAP-Net method proposed by Doersch et al. [2022] on the two first metrics, and it achieves comparable accuracy on occlusion. We also include the scores of the initial trajectory produced by the shortest-path algorithm (Ours w/o Poisson), highlighting the benefit of including the Poisson integration step in our method. Finally, we provide the scores obtained by our method when fixing the keyframe to be the first visible ground-truth 2D position of each track (Ours 1kf, first), as this setting is closer to the way users would interact with our system. As noted by Doersch et al. [2022], this keyframe selection strategy yields lower performance.

Multiple keyframes. Since we designed our algorithm with interactive control in mind (G3), we now evaluate how adding keyframes improves the resulting trajectory. The last row of Table 1 (Ours 2kf) reveals that positioning one keyframe at the start of the visible trajectory, and another keyframe at the end, suffices to improve the quality of the track significantly. We additionally ran an experiment where we progressively increased the number of keyframes for each of the 25 tracks of one representative video (50 frames total), adding each new keyframe as the mid-point of the two most distant existing keyframes. The inset shows that the tracking accuracy quickly increases with additional keyframes, and eventually saturates as keyframes get close together.



Qualitative comparison. Fig. 7 provides a visual comparison between a few of our tracks and the ones predicted by TAP-Net, along with the corresponding ground truth. We include additional video comparisons as supplemental materials. Overall, the tracks produced by our method exhibit less jitter than the ones predicted by TAP-Net



Fig. 7. Visual comparison of our tracking trajectories against *TAP-Net* [Doersch et al. 2022]. For visualization purposes, we compensate for camera motion by rendering 3D trajectories projected onto the first frame of the video clip. Since the ground truth and *TAP-Net* trajectories are provided as 2D image-space points, we unproject them to 3D using the same cameras and depth maps as used by our method. Our method produces 3D trajectories that are more precise (higher Average Jaccard, AJ) and smoother than *TAP-Net* (top row). While our method can lose track of a point due to occlusion (bottom left), adding a keyframe often suffices to resolve such issues (inset). Both our method and *TAP-Net* struggle to recover subtle composed motions, such as the rotation of the car’s wheel (bottom, right).

Table 1. Evaluation of our tracking algorithm on 24 videos of the *TAP-Vid DAVIS* benchmark with a single keyframe (1kf). Adding an extra keyframe (2kf) yields a significant improvement in all metrics (higher is better). We report the results of *TAP-Net* [Doersch et al. 2022] computed on the 24-videos subset for which we obtained a successful 3D reconstruction.

Method	Avg Jaccard (\uparrow)	$< \delta_{avg}^x$ (\uparrow)	Occlusion (\uparrow)
<i>TAP-Net</i> (1kf, strided)	37.2%	52.7%	80.2%
Ours (1kf, strided)	40.8%	59.2%	80.6%
Ours (1kf, strided, w/o Poisson)	18.3%	31.6%	75.1%
Ours (1kf, first)	31.6%	50.9%	75.2%
Ours (2kf)	45.5%	67.3%	78.3%

– a property that is essential to achieve convincing video doodles. We also stress that *TAP-Net* only supports a single query point per track, while our method allows users to correct tracking failures due to occlusion or drift by adding keyframes. Moreover, while *TAP-Net* predicts occlusion at a single 2D pixel, our 3D trajectories enable depth testing against a predicted dense depth map to render occlusions over the entire planar canvas.

6.2 User study

We conducted a study with 7 users, 5 being novices while the other 2 being the professional motion graphics designers we interviewed during our formative study (Section 3). The five novices participated in-person by drawing on a Wacom Cintiq 16 tablet, whereas the two professionals participated remotely using only a mouse or trackpad.

Participants were first introduced to our system via a short video tutorial, and were then given two tasks:

Task 1 - Goal-directed (10 minutes).

Participants are shown an example video doodle that they have to reproduce, shown in inset. The result should be made of a static doodle (bridge) and a dynamic doodle (cloud of steam coming out of the locomotive). This task was designed so that users could experience all the main features of our system, and was inspired by the goals of online tutorials (see T3 in supplemental materials).

Task 2 - Open-ended (30 minutes).

Participants are asked to create two novel video doodles by choosing among 15 short video clips (3” duration on average).

Finally, participants answered a questionnaire about the different features of our system. While we conducted this study with a small group of participants, the collected material represents 4.5 hours of interaction with our system, corresponding to a total of 49 canvases. In the following section, we detail important insights we gained from this data, including typical usage of the system’s features, unexpected user behavior, and suggestions for improvement. We provide all 21 video doodles as supplemental materials to illustrate the type of effects that users of our system could create after only a few minutes of practice.

Everyone can create video doodles with our system – fast. Fig. 11 provides a gallery of results created by the participants during the open-ended task, along with their completion time. Creating the moderately complex video doodle of Task 1 only took 10’30” on average (sd = 03’07”). Creating a video doodle from scratch – including



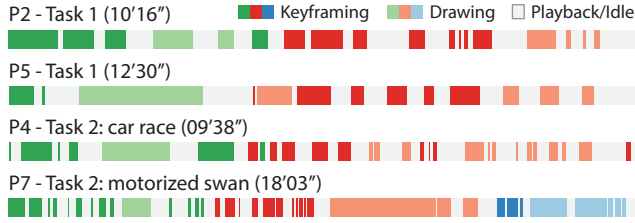


Fig. 8. Timelines of performed operations. In this visualization, different hues (green, red, blue) depict different canvases, and brightness differentiates keyframing a canvas (dark colors) from drawing on a canvas (light colors). Participants roughly spent as much time keyframing as drawing, and sometimes alternate between the two operations (P4).

time to brainstorm and experiment ideas – took an average of 14'00" ($sd = 07'05''$). Both of the professional participants emphasized the speed with which they could create video doodles with our system: *"I feel like we just raced through all those different things because it was so easy and quick to use. If I'm working in After Effects, I have to spend a lot more time worrying about the tracking, or about whether the canvas is going behind this thing or this other thing. Here it just happened automatically and it worked how I thought it should."* (P1)

Furthermore, all participants were highly satisfied with the video doodles they created (score of 4.7 on a 5-point Likert scale). Many participants were enthusiastic at the idea of using our prototype with videos that they would capture themselves.

Automating tracking and 3D rendering frees up time for doodling. By analyzing usage logs, we find that participants spend around half of their time doodling and creating compelling frame-by-frame animations (see Fig. 8 for a temporal breakdown of typical sessions). On average, 49% of a session was dedicated to drawing ($sd = 17\%$, $min/max = 24/89\%$, measured as the time spent with the drawing panel open). Even novices picked up quickly the concept of frame-by-frame animation, and all succeeded in creating such animation clips. On average, participants drew 3.3 ($sd = 1.8$) frames per canvas, with 6.0 ($sd = 9.2$) strokes per frame. We also observe that participants sometimes alternate phases of drawing with keyframing to adjust the trajectory of a canvas after drawing on it, which happened in 19 out of the 49 canvases created over all video doodles.

These observations suggest that we achieved our design goals **G1** and **G2** – by decoupling 2D drawing from 3D in-scene embedding and tracking, users are able to focus on the creative task of doodling: *"What I found myself spending more time on was redoing the drawings to make them laser or like add more stuff. Since the 'hard part' – the tracking – is being taken care of, I could focus on drawing the shapes that I want."* (P2)

Nevertheless, some participants would have liked to be able to draw directly over the video in complement to the rectified drawing panel, or have the option to vary the brush shape and texture, two features that we did not implement in our prototype.

Combining tracking with keyframing keeps users in control. All participants quickly grasped the concept of keyframing to control the position and orientation of the canvases. They used an average of 3.3 ($sd = 3.0$) position keyframes and 2.5 ($sd = 1.8$) orientation

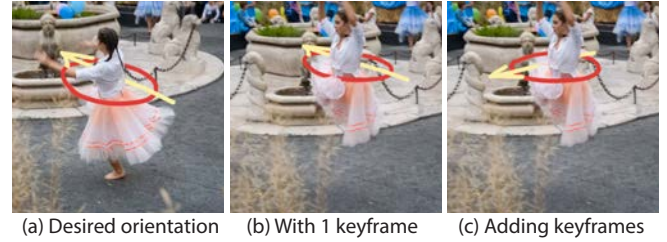


Fig. 9. Limitation. The dancer rotates on herself as she jumps. Given a single keyframe, our tracking algorithm follows the upward motion of the body, but not its rotation (a-b). The user can achieve the desired effect by keyframing the canvas rotation (c).

keyframes per dynamic canvas. Making the canvas track a particular object required 1 or 2 position keyframes in the majority of cases. Out of the 26 dynamic canvases, only 6 canvases required 5 keyframes or more. This extra work was needed when the tracked object gets occluded along its trajectory (P1 - swing, P2 - dancer's hand), or when the desired trajectory does not strictly follows a single point of a moving object (e.g., P5 - lama where the backpack first follows the lama's flank, then its rear).

The effort that participants put into perfecting their results with more keyframes shows that the interface gave them a sense of agency in the tracking task, in accordance with our design goal **G3**: *"Overall I felt like I was in control, and that it was making smart automatic decisions for me."* (P1) *"Adjusting the plane with the keyframes was super simple, and it's great that you can adjust things to make them perfect. You get the [automatic] trajectory and that does most of the work for you, but then you can tweak it."* (P2)

Still, expert users value the more advanced features available in professional software: *"With After Effects tracking tool, there are so many more options that you have, so I feel like I have more control over the tracking at a pixel level, since I can zoom in and make it perfect."* (P1) One of the experts also reported having difficulty positioning the canvas in depth at first, which suggests possible improvement in visualizing or controlling depth.

Working around limitations. Our choice of embedding strokes in planar canvases prevents the creation of non-planar doodles. Several participants found creative solutions to work around this limitation, such as placing two orthogonal canvases to depict sections of the cloud of steam (Fig. 11, P6 - train), or drawing different doodles to depict the side and back views of a backpack on the lama, taking care of switching between these doodles to match the viewpoint in the video (Fig. 11, P5 - lama).

6.3 Limitations.

Our system is limited to planar canvases and as such cannot produce 3D freeform strokes, for instance to draw a swirling ribbon around an object. Future work could explore the definition of non-planar canvases based on parametric shapes [Ikeda and Fujishiro 2021] or height fields [Arora et al. 2018]. Furthermore, our tracking algorithm assumes that the object to track is oriented towards its dominant motion trajectory. Fig 9(a,b) illustrates a case where this assumption does not hold, as the dancer rotates on herself as she

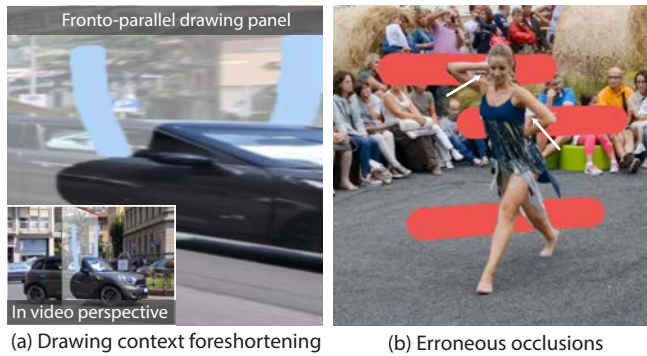


Fig. 10. Limitations. When the canvas is highly foreshortened, the video context appears with strong distortions in the fronto-parallel drawing panel (a). We rely on existing video depth estimation methods to handle occlusions. While most of the silhouette of this dancer is well captured, erroneous depth yield wrong occlusions in small regions around her arms (b).

moves across the scene. While our algorithm does not capture this in-plane rotation, the user can reproduce it by adding keyframes (Fig 9c). Orientation tracking might be improved by extending our user-guided point tracking approach to track multiple nearby points simultaneously. Our algorithm also tends to lose track of objects when they are too thin, or when they become too occluded. Prior methods proposed to use *skip edges* to deal with occlusions as part of the shortest path optimization [Su et al. 2018], but this mechanism increases the complexity of the graph significantly. In our experience, most issues caused by occlusion can be resolved by adding a few keyframes. Better tracking of thin objects could be achieved by increasing the resolution of the graph, potentially relying on a multi-scale strategy to keep the problem tractable, as has been suggested by Bian et al. [2022]. While we focused on general point tracking for maximum flexibility, integrating domain-specific algorithms (e.g. body pose tracking [Güler et al. 2018], or hand trajectory prediction [Liu et al. 2022b]) within our keyframe-based interface could improve robustness for specific use cases.

While we provide context in the drawing panel by rectifying the underlying video with a homography, strong distortion appears when the canvas is too foreshortened (Fig. 10a). Future work could explore the use of more advanced novel-view-synthesis techniques, possibly by leveraging the depth maps and multiple views of the scene we have access to.

Finally, our system relies on the depth map provided by *Robust Consistent Video Depth Estimation* to render occlusions automatically. While this solution often suffices for doodles made of sparse strokes, errors around the silhouette of the occluder can appear on densely painted doodles (Fig. 10b). Users can fall-back to existing masking tools to handle such cases, although a more integrated solution would consist in providing sparse user corrections to the depth estimation algorithm to refine its result.

7 CONCLUSION

Recent progress in computer vision and recording devices makes 3D reconstruction readily available from casually-captured videos,

a trend that is likely to grow with the democratization of depth sensors and on-camera SLAM systems. In this paper, we showed how depth and motion information can be leveraged to ease the creation of *video doodles*, a popular media that mixes videos with hand-drawn animations. Depth and estimated cameras allow us to embed hand-drawn doodles in 3D with correct perspective and occlusion effects, while our novel controllable tracking method allows us to make the doodles move along with objects in the scene. A user study and resulting artifacts demonstrate the effectiveness of our interface and our keyframe-based tracking algorithm to create a wide-range of 3D effects. We believe that additional interactions between real and hand-drawn content could be investigated in the future, for instance to enrich the doodles with secondary motion [Willett et al. 2017] or particle effects [Kazi et al. 2014] triggered by the objects they track.

ACKNOWLEDGMENTS

We thank our study participants for their time and effort, Wilmot Li for inspiration all along the project, Pascal Tempier for infrastructure support, Nicolas Rosset for help with testing the user interface, Felix Hähnlein, Timothée De Graeve, Rubaiat Habib and his daughters for appearing in or providing input footage. We thank the content creators of all tutorials cited in Supplemental Materials for sharing valuable information on how to make video doodles, and we thank LeopARTnik, Joseph Diaz and @murtzell for inspiring us and permitting usage of their creations as examples in the accompanying video. We thank the anonymous reviewers for their helpful comments. This work was supported by ERC Starting Grant D3 (ERC-2016-STG 714221), and by software and research donations from Adobe.

REFERENCES

- Adobe. 2022. After Effects. <https://www.adobe.com/products/aftereffects.html>.
- Aseem Agarwala, Aaron Hertzmann, David H Salesin, and Steven M Seitz. 2004. Keyframe-based tracking for rotoscoping and animation. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 23, 3 (2004).
- Brian Amberg and Thomas Vetter. 2011. GraphTrack: Fast and globally optimal tracking in videos. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Apple. 2022. ARKit. <https://developer.apple.com/augmented-reality/arkit/>.
- Rahul Arora, Rubaiat Habib Kazi, Tovi Grossman, George Fitzmaurice, and Karan Singh. 2018. SymbiosisSketch: Combining 2D & 3D Sketching for Designing Detailed 3D Objects in Situ. In *Proc. ACM SIGCHI Conference on Human Factors in Computing Systems*.
- Seok-Hyung Bae, Ravin Balakrishnan, and Karan Singh. 2008. ILoveSketch: as-natural-as-possible sketching system for creating 3d curve models. In *ACM Symposium on User Interface Software and Technology (UIST)*.
- Zhangxing Bian, Allan Jabri, Alexei A. Efros, and Andrew Owens. 2022. Learning Pixel Trajectories with Multiscale Contrastive Random Walks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Blender. 2022a. Blender Grease Pencil. <https://www.blender.org/features/story-artist/>.
- Blender. 2022b. Blender Motion Tracking. https://docs.blender.org/manual/en/latest/movie_clip/tracking/index.html.
- BorisFX. 2022. Mocha Pro. <https://borisfx.com/products/mocha-pro/>.
- Nicolas Boumal. 2013. Interpolation and regression of rotation matrices. In *International Conference on Geometric Science of Information*. Springer, 345–352.
- Aeron Buchanan and Andrew Fitzgibbon. 2006. Interactive feature tracking using kd trees and dynamic programming. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Mental Canvas. 2022. Mental Canvas Application. <https://mentalcanvas.com/>.
- Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. 2019. OpenPose: Real-time Multi-Person 2D Pose Estimation using Part Affinity Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019).

- Carl Doersch, Ankush Gupta, Larisa Markeeva, Adria Recasens Contente, Kucas Smaira, Yusuf Aytar, Joao Carreira, Andrew Zisserman, and Yi Yang. 2022. TAP-Vid: A Benchmark for Tracking Any Point in a Video. In *NeurIPS Datasets Track*.
- Julie Dorsey, Songhua Xu, Gabe Smedresman, Holly Rushmeier, and Leonard McMillan. 2007. The mental canvas: A tool for conceptual architectural design and analysis. In *Pacific Conference on Computer Graphics and Applications*.
- Pierre Dragicevic, Gonzalo Ramos, Jacobo Bibliowicz, Derek Nowrouzezahrai, Ravin Balakrishnan, and Karan Singh. 2008. Video browsing by direct manipulation. In *Proc. ACM SIGCHI Conference on Human Factors in Computing Systems*.
- Ruofei Du, Eric Turner, Maksym Dzitsiuk, Luca Prasso, Ivo Duarte, Jason Dourgarian, Joao Afonso, Jose Pascoal, Josh Gladstone, Nuno Cruces, et al. 2020. DepthLab: Real-time 3D interaction with depth maps for mobile augmented reality. In *Proc. ACM Symposium on User Interface Software and Technology (UIST)*.
- Foundry. 2022. Nuke. <https://www.foundry.com/products/nuke-family/nuke>.
- Dan B Goldman, Brian Curless, David Salesin, and Steven M Seitz. 2006. Schematic storyboarding for video visualization and editing. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 25, 3 (2006).
- Dan B Goldman, Chris Gonterman, Brian Curless, David Salesin, and Steven M Seitz. 2008. Video object annotation, navigation, and composition. In *Proc. ACM symposium on User Interface Software and Technology (UIST)*. 3–12.
- Rıza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. 2018. Densepose: Dense human pose estimation in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7297–7306.
- João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. 2014. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37, 3 (2014), 583–596.
- Eric Horvitz. 1999. Principles of mixed-initiative user interfaces. In *Proc. ACM SIGCHI conference on Human Factors in Computing Systems*.
- Riwano Ikeda and Issei Fujishiro. 2021. SpiCa: Stereoscopic Effect Design with 3D Pottery Wheel-Type Transparent Canvas. In *ACM SIGGRAPH Asia 2021 Technical Communications*.
- Allan Jabri, Andrew Owens, and Alexei A Efros. 2020. Space-Time Correspondence as a Contrastive Random Walk. *Advances in Neural Information Processing Systems* (2020).
- Yoni Kasten, Dolev Ofri, Oliver Wang, and Tali Dekel. 2021. Layered neural atlases for consistent video editing. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)* 40, 6 (2021), 1–12.
- Rubaiat Habib Kazi, Fanny Chevalier, Tovi Grossman, Shengdong Zhao, and George Fitzmaurice. 2014. Draco: Bringing Life to Illustrations with Kinetic Textures. In *Proc. ACM SIGCHI Conference on Human Factors in Computing Systems*.
- KenTools. 2022. GeoTracker. <https://keentools.io/products/geotracker-for-after-effects>.
- Felix Klose, Oliver Wang, Jean-Charles Bazin, Marcus Magnor, and Alexander Sorkine-Hornung. 2015. Sampling based scene-space video processing. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 34, 4 (2015), 1–11.
- Johannes Kopf, Michael F. Cohen, and Richard Szeliski. 2014. First-person Hyper-lapse videos. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 33, 4 (2014).
- Johannes Kopf, Kevin Matzen, Suhil Alsisan, Ocean Quigley, Francis Ge, Yangming Chong, Josh Patterson, Jan-Michael Frahm, Shu Wu, Matthew Yu, et al. 2020. One shot 3D photography. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 39, 4 (2020).
- Johannes Kopf, Xuejian Rong, and Jia-Bin Huang. 2021. Robust Consistent Video Depth Estimation. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Germán Leiva, Cuong Nguyen, Rubaiat Habib Kazi, and Paul Asente. 2020. Pronto: Rapid augmented reality video prototyping using sketches and enaction. In *Proc. ACM SIGCHI Conference on Human Factors in Computing Systems*. 1–13.
- Wenbin Li, Fabio Viola, Jonathan Starck, Gabriel J Brostow, and Neill DF Campbell. 2016. Roto++ accelerating professional rotoscoping using shape manifolds. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 35, 4 (2016).
- Yuwei Li, Xi Luo, Youyi Zheng, Pengfei Xu, and Hongbo Fu. 2017. SweepCanvas: Sketch-based 3D prototyping on an RGB-D image. In *Proc. ACM Symposium on User Interface Software and Technology (UIST)*. 387–399.
- Pengpeng Liang, Yifan Wu, Hu Lu, Liming Wang, Chunyuan Liao, and Haibin Ling. 2018. Planar object tracking in the wild: A benchmark. In *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE.
- Jian Liao, Adnan Karim, Shivesh Singh Jadon, Rubaiat Habib Kazi, and Ryo Suzuki. 2022. RealityTalk: Real-Time Speech-Driven Augmented Presentation for AR Live Storytelling. In *Proc. ACM Symposium on User Interface Software and Technology (UIST)*. Article 17, 12 pages.
- Jingyuan Liu, Hongbo Fu, and Chiew-Lan Tai. 2020. Posetween: Pose-driven tween animation. In *Proc. ACM Symposium on User Interface Software and Technology (UIST)*.
- Shaowei Liu, Subarna Tripathi, Somdeb Majumdar, and Xiaolong Wang. 2022b. Joint hand motion and interaction hotspots prediction from egocentric videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3282–3292.
- Sean J Liu, Maneesh Agrawala, Stephen DiVerdi, and Aaron Hertzmann. 2022a. ZoomShop: Depth-Aware Editing of Photographic Composition. In *Computer Graphics Forum*, Vol. 41. Wiley Online Library, 57–70.
- Xuan Luo, Jia-Bin Huang, Richard Szeliski, Kevin Matzen, and Johannes Kopf. 2020. Consistent video depth estimation. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 39, 4 (2020), 71–1.
- Maximilian Mayer, Philipp Trenz, Sebastian Pasewaldt, Mandy Klingbeil, Jürgen Döllner, Matthias Trapp, and Amir Semmo. 2021. MotionViz: Artistic Visualization of Human Motion on Mobile Devices. In *ACM SIGGRAPH 2021 Appy Hour*.
- Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. 2015. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics* 31, 5 (2015).
- Cuong Nguyen, Yuzhen Niu, and Feng Liu. 2013. Direct Manipulation Video Navigation in 3D. In *Proc. ACM SIGCHI Conference on Human Factors in Computing Systems*.
- Seoung Wug Oh, Joon-Young Lee, Kalyan Sunkavalli, and Seon Joo Kim. 2018. Fast video object segmentation by reference-guided mask propagation. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. 2016. A Benchmark Dataset and Evaluation Methodology for Video Object Segmentation. In *Computer Vision and Pattern Recognition*.
- Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alexander Sorkine-Hornung, and Luc Van Gool. 2017. The 2017 DAVIS Challenge on Video Object Segmentation. *arXiv:1704.00675* (2017).
- Alex Rav-Acha, Pushmeet Kohli, Carsten Rother, and Andrew Fitzgibbon. 2008. Unwrap mosaics: A new representation for video editing. *ACM Transactions on Graphics (Proc. SIGGRAPH)* (2008).
- Runway. 2022. RunwayML. <https://app.runwayml.com/>.
- Nazmus Saquib, Rubaiat Habib Kazi, Li-Yi Wei, and Wilmot Li. 2019. Interactive Body-Driven Graphics for Augmented Video Performance. In *Proc. ACM CHI Conference on Human Factors in Computing Systems*.
- Ryan Schmidt, Azam Khan, Gord Kurtenbach, and Karan Singh. 2009. On Expert Performance in 3D Curve-Drawing Tasks. In *Proc. Symposium on Sketch-Based Interfaces and Modeling (SBIM)*.
- Johannes Lutz Schönberger and Jan-Michael Frahm. 2016. Structure-from-Motion Revisited. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Noah Snavely, C Lawrence Zitnick, Sing Bing Kang, and Michael Cohen. 2006. Stylizing 2.5-D video. In *Proc. Symposium on Non-Photorealistic Animation and Rendering*.
- Tibor Stanko, Stefanie Hahmann, Georges-Pierre Bonneau, and Nathalie Saguin-Sprynski. 2017. Shape from sensors: Curve networks on surfaces from 3D orientations. *Computers & Graphics (Proc. SMI)* 66 (2017).
- Qingkun Su, Xue Bai, Hongbo Fu, Chiew-Lan Tai, and Jue Wang. 2018. Live sketch: Video-driven dynamic deformation of static drawings. In *Proc. ACM SIGCHI Conference on Human Factors in Computing Systems*. 1–12.
- Ryo Suzuki, Rubaiat Habib Kazi, Li-yi Wei, Stephen DiVerdi, Wilmot Li, and Daniel Leithinger. 2020. RealitySketch: Embedding Responsive Graphics and Visualizations in AR through Dynamic Sketching. In *Proc. ACM Symposium on User Interface Software and Technology (UIST)*.
- Zachary Teed and Jia Deng. 2020. RAFT: Recurrent all-pairs field transforms for optical flow. In *European Conference on Computer Vision (ECCV)*. 402–419.
- James Townsend, Niklas Koep, and Sebastian Weichwald. 2016. Pymanopt: A Python Toolbox for Optimization on Manifolds using Automatic Differentiation. *Journal of Machine Learning Research* 17, 137 (2016), 1–5.
- Julien Valentin, Adarsh Kowdle, Jonathan T Barron, Neal Wadhwa, Max Dzitsiuk, Michael Schoenberger, Vivek Verma, Ambrus Csaszar, Eric Turner, Ivan Dryanovski, et al. 2018. Depth from motion for smartphone AR. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)* 37, 6 (2018), 1–19.
- Nora S. Willett, Wilmot Li, Jovan Popovic, Floraine Berthouzoz, and Adam Finkelstein. 2017. Secondary Motion for Performed 2D Animation. In *Proc. ACM Symposium on User Interface Software and Technology (UIST)*.
- Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. 2018. PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes. *Robotics: Science and Systems (RSS)* (2018).
- Xiuming Zhang, Tali Dekel, Tianfan Xue, Andrew Owens, Qiurui He, Jiajun Wu, Stefanie Mueller, and William T Freeman. 2018. Mosculp: Interactive visualization of shape and time. In *Proc. ACM Symposium on User Interface Software and Technology (UIST)*.
- Zhoutong Zhang, Forrester Cole, Richard Tucker, William T Freeman, and Tali Dekel. 2021. Consistent depth of moving objects in video. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 40, 4 (2021), 1–12.



Fig. 11. Video doodles created with our system by participants of our user study. Participants created static doodles (text in P1 - hike, finishing line in P7 - motorbike race) as well as dynamic doodles (P3 - angel & devil child, P5 - squirrel friend). Several of these doodles get occluded by real objects (poles in P4 - car race, face on the tree in P5 - squirrel friend), and are synchronized with specific video events (yellow sparkles when P2's dancer touches the ground, water splash when P6's horse ends its jump).



Fig. 12. Additional video doodles created with our system, including sport instructions (Climbing, Tennis), annotations and highlights (Travel vlog, Climbing jump), and fun cartoons (Flamingo, Comics parkour). Please see supplemental materials for the corresponding videos.