Match-free Inbetweening Assistant (MIBA): A Practical Animation Tool without User Stroke Correspondence

Shuhong Chen[1] and Matthias Zwicker[1]

University of Maryland - College Park, College Park, MD, USA
{shuhong,zwicker}@umd.edu

Abstract. In traditional 2D frame-by-frame animation, inbetweening (interpolating line drawings, abbr. "IB") is still a manual and labor-intensive task. Despite the abundance of literature and software offering automation and claiming speedups, animators and the industry as a whole have been hesitant to adopt these new tools. Upon inspection, we find prior work often unreasonably expects adoption of novel stroke-matching workflows, naively assumes access to adequate center-line vectorization, and lacks rigorous evaluation with professional users on real production data. Facing these challenges, we leverage optical flow estimation and differentiable vector graphics to design a "Match-free Inbetweening Assistant" (MIBA). Unrestricted by the need for user stroke correspondence, MIBA integrates into the existing IB workflow without introducing additional requirements, and makes the raster input case feasible thanks to its robustness to vectorization quality. MIBA's simplicity and effectiveness is demonstrated in our comprehensive user study, where users with professional IB experience achieved 4.2x average speedup and better chamfer distance scores on real-world production data, given only a 5-minute tutorial of new functionality.

Keywords: 2D animation · in-betweening

1 Introduction

In traditional 2D frame-by-frame animation, animators often draw "pose-to-pose": first completing "cleaned-up keyframes" (or "CU") at several critical poses of a sequence, before spatially interpolating them with "inbetweens" (or "IB") at intervals specified by a "timing grid" (Fig. 1). CUIB (a.k.a. "douga" in Japanese) can be drawn as vectors or rasters, but they are typically saved as aliased rasters to then be bucket-filled by digital ink and paint staff (DIP). The IB process demands precise linework and is notoriously time-consuming, taking anywhere between 5-40 minutes per frame depending on the difficulty. Across a single 24-minute episode at 12 frames per second, thousands of such inbetweens are drawn by hand; this scales to tens of thousands of drawings for seasonal shows and feature films.

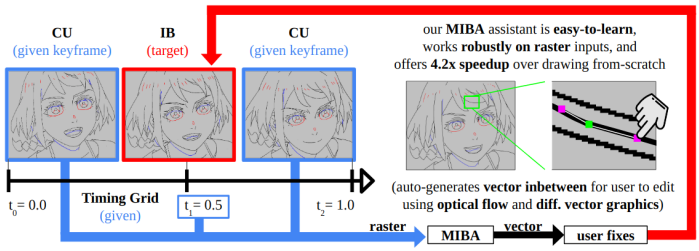

Fig. 1: Our MIBA system assists the inbetweening (IB) task. In traditinoal 2D frame-by-frame animation, animators often draw "pose-to-pose", finishing **"cleaned-up keyframes" (CU)** at critical poses of a sequence first, before completing the **"inbetweens" (IB)** that spatio-temporally interpolate based on a specified **"timing grid"**. IB is notoriously labor-intensive and is still drawn manually in many productions, since existing automatic methods often fail on raster inputs and are incompatible with artist workflows. Leveraging state-of-the-art optical flow [14] and differentiable vector graphics [9], our MIBA system works **robustly on scanned-in raster** inputs, and integrates into the existing workflow so well that it can be **learned in 5 minutes** by experienced animators in the industry. Tested by professional inbetweeners on real production data, **MIBA sped up IB drawing by 4.2x** on average in our user study. © Studio NUT

The academic community has proposed many systems for offering computational IB assistance, typically in the form of matching vector strokes across keyframes to interpolate control points [3,5,7,11,16–19]. But despite the abundance of work, there is still little adoption of automatic methods in industry. While there are many artistic, economic, and cultural factors contributing to this lack of progress, we observe that the research community has repeatedly overlooked major design considerations that have significant impact on practical usability and system evaluation. Specifically:

Consideration 1: Artists should not be expected to alter their workflow. It is prohibitively expensive for artists to make major changes to their established way of doing things, and invest time and effort into learning the intricacies of new tooling. Yet, the systems proposed in the literature consistently introduce additional requirements and functions that burden the animator (see summary in Tab. 1). By learning the existing workflow from professionals, we designed MIBA to automate specific steps without intrusively adding new ones (Fig. 2).

Consideration 2: Animators do not have vector keyframe inputs adequate for prior vector-based methods. Prior work has mostly focused on interpolation between matched vector strokes, but implicitly assumes properly-connected, noise-free, and/or nearly-identical vector topology on both input keyframes (Tab. 1). Unfortunately, we find that these requirements are satisfied by neither off-the-shelf vectorizations nor by artist-drawn vectors (Fig. 4, Fig. 6). MIBA on the other hand is robust to input vectorization quality, and so can feasibly handle the raster input case.

Consideration 3: Evaluation should be precise and representative of real-world animation production. Relevant IB user studies are few; most have participants without professional IB experience, tend to evaluate qualitatively on simple animations, lack details about procedures and training, and neglect to report key metrics like speedup and evaluation against ground truth (see summary in Tab. 3). We evaluate MIBA with a comprehensive user study addressing all these points.

Taking into account these three key considerations, we make the following contributions:

- **MIBA**, a "Match-free Inbetweening Assistant" leveraging deep optical flow and differentiable vector graphics [9, 14]. Unrestricted by the need for user stroke correspondence, MIBA seamlessly integrates into the existing IB workflow, and works well on raster input thanks to its robustness to vectorization quality.
- A **comprehensive user study** evaluating our MIBA system, in which users with professional IB experience achieved both 4.2x average task speedup and better chamfer distance scores w.r.t. ground-truth on real-world production data, given only a 5-minute tutorial of MIBA's functionality.

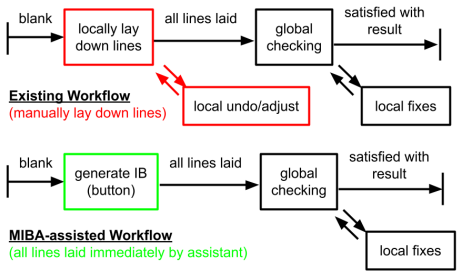

Fig. 2: MIBA non-intrusively assists the existing workflow. Our proposed system (bottom) seamlessly integrates into the existing workflow as described by professionals (top). Laying down all predicted lines with a simple click advances users to the "check+fix" stage of their familiar workflow, characterized by zooming globally/locally to find/fix errors and missing lines. By replacing the time-consuming step of zooming in to lay lines, we save significant effort without deviating from established practices.

2 Related Work

Prior vector-based inbetweening works first and foremost address the problem of vector stroke correspondence [3, 16, 19]. They assume two vector keyframe inputs with extremely similar and well-connected topologies, between which they match strokes. Based on the discovered 1-to-1 correspondence, control points are

	Additional requirements / tools to learn
MIBA (ours)	- **none** (raster input, no vector requirements) (new assistant operated by button click)
BetweenIT	- requires adequate vector input for 1-to-1 stroke correspondence - lasso tool for stroke correspondence - point tool for vector correspondence - path tool for trajectory guidelines
VGC	- requires learning completely new time-vector topology data primitives
DiLight	- requires well-connected vectors - path tool for stroke correspondence - must set stroke matching tolerance param.
FTP-SC	- requires adequate vector input for 1-to-1 stroke correspondence - matching tool for stroke correspondence
Narita et. al.	- cannot generate vector output, only rasters are supported
CACANI	- requires adequate vector input for 1-to-1 stroke correspondence - requires stroke depth layering - requires stroke occlusion orientation - new grouping, linking, inverting tools
AnimeInbet	- requires strictly straight-line vectors (does not support curved lines) - requires well-connected and densely-sampled vectors

Table 1: Comparison of new requirements and tooling in prior work. It is costly for animators to make major changes to their established workflows, and invest time and effort into learning new tooling. Yet, systems proposed in academia consistently introduce additional requirements and functions that burden the animator. Our MIBA framework on the other hand is designed to accelerate IB with a simple button click. [3,5,7,11,13,16,19]

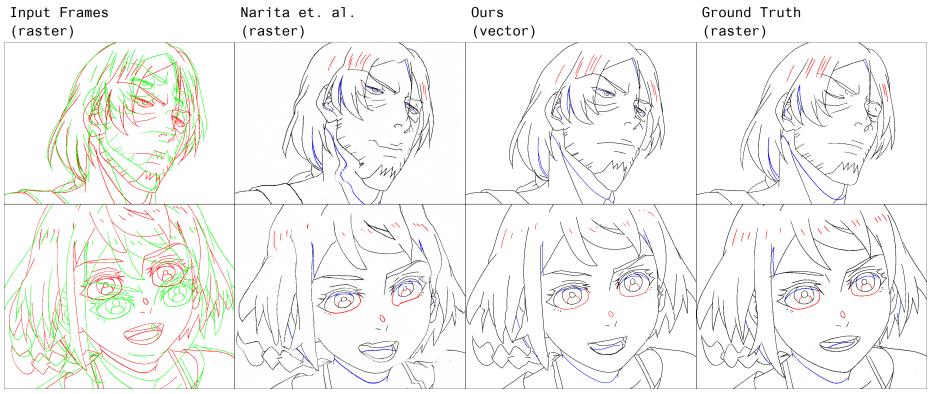

Fig. 3: Limitations of raster output. While allowing raster output removes concerns of vector topology [11], we lose vector editability, which is critical to the "check+fix" stage of the existing IB workflow (Fig. 2). Raster outputs like the ones generated by Narita et. al. are often plagued with artifacts and poor line quality, which the artist must completely erase and redraw, even for minor imperfections. Our MIBA system addresses the vector topology concerns directly without resorting to less editable rasters, and often performs even better than raster warping due to our additional alignment optimizations (Sec. 3.2). © Studio NUT

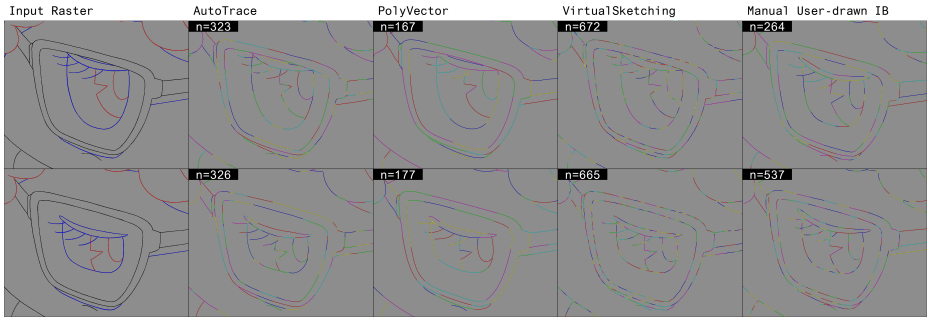

Fig. 4: Challenges of 1-to-1 stroke correspondence. Prior work [16, 19] unreasonably expects vectorizers and users to produce 1-to-1 topology for their IB systems to function. Even for somewhat similar frames (top vs. bottom rows), off-the-shelf vectorizers [2, 10, 15] will return incompatible topologies with highly variant stroke count. In the last column, we show the two corresponding IBs manually drawn by a user in our study; this shows that topology correspondence is not considered in the existing manual workflow. Our MIBA system is free of vector-vector correspondence assumptions, and thus does not require new curve-matching tools for animators to operate. © Studio NUT

interpolated to derive the target IB vector representation. BetweenIT [16] and FTP-SC [19] provide a number of semi-automatic user tools to achieve this exact vector match, while DiLight [3] proposes guideline tools and loosens the strict correspondence requirement. More recently, AnimeInbet [13] proposed a method of fusing the two graph topologies. However, we often see that neither off-the-shelf vectorizers [2, 10, 15] nor users provide adequate vectorization quality for these methods (Fig. 4 & Fig. 6). In order to break free of the limitations imposed by vectorization, our match-free methodology completely discards the need to correspond disparate vectors, achieving a simple-to-use tool robust enough to work on vectorizations of raster scan-ins.

Stroke occlusion resolution and aesthetic non-linear curve interpolation are other aspects of IB work. CACANI [7] for example proposes a layered and oriented representation of strokes, and is able to infer occlusions automatically. BetweenIT [16], FTP-SC [19], and DiLight [3] all propose their own methods for allowing user-specified non-linear trajectories. In our work on MIBA, however, we have found that simple linear interpolation works well for real-world production IB data, and that improper occlusion is relatively quick for users to fix. We thus focus on removing the stroke correspondence paradigm; however, as the occlusion and interpolation techniques are orthogonal to our match-free contribution, they can be added after our workflow if desired.

Another very different approach to IB is taken by Narita et. al. [11], who discard the vector representation altogether in favor of rasters. They propose the direct use of optical flow to warp between keyframes (similar to video frame interpolation systems commonly used for natural RGB videos [1, 4, 6]), and improve flow estimation on sparse line drawings with the distance transform. While

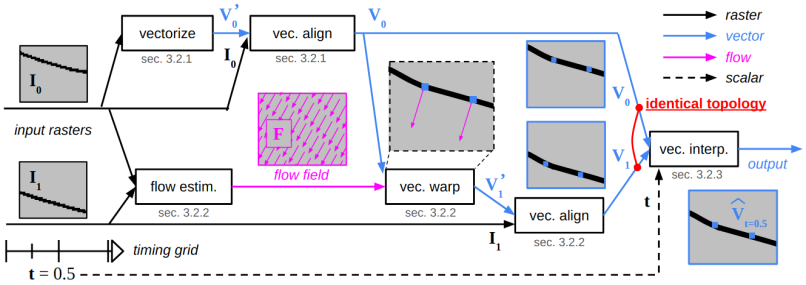

Fig. 5: Schematic of our proposed system. Given the left two raster keyframes (I_0, I_1) and the interpolating position on the timing grid ($t = 0.5$), our system produces a vector inbetween ($\hat{V}_t$, right) that can then be adjusted as needed, without requiring the user to specify stroke correspondences between vectorized versions of the input frames. The key to achieving this match-free framework lies in our ability to robustly warp and align the vectorization of the first frame (V_0, top) to a raster of the second frame (I_1, bottom). This way, we obtain topologically identical vectors (red) aligned to respective frames, ready for interpolation. The system leverages optical flow estimation and differentiable vector graphics to produce reasonable stroke-raster alignment.

this approach indeed relieves the user of vector considerations, the results are heavily dependent on optical flow performance. As failure cases are common for animations with large displacements and sparse lines, the user would need to edit an inflexible raster representation (Fig. 3). Our MIBA on the other hand tackles vector representation issues, without resorting to rasters.

Yet another alternative is proposed by Boris et. al. [5], who introduce a novel "vector animation complex" data structure to manage interpolation of topologies across both space and time. However, the new representation is a departure from the conventional vector graphics paradigm, and inhibitively requires the artist to learn a new framework of thought in order to inbetween.

As summarized in Tab. 1 and illustrated in Fig. 2, our MIBA system distinguishes itself from prior work by not requiring significant changes to animator tooling or workflow. Additionally, we provide one of the most comprehensive user studies for IB in the literature to evaluate MIBA (Tab. 3); we report metrics with respect to ground-truth production data from the real world, provide explicit details on speed and interaction gains, and test with professional IB animators.

3 Methodology

Our system operates on a single timing grid sequence at a time (Fig. 1). As illustrated in Fig. 2, MIBA lays down predictions of where the vector lines of IB should be placed. Our system uses a match-free method to return a vector IB drawing for each interval specified by the timing grid. The user may then use common vector editing tools, or choose to rasterize at any time and use

common raster tools. Below, we define the input and output representations more formally, before describing our algorithm and provided user interaction tools.

3.1 Input/Output Representations

The inputs to the MIBA system are: a timing grid T, and two cleaned keyframes (Fig. 5, left-most side). The timing grid T is a sorted array of unique values $t_i \in [0, 1]$, where $t_0 = 0$ and $t_1 = 1$. The two CU keyframes are assumed to be rasters (I_0, I_1 in Fig. 5). In the case where a vector representation is available, we still rasterize before inputting to our system; this ensures consistency of outputs, and relieves any mental burden on the animators to consider line topology when drawing CU. Similar to the IB to be generated, the CU keyframes are binary-aliased for eventual digital coloring with paint-bucket tools. The CU often only have few colors aside from black; by default in the Japanese pipeline, red denotes highlights, blue is shadow, and green is a special effect or second shadow/highlight.

The output representation of MIBA is a vector graphics representation ($\hat{V}_t$ in Fig. 5), which can be rasterized if desired. We select to use cubic Bézier curves, with a polar representation for control points. The representation backend supports arbitrary graph connectivity, though for our experiments we only work with acyclic graphs, similar to SVGs. In practice, our system renders curves as short piecewise-linear line segments; we thus support variable line width across a single curve, although we found that a single global thickness worked well enough for our specific data.

3.2 Match-free Inbetweening

Our proposed match-free inbetweening method is illustrated in Fig. 5. Similar to previously proposed frameworks, each curve of the output is derived by interpolating between the vertex/handle coordinates of two existing curves (V_0 and V_1, one representing each vectorized input image). However, to fit this paradigm, prior work struggles at finding the two correct corresponding strokes to inbetween from each input, because the input vectorizations have either unreasonably different topology (Fig. 4) or inadequately noisy connectivity (Fig. 6).

The key insight of our MIBA method is that we can warp one frame's vector (V_0) to align with the other's raster (I_1). This way we do not need to match two disparate vectorizations, and instead can directly interpolate between two coordinate-value configurations of the same vector topology (Fig. 5, red). Put another way, we find offsets from one vector to look like the other raster; interpolation equates to moving by a specified fraction of those offsets.

Naturally, since no matching occurs between different vectorizations, MIBA is robust to the connectivity and noisiness of the input vectors; by discarding the stroke correspondence paradigm, we make IB assistance on vectorized scanned-ins a feasible reality (Fig. 6). In addition, animators can easily operate MIBA

without the intricate stroke correspondence and match guidance tooling required by prior work (Tab. 1).

From the perspective of solving the stroke occlusion problem, MIBA intrinsically resolves occlusion at the warp and align stages. At these steps, strokes are effectively occluded by shortening curves to the occlusion boundary, in order to satisfy alignment to the opposing raster frame. Note that this occurs independently of local topology at the junction, as the optimization condition is imposed directly on a raster render of the stroke graph. While we found that improper occlusions still may appear from imperfect alignments, animators are able to fix them relatively quickly.

Note that MIBA is asymmetric with respect to the input frame order; in other words, the system output for interpolating $t = 0.5$ will be different if the two input CU keyframes are swapped. Users can choose which direction to use MIBA, although in practice we find many users simply stick with the default forward direction.

The subsections below describe in more detail the steps outlined in Fig. 5, and how we leveraged state-of-the-art optical flow and differentiable vector graphics to achieve this non-trivial vector-raster alignment across frames.

Vectorization with Alignment Post-processing As previously mentioned, MIBA only works with a single vector topology that does not need to be matched; thus the method is robust to the input vector representation quality. In Fig. 6, we demonstrate our system with Weber AutoTrace [15], PolyVectorization [2] (simplified with [12]), and Virtual Sketching [10]; in all three cases, our system was able to deliver similar reasonable results. We default to using AutoTrace in our program for its quick processing speed (refer to suppl. for details).

Across the different vectorization algorithms, we inevitably found imperfections in the linework that did not match with the input raster; usually, this came in the form of "wobbly" curves. To improve the base vectorizations (V_0'), we optimize the vector image to match the input raster (I_0) using DiffVG [9] by minimizing

$$V_0 = \underset{\theta}{\operatorname{argmin}} L_2\Big(R(\theta),\ I_0\Big), \tag{1}$$

where θ represents the vertices and Bézier control points initialized at V_0', and R denotes the DiffVG differentiable vector rendering operation. The optimization is run for 32 iterations, with Adam [8] on MSE image loss with unit learning rate ($lr = 1.0$). To facilitate alignment of initially distant lines, we additionally apply Gaussian blur to the differentiable render before loss evaluation for the first half of optimization (with a ramping sigma).

Vector Warping & Alignment The goal of warping here is to roughly position the post-processed vectorization (V_0) over the other raster frame (I_1), to initialize the more expensive alignment optimizations. We estimate the optical flow (F)

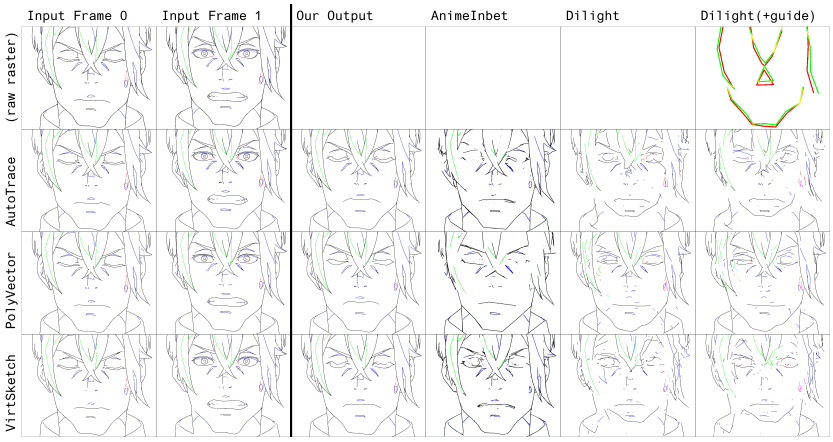

Fig. 6: Inadequate vectorization quality for prior work. Prior vector-based methods without strict 1-to-1 stroke matching requirements (like AnimeInbet [13] and Dilight [3]) are highly sensitive to the quality of input vectors. Both AnimeInbet and Dilight (rightmost columns) produce unusable artifact-ridden results, regardless of which off-the-shelf vectorizer is used to preprocess the raster input (left columns, AutoTrace [15], PolyVector [2], VirtualSketching [10]). This demonstrates the impracticality of prior methods on the very common raster input use case. On the other hand our robust MIBA system innately handles vectorization, achieving consistently reasonable results across different vectorizers. © Studio NUT

between the two raster inputs using an off-the-shelf RAFT model [**?**]; inspired by previous work on raster inbetweening of line art [11], we preprocess the sparse line drawings into dense distance transform images to improve the flow estimation.

To perform the warp, we simply offset each vertex by the flow sampled at its image coordinates, denoted as $V_1' = V_0 + F[V_0]$. Even without modifying the polar Bézier handle values, we found that the warp gave results that were reasonable, but not yet acceptable for interpolation without further alignment.

With the vectorization of the first frame roughly warped to the second raster, we remove remaining alignment imperfections by DiffVG optimization [9]. The optimization process is the same as vector postprocessing previously described in Sec. 3.2, but θ is initialized at V_1' and the raster target is instead I_1:

$$V_1 = \underset{\theta}{\operatorname{argmin}}\, L_2\Big(R(\theta),\ I_1\Big). \tag{2}$$

Note that there still may be imperfect alignments, since the inherent topology of the two frames is often different (Fig. 3, Fig. 4, Fig. 6). However, we find that in many cases these incompatibilities can be quickly fixed by the animator after interpolation, still at a time discount compared to manually laying down all the lines. Note also that optical flow warping is critical for avoiding degenerate cases with large displacements; please refer to the supplementary for more detailed explanation.

Vector Interpolation Despite the emphasis that prior work puts on interpolating aesthetically between two curves [3,16,19], we find that a simple linear interpolation of vertex coordinates and polar Bézier handle values is sufficient enough to satisfy professional IB animators. As the choice of interpolation method here is orthogonal to our contribution of match-free assistance, this step can be freely interchanged with interpolation schemes from other work. For timing grids with more than one IB, we simply cache the alignment offsets and lerp to new intervals as needed. Once the rasterization of the first frame is appropriately offset to the target IB, the user is free to correct any imperfections of the MIBA output using vanilla vector manipulation tools.

3.3 User Interaction

We implemented an interactive web browser app with Vue.js. Our app has basic features typical to modern IB software, including a navigable viewport, toolbar with options, raster and vector layers, frame/layer selection panels, onionskinning and frame-flipping, tool keybindings, undo/redo history, etc. Pen/stylus input is supported, and is functionally equivalent to the mouse.

MIBA assistance is implemented as buttons attached to each timing grid sequence the user is asked to IB. The user clicks on the provided "assist" button, which provides a preview of the generated IBs, and then clicks "use" on the previews they would like to use. This inserts the MIBA output as a vector layer on the frame in question, which the user is then free to edit or rasterize. As MIBA is asymmetric with respect to input frame order, each timing grid is equipped with two "assist" buttons, one for each direction; the user can elect to preview and use either as they please. In order to disable the assistant for certain parts of the user study, we simply remove the "assist" buttons from the interface; all other parts of the program work as a typical piece of 2D animation software.

Standard vector, raster, and layer editing tools are provided: selection, copy-paste, layer transformation, vector drawing (with line color, width, and curvature), vector manipulation (vertices and Bézier handles), stroke deletion, and raster brush (with color, width, and erase). We additionally provide an "unpeg" function; this allows users to temporarily shift and align other images in the sequence to visually reduce the gap between the lines to inbetween.

Note that the entirety of our proposed MIBA system can be operated with simple button clicks; as summarized in Tab. 1, this contrasts with prior work, in which additional requirements must be considered, workflows must be rearranged, and new intricate tooling must be learned. Aside from our straightforward MIBA buttons, all the other functions in our program as described above can be considered "vanilla" tooling for 2D animation software that digital animators are already familiar with. This simplicity reflects how MIBA integrates seamlessly with the existing manual workflow, without burdening the artist with an intrusive new framework.

usr. ID	sample-A1									sample-A2									third sample (time permitting)									
	time				interaction		chamfer distance			time				interaction		chamfer distance			samp.	time				interaction		chamfer distance		
	est.	man.	asst.	spd+	man.	asst.	gain	man.	asst.	gain	est.	man.	asst.	spd+	man.	asst.	gain	man.	asst.	gain	ID	est.	man.	asst.	spd+	man.	asst.	gain	man.	asst.	gain
1	5:00	13:50	4:50	**2.8x**	23	39	**-70%**	26.0	7.5	**71%**	5:00	12:27	3:40	**3.3x**	17	23	**-35%**	25.9	6.6	**74%**	B	30:00	56:14	6:50	**8.2x**	170	37	**78%**	20.1	16.0	**20%**
2	-	16:25	4:10	**3.9x**	142	63	**56%**	10.2	8.2	**20%**	-	11:24	4:15	**2.6x**	147	75	**49%**	9.4	7.3	**23%**	C	-	8:28	2:09	**3.9x**	108	19	**82%**	8.8	6.8	**23%**
3	10:00	16:54	7:50	**2.1x**	175	118	**33%**	14.0	7.1	**50%**	10:00	15:28	9:15	**1.6x**	103	143	**-39%**	11.2	6.3	**44%**	D	10:00	19:13	9:23	**2.0x**	160	124	**23%**	5.8	5.2	**11%**
4	10:00	18:33	1:16	**14.6x**	77	17	**78%**	10.1	8.1	**19%**	10:00	13:46	1:10	**11.8x**	68	2	**97%**	10.2	7.2	**29%**	E	20:00	45:38	9:12	**4.9x**	297	105	**65%**	6.8	6.7	**1%**
5	5:00	28:56	5:55	**4.8x**	372	114	**69%**	8.9	7.1	**20%**	5:00	29:21	9:19	**3.1x**	459	123	**73%**	8.4	6.5	**22%**	-	-	-	-	-	-	-	-	-	-
6	20:00	29:05	4:59	**5.8x**	210	81	**61%**	20.5	8.1	**61%**	20:00	24:07	2:05	**11.5x**	239	16	**93%**	18.4	6.4	**65%**	-	-	-	-	-	-	-	-	-	-
7	2:00	39:19	16:04	**2.4x**	210	163	**22%**	9.5	7.3	**23%**	2:00	29:10	5:42	**5.1x**	142	57	**60%**	7.3	7.5	**-3%**	-	-	-	-	-	-	-	-	-	-
8	20:00	40:49	16:15	**2.5x**	249	137	**45%**	21.2	8.1	**62%**	20:00	29:37	7:23	**4.0x**	279	97	**65%**	17.9	8.2	**54%**	-	-	-	-	-	-	-	-	-	-

Table 2: User study results. A total of N=8 users participated in our study, of which all but one have over 2 years of professional IB experience. Participants were asked to inbetween two frames between the keys of sample A (first two column groups), and a third sample if time permitted. We ask each participant to draw each IB twice, once manually and once using our MIBA output as a starting point (man. vs. asst.). We consistently observe not only the significant time speedups and reduction in required interactions, but also a reduction in chamfer distance w.r.t. the ground truth IB when using the assistant (positive delta is good); this is usually due to the assistant missing fewer lines than users. The user study demonstrates that MIBA can successfully reduce the workload and improve accuracy on IB tasks in real production situations.

	used prod. data	reports metrics w.r.t. GT	reports exact times & speedup	reports interact. eff. gain	N users	training time for new tools	user professional experience
MIBA (ours)	Y	Y	4.2x	45.3%	8	5 min	2+ years of IB (all except one user)
BetweenIT [16]	Y	-	-	83%	unclear	an afternoon	tonal department (different from IB)
VGC [5]	-	-	-	-	-	-	-
DiLight [3]	-	-	-	-	unclear	unclear	professional animators (unclear if IB)
FTP-SC [19]	Y	Y	-	93.6%	5	unclear	1 animator, 4 graduate students
Narita et. al. [11]	Y	Y	-	-	-	-	-
CACANI [7]	-	-	3.18x	-	-	-	-
AnimeInbet [13]	-	Y	-	-	36*	-	-

Table 3: Comparison to evaluation performed in prior work. Whereas prior work generally lacks critical details and measurements for their user studies, we provide comprehensive descriptions of the context and results of our study. Our speedup and gain are averages calculated from Tab. 2; note that the numerical reports are not directly comparable, as the interactions and task loads are different in other frameworks. Notice that unlike in other studies, professional IB animators were able to achieve improved results with MIBA after only a 5-minute tutorial of new features. (*) Note that AnimeInbet's user study consisted of only perceptual ranking, without any user-tool interaction. [3, 5, 7, 11, 13, 16, 19]

user	pref.*	IB xp.	check xp.	device	intu.	qual.
1	CSP	>2yrs	>2yrs	pen	5.0	4.0
2	Stylos	>2yrs	>2yrs	pen	5.0	5.0
3	CSP	<6mo	-	mouse	4.0	4.0
4	CSP	>2yrs	>2yrs	mouse	5.0	5.0
5	TB	>2yrs	>2yrs	pen	3.0	4.0
6	Flash	>2yrs	>2yrs	pen	2.5	3.0
7	CSP	>2yrs	-	pen	3.0	4.0
8	Moho	>2yrs	>2yrs	mouse	3.5	4.5

Table 4: User metadata and questionnaire. All but one user has over two years of professional IB experience, and six of eight have over two years of IB checking experience (a more senior supervisory role); for context, IB training can take between one to six months. Three participants used primarily the mouse, due to tablet unavailability, pen incompatibility, or by choice. While most were relatively satisfied with the assistant output quality (average 4.2 on 1-5 scale), the perception of our program's intuitiveness to use scored lower (average 3.9 on 1-5 scale). We believe the "intuitiveness" variation depends largely on our app's similarity to the artist's most comfortable software ("pref" column); this is a testament to the importance of fitting the user's existing workflow and tool preferences. (*) Abbreviations: Clip Studio Paint, RETAS Stylos, ToonBoom, Adobe Flash/Animate, Moho.

4 User Study Evaluation

4.1 Data & Participants

We evaluate our system on real production data provided by Japanese anime production studios; please refer to the supplementary file for a detailed description of the data. Note that many prior works neglect to evaluate on real-world production data (Tab. 3, first column). Previous methods evaluating real production data either strongly assumed adequate-quality vectorizations [16,19], or do not support vector output representations at all [11]; our method on the other hand is able to process raw raster CU keyframe inputs, and give vector outputs. Another distinguishing point of our data is that our lines are colored (i.e. shadow and highlight lines are present); while previous methods may have been extended to more than one line color, none have shown results on production data with this common characteristic.

All $N = 8$ participants recruited were professional animators with either industry (7) or freelance (1) experience (Tab. 4). All users had at least two years of IB experience, with the exception of one user with under 6 months experience. Six of the eight participants also had over two years of IB checking experience; for context, CUIB checking (a.k.a. "douga-kensa") is a supervisory task typically given to experienced CUIB/douga animators. For comparison, previous studies have often given vague descriptions of users' experience levels, recruited participants without direct IB experience, or even fail to mention the total number of users (Tab. 3).

4.2 Procedure

The study typically lasted two hours for each participant. Users were first brought to the tutorial page with the assistant disabled, and were given a 25-minute tutorial walking through the program basics (pen tablet setup, navigation, display options, and vector/raster tools excluding the proposed assistant). Three of eight participants used primarily the mouse, due to tablet unavailability, pen incompatibility, or by choice (Tab. 4).

After familiarizing with the vanilla tools, we enabled the MIBA assistant buttons, and gave a 5-minute explanation on how to use them. Studies in prior work often neglect to report the amount of time it took to teach participants the new tool (Tab. 3). However, we believe the training time is a key factor in determining whether animators can adopt the assistants. Tools with additional requirements (Tab. 1) take time to habituate, and discourage usage; our MIBA on the other hand is easy enough to explain in five minutes.

At this point, users have spent roughly 30 minutes getting to know the program features. Participants then moved on to timed tasks. For each sequence, we first asked users their time estimate for one of the IB frames in their most comfortable program. Users were then asked to draw the IBs manually (with the MIBA assistant disabled), and then to start over again using the assistant. All participants started with sample A (which had two IBs to complete, marked "A1" and "A2" in Tab. 2), and if time permitting were given a second sample from B-E (all with only one IB frame). Please refer to the suppl. for an additional discussion on the order of frames drawn.

At the end of the study, users were asked a brief questionnaire (Tab. 4) rating the program's intuitiveness to use (on a 1-to-5 scale), the quality of the assistant's predictions (also 1-to-5), and what additional features they would need to use the program for IB work.

4.3 Metrics

We make several measurements for each frame drawn by the users, and are mainly interested in comparing between manual and MIBA-assisted performance on a per-frame basis. While comparisons could be made across different users, there are many significant factors that cannot be controlled for, such as the similarity between our program and each user's most comfortable software.

In Tab. 2, we show in bold for each frame of each user: the time speedup from using the assistant, the effort gain in terms of tool interaction count (history length), and the reduction in chamfer distance with respect to the ground truth. The chamfer distance here is calculated as defined in prior work showing its relevance to perceptual line quality [4], and is measured between black-and-white binarized ground truth and an aliased rasterization of the user drawing. In addition to the drawing results, we include qualitative responses to the questionnaire at the end of the study in Tab. 4 (last two columns, on a 1-to-5 scale).

Note that prior studies repeatedly neglected to report the exact time speedup afforded by their proposed methods (Tab. 3). Some may extrapolate based on

the interaction effort gain, but the reduction in number of interactions does not map directly to time savings, nor does it allow generalized comparison between methods. Our results explicitly report the manual vs. assisted times, interactions, and chamfer distance broken down by frame and by user (Tab. 2); we believe this evaluation gives a much more complete picture of our tool's practicality.

5 Limitations and Future Work

A major limitation of our work is the computational cost. Each cached alignment for a sequence requires one minute of compute on a GTX 1080Ti; AutoTrace [15] vectorization takes around 2 seconds, both DiffVG vector alignments take 20-30 seconds, and all other steps are negligible (below one second each). While optical flow may be compute intensive, the GPU forward pass (even with distance transform [4]) is still less than one second. In future work, it is possible to improve the performance of vectorization and DiffVG.

Another limitation of our current system is the lack of trajectory control. Many prior works [3, 7, 16] have stressed the importance of supporting user-editable trajectories by non-linear arcs. However, in practice, we have found the majority of inbetweens in our datasets to be linear interpolations of CU keyframes. This may be due to differences between the western and eastern pipelines (our data comes from the latter). Note that most trajectory editing techniques in prior work can be implemented alongside MIBA, as they are orthogonal to our contribution of match-free assistance.

6 Conclusions

In conclusion, we present an assistant for inbetweening traditional 2D animation. We found that prior work often unreasonably expects artists to adopt new workflows, naively assumes access to adequate vectorization, and lacks evaluation with professional users on real production data. Facing these challenges, we leverage optical flow and differentiable vector graphics to design a "Match-free Inbetweening Assistant" (MIBA). Unrestricted by the need for stroke correspondence, MIBA integrates into the existing IB workflow without introducing additional requirements, and makes the raster input case feasible thanks to its robustness to vectorization quality. MIBA's simplicity and effectiveness is demonstrated in our user study, where users with professional IB experience achieved 4.2x average speedup and better chamfer scores on real-world production data, given only a 5-minute tutorial of new functionality.

Acknowledgements

The authors would like to thank Studio NUT for kindly providing the production data samples used in our user study and which appear in this paper and supplementary files, as well as all the animators who participated in our user study and shared their valuable domain expertise.

References

1. Bao, W., Lai, W.S., Ma, C., Zhang, X., Gao, Z., Yang, M.H.: Depth-aware video frame interpolation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3703–3712 (2019)
2. Bessmeltsev, M., Solomon, J.: Vectorization of line drawings via polyvector fields. ACM Transactions on Graphics (TOG) **38**(1), 1–12 (2019)
3. Carvalho, L., Marroquim, R., Brazil, E.V.: Dilight: Digital light table–inbetweening for 2d animations using guidelines. Computers & Graphics **65**, 31–44 (2017)
4. Chen, S., Zwicker, M.: Improving the perceptual quality of 2d animation interpolation. arXiv preprint arXiv:2111.12792 (2021)
5. Dalstein, B., Ronfard, R., Van De Panne, M.: Vector graphics animation with time-varying topology. ACM Transactions on Graphics (TOG) **34**(4), 1–12 (2015)
6. Huang, Z., Zhang, T., Heng, W., Shi, B., Zhou, S.: Rife: Real-time intermediate flow estimation for video frame interpolation. arXiv preprint arXiv:2011.06294 (2020)
7. Jiang, J., Seah, H.S., Liew, H.Z.: Stroke-based drawing and inbetweening with boundary strokes. In: Computer Graphics Forum. vol. 41, pp. 257–269. Wiley Online Library (2022)
8. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
9. Li, T.M., Lukáč, M., Gharbi, M., Ragan-Kelley, J.: Differentiable vector graphics rasterization for editing and learning. ACM Transactions on Graphics (TOG) **39**(6), 1–15 (2020)
10. Mo, H., Simo-Serra, E., Gao, C., Zou, C., Wang, R.: General virtual sketching framework for vector line art. ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2021) **40**(4), 51:1–51:14 (2021)
11. Narita, R., Hirakawa, K., Aizawa, K.: Optical flow based line drawing frame interpolation using distance transform to support inbetweenings. In: 2019 IEEE International Conference on Image Processing (ICIP). pp. 4200–4204. IEEE (2019)
12. Schneider, P.J.: An algorithm for automatically fitting digitized curves. Graphics gems **1**, 612–626 (1990)
13. Siyao, L., Gu, T., Xiao, W., Ding, H., Liu, Z., Loy, C.C.: Deep geometrized cartoon line inbetweening. ICCV (2023)
14. Teed, Z., Deng, J.: Raft: Recurrent all-pairs field transforms for optical flow. In: European conference on computer vision. pp. 402–419. Springer (2020)
15. Weber, M.: Autotrace - converts bitmap to vector graphics `https://autotrace.sourceforge.net/`
16. Whited, B., Noris, G., Simmons, M., Sumner, R.W., Gross, M., Rossignac, J.: Betweenit: An interactive tool for tight inbetweening. In: Computer Graphics Forum. vol. 29, pp. 605–614. Wiley Online Library (2010)
17. Yang, W.: Context-aware computer aided inbetweening. IEEE transactions on visualization and computer graphics **24**(2), 1049–1062 (2017)
18. Yang, W., Feng, J., Wang, X.: Structure preserving manipulation and interpolation for multi-element 2d shapes. Computer Graphics Forum **31** (2012)
19. Yang, W., Soon, S.H., Chen, Q., Liew, H.Z., Sýkora, D.: Ftp-sc: Fuzzy topology preserving stroke correspondence. Computer Graphics Forum **37** (2018)