A Video-based Interface for Hand-Driven Stop Motion Animation Production  
Xiaoguang Han  Hongbo Fu  Hanlin Zheng  Ligang Liu  Jue Wang

Abstract—Stop motion is a well-established animation technique, but its production is often laborious and requires craft skills. We present a new video-based interface which is capable of animating the vast majority of everyday objects in stop motion style in a more flexible and intuitive way. It allows animators to perform and capture motions continuously instead of breaking them into small increments and shooting one still picture per increment. More importantly, it permits direct hand manipulation without resorting to rigs, achieving more natural object control for beginners. The key component of our system is a two-phase keyframe-based capturing and processing workflow, assisted by computer vision techniques. We demonstrate that our system is efficient even for amateur animators to generate high quality stop motion animations of a wide variety of objects.

Index Terms—stop motion, animation, production, user interface

1 INTRODUCTION

Stop motion is a popular animation technique to make a physical object (e.g., a puppet) appear to move on its own as if by magic. Traditional stop motion production is a laborious process, requiring manually moving the target object in small increments and shooting one still picture of the object per increment, where hands or other tools that are driving the movement should be avoided. Since the object’s motion must be broken up into increments, animators have to concentrate on subtle object movement across neighboring frames rather than the desired continuous motion as a whole, making the production process not-intuitive especially for novice users.

Another limitation of the traditional stop motion workflow is that in many scenarios like animating a puppet flying in the air, simply holding the puppet with hands would cause large occlusion regions, which are difficult to recover even using the state-of-the-art image completion techniques. A common solution is to use a supporting rig that is attached to the puppet to prop the puppet securely. The rig can be later removed digitally in post-production, given tiny areas of caused occlusion. However, most professional rigs are expensive while homemade rigs demand the availability of suitable mechanical components (e.g., heavy weight plate, a long strand of wire, etc.) and some degree of manual dexterity in assembling and working with tools. Furthermore, attaching the rig to the object being animated could be tricky, especially when the object is too heavy, too fragile, or too smooth to attach onto.

To address these limitations, we present a video-based stop motion production system. Our key contribution is a novel two-phase stop motion capturing workflow (Figure 2), in which an animator performs a desired object motion twice. In the first phase, the animator holds the object using his/her hand(s) directly and performs the planned motion in a continuous fashion. In the second phase, the animator performs a similar motion, with his/her hand(s) holding at different positions of the object. Powered by computer vision techniques, our system can align images captured in the two phases and combine them together to create a composite where the hands are removed and occluded object parts are recovered. Such pipeline not only allows the animator to focus on the desired global motion without breaking it down to small increments, but also permits the replacement of rigs with hands, achieving more flexible and intuitive control of object motion.

In practice it is difficult to perform exactly the same motion twice, and our system does not require the animator to do that. Instead, for the second phase, we employ a keyframe-based capturing and processing approach using the first video as reference. Specifically, given the video captured in the first phase, our system automatically analyzes the object motion and presents the user a series of keyframes. In the second phase, for each keyframe, the user is required to hold the object in front of the camera so that both its position and orientation match with the given keyframe. Employing computer vision techniques, our system automatically combines the two images together to remove the hands in the keyframe. Finally, temporal interpolation is performed between keyframes to remove hands from intermediate frames as well, resulting in a complete stop-motion animation video (see an example in Figure 1). An efficient user interface (UI) is developed to facilitate this capturing and hand removal process, as described in Section 3.

Our work is complementary to, and can be used together with the existing stop motion techniques (Figure 1). Since our system requires the alignment and composition of the same object captured at different times, it is more suitable for

• X. Han and H. Fu are with the School of Creative Media, the City University of Hong Kong. E-mail: hanxiaoguangcagd@gmail.com, hongbofu@cityu.edu.hk
• H. Zheng is with the Department of Mathematics, Zhejiang University. This work was done when H. Zheng was a Research Associate at CityU HK. E-mail: zhenghanlin.zju@gmail.com
• L. Liu is with the School of Mathematical Sciences, the University of Science and Technology of China. E-mail: ligang.liu@gmail.com
• J. Wang is with Adobe Research. E-mail: juewang@adobe.com
objects with near-rigid parts (not requiring the whole object to be rigid). We demonstrate that our technique is efficient for animating the vast majority of objects in our everyday surroundings (see animations in the accompanying video). Furthermore, we show that even novice animators are able to use our system to turn a story idea into a high-quality stop motion animation.

2 Related Work

Stop motion production. Stop motion is a well-established animation technique. There exist various software systems dedicated to stop motion production, designed either for professionals (e.g., DragonFrame) or for amateur animators (e.g., Stop Motion Pro, iStopMotion). However, the underlying technology has advanced very little since day one. Previous systems rely on the constant stopping and starting of the camera to allow for slight object movement between two consecutive shots. Although our video-based interface seldom stops the camera, we still regard it as a stop motion production technique given its goal of generating stop-motion-style output. In traditional stop motion production, puppets with highly articulated armatures or clay figures are often used due to their ease of repositioning. However, their making requires sculpting and specialty skills such as brazing. In contrast, we focus on the animation of everyday physical objects, enabling easy production of stop motion animation even for beginners.

Video-based animation. Many systems have been proposed to create a non-photorealistic or cartoon animation from a video sequence, by stylizing the appearance of the video Stop motion, on the other hand, requires maintaining the original appearance and motion of the object while removing the user’s hands in the final animation. Barnes et al. [1] present a video-based interface for 2D cutout animation (one type of stop motion). Their system tracks 2D paper puppets being physically moved by an animator, and renders them using their corresponding images in the puppet database to remove the animator’s hands. However such systems can only handle planar objects without 3D rotation or self-occlusion, due to the limited number of images in the puppet database. Our system does not use such an example-driven approach, and thus can handle more complicated 3D shape and motion, as well as the interaction between the object and its surrounding environment (e.g., dynamic specular reflection in Figure 1). Recently Held et al. [2] present a Kinect-based system for producing 3D animations using physical objects as input. However, their output is computer-generated animations, rendered using the reconstructed 3D models of the physical objects, while our intention is to create animations with videos and pictures of real, physical objects.

Video segmentation and completion. Our system needs to segment out parts of the object occluded by hands, and then faithfully complete the occluded areas. There exist a variety of interactive rotoscoping [3] and video object cutout methods [4]. These methods are designed for accurate object segmentation and generally require a large amount of user interaction for achieving high accuracy. Given that our final goal is the reconstruction of the parts occluded by hands, a rough segmentation is sufficient in our case (Figure 6), which significantly simplifies the problem and reduces the amount of user intervention.

Many video completion methods have been proposed recently (see [5] and references therein). These approaches typically use 3D patches as primitives for maintaining spatial-temporal coherence, and operate by filling in a given missing region with content from the available parts of the same input video. Using 3D patches is effective on filling holes in the background region, but is less applicable for dynamic foreground objects whose motion is complex and topology could change from frame to frame. Furthermore, the data needed for completion may not exist in other parts of the video. Our keyframe-based recapturing and completion method avoids these problems and significantly eases the difficulty of video object completion.

Image completion and enhancement using multiple inputs. Our work heavily relies on the success of data-driven image completion, which is able to complete large holes in an image by borrowing suitable patches from other images. Agarwala et al. [6] present a digital photomontage framework for combining parts of different photographs into a single composite for a variety of applications. Our system uses a similar approach for combining two images captured in the two phases together to create a final composite. However the photomontage system only deals with pre-aligned still images, while in our work we have to provide solutions to align images taken at different times, and propagate the image completion results on keyframes to all other frames in a temporally coherent way.
3 USER INTERFACE AND WORKFLOW

This section introduces our system from the user's perspective. We will first describe our two-phase capturing process, followed by a set of interactive tools for stop motion keyframing and hand removal (see also examples in the accompanying video). The underlying algorithms behind the user interface will be described in Section 4.

System setup. Our system requires a video camera connected to a computer. Since our main target user groups are amateur animators, in our current setup we use an inexpensive consumer USB webcam (i.e., Logitech HD Pro Webcam C910) for its simplicity and affordability (Figure 1 (left)). Note that although video cameras might be used in traditional stop motion production as well, they are mainly used for capturing still pictures only. Since our main focus is on object motion, we fix the camera on a tripod in our experiments. We will discuss possible solutions for using moving cameras in Section 6.

3.1 Two-Phase Capturing

Phase I. The first phase starts with capturing a clean background plate without the object or the animator. The animator then holds the object directly by hand and performs a desired motion by continuously moving the object in front of the camera (Figure 1). The animator is allowed to change hand placement during capturing, though it may cause the captured motion to be less fluent, requiring additional motion editing in post production.

To achieve the maximum efficiency using our system, the animator is required to hold the object only at its near-rigid parts and to avoid occluding non-rigid parts by his/her hands. This is crucial to the success of the computer vision techniques our system employs. Note that it does not require the whole object to be near-rigid. To ease the hand removal process, there are several rules of thumb for hand placement (Figure 3), listed below in roughly descending order of importance:

- Minimize the total area of the occluded regions on the object.
- Avoid unnecessary overlapping between the user and object, e.g., standing right behind the object (see a negative example in Figure 3 (right)).
- Set up proper lighting conditions to reduce the shadow caused by the animator’s hands or other body parts.

Prior to the second capturing phase, a set of keyframes and the regions of interest (ROIs) must be identified (e.g., the rectangles in orange in Figure 2). One or more occluded parts undergoing the same near-rigid transformations are included in the same ROIs for further processing. For each ROI, the user first creates a tracking window by specifying a bounding box loosely enclosing the ROI on the first frame. The windows will then be automatically propagated to other frames using motion tracking. The user then examines, and corrects if necessary, the bounding boxes on each frame to make sure that the occluded areas of the object are enclosed (i.e., not necessarily the entire object). Next, based on motion analysis, our system automatically suggests a series of keyframes, which the user can refine in the second stage as well. Generally, dense keyframes will be generated if the video involves changing hand-object configurations, or complex object movement such as 3D rotation.

Phase II. In this phase, keyframes selected in Phase I are sequentially processed through a specially-designed user interface. Starting from the first keyframe, the user physically
moves the object to align it with the previously captured object in each keyframe (Figure 2). This time the hand position must be different so that the previously occluded parts are clearly visible in the newly captured image. For more reliable computation, the user is also suggested to place the hands as far away from the previously occluded parts as possible.

We develop an interface based on the idea of onion skinning to facilitate this keyframe-based capturing process. It displays a semi-transparent overlay of the current keyframe and the live frame (Figure 4 (left)). Such visual feedback is greatly helpful for examining the accuracy of alignment. A snapping-like feature is integrated in this interface so that when the object is close enough to its position and orientation in the keyframe, our system automatically warps the current live frame to better align with the keyframe. This snapping feature can greatly relieve the user from the burden of precise alignment. A window of superimposed edge maps is also displayed to guide the alignment process (Figure 4 (right)). Note that onion skinning is also an important feature in traditional stop motion animation and digital animation software like Adobe Flash. However, the purpose of using onion skinning in those approaches is to help the user estimate the change across frames (i.e., to ensure that the movement across frames is neither too small nor too radical) rather than examine the alignment of two frames.

The user is recommended to perform interactive object alignment at the same side as the camera, with the display at the opposite side. Such scenario is similar to a 6-DOF docking task that is often used in evaluating the performance of 6-DOF input devices [7]. From this point of view the object essentially serves as a physical input controller with six degrees of freedom (i.e., 3 for translation and 3 for rotation). Since the user can benefit from kinesthetic feedback about the position and orientation of the physical object, our alignment interface is thus intuitive and efficient (see a live demo in the accompanying video).

Our system has an optional feature which automatically decides whether an alignment is good enough to enable more fluent capturing experience. The user still has full control in the process. He or she can manually signal a good alignment and the corresponding image will then be captured and paired to the current keyframe. Our system also provides controls for back and forth keyframe toggling in case the user is not satisfied with the automatic alignment decision.

3.2 Interactive Hand Removal
After the two-phase capturing process, we have a performance video with a set of selected keyframes from Phase I, and another captured and well-aligned image for each keyframe from Phase II. In the next step, for each keyframe the user invokes a semi-automatic procedure to roughly segment the hands and complete the occluded object regions in the ROIs using image composition. The hand masks and the hand removal results in the keyframes will then be automatically propagated to the intermediate frames between them for restoring occluded regions, resulting in a complete stop motion animation video (Figure 2).

Given the pair of images for each keyframe, our system automatically generates a composite by combining parts of the two images together to remove the hands, using an optimization approach. If the composite contains errors, the user can refine it using brush tools (Figure 5), which are treated as hard constraints in the optimization algorithm. Our system also produces an object mask and a hand mask based on the completion result, which will be used for temporal propagation. The compositing and the masks are updated instantly to give the user rapid visual feedback after each brush stroke is drawn (see a live demo in the accompanying video).

It is worth emphasizing that to generate visually pleasing animation results, our system requires a faithful reconstruction of the occluded parts, but only a rough segmentation of the hands and the object. In fact, a hand mask that is slightly larger than the ground-truth is preferred to reduce artifacts caused by shadows of the hands casting on the object. A rough segmentation of the object is also sufficient for temporal propagation, as we will explain in detail in Section 4.3.2. As a result, for most examples shown in the paper, only a small number of user strokes are needed to achieve satisfactory results.

Finally, the user employs a paint brush to mark the background regions that are outside the keyframe ROIs, but are occluded by hands or other body parts. Since the user is suggested to avoid unnecessary overlap between the user and the object during the performance capturing in Phase I, these occluded regions outside the ROIs can be easily removed and completed using the clean background plate. The resulting occlusion masks are expanded slightly and projected to the intermediate frames to automatically complete the occluded regions there.

4 ALGORITHM AND IMPLEMENTATION
In this section we describe the algorithm and implementation details that support the user interface presented in Section 3.

4.1 ROI and Keyframe Specification
From the algorithm point of view, we are only interested in the local region where the object is occluded by the animator's hands. For brevity, we assume a single occluded rigid part, i.e., a single ROI. Multiple ROIs can be processed sequentially using the same pipeline. As mentioned in Section 3.1, we let the user specify an ROI, i.e., a bounding box which loosely

Fig. 4: UI for keyframe-based alignment: onion skinning with snapping (left) and without snapping (middle), and superimposed edge maps (right).
encloses this region (Figure 5 (a)). The bounding box is then automatically tracked in the rest of the sequence. As we will explain in Section 4.4, we assume that the object’s motion is relatively slow in the captured video, and thus the bounding box can be well tracked by computing optical flow between adjacent frames using a state-of-the-art approach [8]. The user can also adjust the bounding box on any frame, which will re-initiate tracking starting from that frame.

After the ROI is specified, our system automatically estimates object motion within the ROI to determine an initial set of keyframes. A keyframe should be placed at times when the object is undergoing a significant appearance change. To detect this we again use optical flow for motion estimation, since it captures complex motions such as deformation and 3D rotation well, which largely affect the appearance of the object. The optical flow is only computed inside the ROI of each frame with respect to the corresponding region in the next frame. We assign a new keyframe when the accumulated average flow velocity is above a pre-defined threshold with respect to the currently selected keyframe. Specifically, let the average magnitude of the flow field from frame \( t-1 \) to \( t \) be \( |v_t| \), and the current keyframe be \( k \), then the next keyframe \( k+1 \) is selected when \( \sum_{i=1}^{t_k} |v_i| > T_v \), where \( T_v \) is a pre-defined threshold (\( T_v = 20 \) by default in our implementation).

It is worth mentioning that keyframe extraction is a well-studied problem for video summarization (see [9] and references therein). However, keyframes in those approaches serve as a short summary of the video, while our keyframes have a different purpose of guiding temporal interpolation for hand removal in the video.

### 4.2 Onion Skinning and Snapping

A basic onion skinning interface is straightforward to implement as a semi-transparent superimposition of the live frame and the current keyframe with adjustable relative opacity (Figure 4 (left)). Although such semi-transparent overlay provides an instant feedback on alignment, it may cause features from individual images difficult to distinguish. We found that an additional window of superimposed edge maps (using the Sobel operator) can greatly help the user identify a satisfactory alignment (Figure 4 (right)).

Our system requires only a rough alignment between the object in the live frame and its counterpart in the keyframe. This is an important feature for our capturing workflow, since in practice it is very hard for the user to manually reach a perfect alignment (Figure 4 (middle)). To relieve the burden from the user, we compute optical flow between the keyframe ROI and the corresponding region in the live frame on the fly while the user is using the onion skinning interface. The flow is weighted by the pixel-wise difference between the two regions to reduce the disturbance of the hands, since the hands are at different positions in the two images. When the objects in the two images are close enough in both position and orientation (thus satisfying the small movement assumption of optical flow), the ROI in the live frame is warped by the flow and snapped onto the ROI in the keyframe to create a more accurate alignment.

For detecting good alignment, currently our system detects and matches SIFT features in the ROI of both images. If the average distance between the matched feature points is below a threshold (3 pixels in our system), a good match is detected. This criterion works well if many SIFT features can be detected (e.g., for the flipping book in the accompanying video). However, for textureless objects where not enough features exist for reliable feature matching, the system disables this feature. Other matching techniques, such as matching object contours, can be potentially incorporated into the system to augment SIFT matching for automatic alignment evaluation.

### 4.3 Hand Removal

The discussion below focuses on hand removal within the ROI, since removing the animator’s hands or other body parts outside the ROI is straightforward (i.e., by cloning the corresponding pixels in the clean background plate).

#### 4.3.1 Semi-automatic Hand Removal on Keyframes

Given the ROI on a keyframe \( k_i \), denoted as \( R_i \) (Figure 5 (a)), and the corresponding ROI on the live frame (Figure 5 (b)), denoted as \( R'_i \), the goal is to remove the regions occluded by the user’s hands and recover the complete object in \( R_i \) with the help of \( R'_i \). To achieve this, our system employs an automatic algorithm and interactive tools.

For simplicity let us temporarily drop the keyframe index \( i \) from notations. Assuming that \( R'_i \) is already warped by optical flow to align with \( R_i \), our system treats hand removal as an image compositing problem, where each pixel \( C_x \) in the final composite \( C \) is assigned to one of the two possible colors: \( R_x \) or \( R'_x \). In general, assuming \( R'_i \) and \( R_i \) are aligned well, if \( x \) is a pixel that is not occluded by hand in both \( R \) and \( R'_i \), then \( R_x \approx R'_x \), and we prefer to assign \( C_x = R_i \) to minimize the
changes made to $R$. On the other hand, if $x$ is occluded in either $R$ or $R'$, then $R_x$ is significantly different from $R_x'$. In this case we will use a background color model to further determine on which image the occlusion happens. If the occlusion happens on $R$, we then assign $C_x = R_x'$ to reconstruct the object color at pixel $x$.

Specifically, for each pixel $R_x$, we compute two color probabilities: the probability that $x$ is not occluded, denoted as $p^f(x)$, and the probability that $x$ is either occluded or a background pixel, denoted as $p^o(x)$. To compute $p^f(x)$, we first select a set of high confidence pixels for which $\|R_x - R_x'\| < \delta$, where $\delta$ is a small constant which we set as 0.05. We then train a Gaussian Mixture Model (GMM) $G^f$ using the colors of these pixels. Finally, we compute $p^f(x)$ as $p^f(x) = G^f(R_x)$. Similarly, to compute $p^o(x)$, we use all pixels on the border of $R$ to train a GMM color model denoted as $G^o$, and compute $p^o(x)$ as $G^o(R_x)$. Note that $G^o$ contains both hand and background colors.

Energy minimization. To ensure the image labeling is spatially coherent and robust to image noise and color shift, we use an optimization approach under the photomontage framework [6] to solve the labeling problem. Mathematically, we minimize the following objective function:

$$E(L) = \sum_x E_d(x, L(x)) + \lambda \sum_x E_s(x, y, L(x), L(y)), \quad (1)$$

where $L(x)$ is the binary label for pixel $x$, $L(x) = 0$ means $C_x = R_x$, and $L(x) = 1$ means $C_x = R_x'$. $\lambda$ is the balancing weight between the two terms (0.5 in our system). $E_d$ is the data term defined as:

$$E_d(x, L(x)) = \begin{cases} p^o(x)/(p^f(x) + p^o(x)) & L(x) = 0, \\ p^f(x)/(p^f(x) + p^o(x)) & L(x) = 1. \end{cases}$$

The smoothness term $E_s$ in Equation 1 ensures a smooth transition between two different labels. We adopt the “matching color” criterion defined in the photomontage framework and define it as:

$$E_s(x, y, L(x), L(y)) = \|R_x - R_y\| + \|R_y - R_x'\|,$n

when $L(x) \neq L(y)$, and 0 otherwise.

The total energy is then minimized by using graph cuts optimization [10], yielding a composite which is presented to the user for review. If the user is satisfied with the result, the system then proceeds to the next keyframe. Otherwise the user can refine the composite by adding brush strokes as hard constraint. Figure 5 (c) shows an example, where the automatically computed composite has a small error. To fix it the user adds some red ($L(x) = 0$) strokes as hard constraint for optimization (Figure 5 (d)), and at an interactive rate our system recomputes the composite, which is artifact-free as shown in Figure 5 (e).

Note that unoccluded background pixels in $R$ will be included for training both $G^f$ and $G^o$, thus the two probabilities values $p^f(x)$ and $p^o(x)$ will both be high for these pixels, and the data term $E_d(x)$ does not favor either label. This is not a problem in our application as these pixels do not affect the goal of hand removal, and we let the smoothness energy $E_s$ play the dominant role for these pixels to create a seamless composite. As shown in the example in Figure 5 (f), although the labeling function does not accurately segment either the object or the hand, the final composite is good enough for our purpose of hand removal.

4.3.2 Automatic Temporal Propagation

Once satisfactory hand removal results are achieved on two adjacent keyframes $k_1$ and $k_2$, with their completion results denoted as $C_{k_1}$ and $C_{k_2}$, our system employs an automatic temporal interpolation method for hand removal on all in-between frames $j$, $k_1 < j < k_2$.

For temporal propagation we further process keyframe $k_1$ to generate a hand mask $M_{k_1}^{hand}$ and an unoccluded object mask $M_{k_1}^{obj}$, with the help of the clean background plate $B$.

Specifically, the hand mask is computed as:

$$M_{k_1}^{hand}(x) = T_{h}(\|R_{k_1}(x) - B(x)\|) \& T_{h}(\|R_{k_1}(x) - C_{k_1}(x)\|), \quad (4)$$

where $T_{h}(\varepsilon)$ is a step function that $T_{h}(\varepsilon) = 1$ if $\varepsilon > T_{c}$, and 0 otherwise. $T_{c}$ is a predefined color threshold which is set to be 0.1 in our system. The unoccluded object mask $M_{k_1}^{obj}$ is then computed as:

$$M_{k_1}^{obj}(x) = T_{h}(\|R_{k_1}(x) - B(x)\|) - M_{k_1}^{hand}(x). \quad (5)$$
Two examples of $M^{obj}$ and $M^{hand}$ masks are shown in Figure 6 (b) and (d).

We then compute optical flow from $R_{k_1}$ to $R_j$, by using only pixels inside the object mask $M^{obj}_{k_1}$ (as data term), which allows more accurate estimation of object motion. The masked flow field is then extrapolated and used to warp $C_{k_1}$ and $M^{hand}_{k_1}$ to frame $j$, denoted as $C'_j$ and $M^{hand}_j$. Similarly, we compute $C'_{k_2}$ and $M^{hand}_{k_2}$ from keyframe $k_2$. We then linearly blend $C'_{k_1}$ and $C'_{k_2}$ to create a reference object image $C_j$, on which the object is complete, and compute a final hand mask for frame $j$ as $M^{hand}_j = M^{hand}_{k_1} \cap M^{hand}_{k_2}$. Finally, to remove the hands and complete the occluded object regions in $R_j$, we apply Poisson blending [11] to seamlessly blend the original ROI $R_j$ with $C_j$ in the hand region $M^{hand}_j$, as shown in Figure 6 (bottom).

Note that in Equations 4 and 5, the masks are computed as per-pixel operations. In theory we could improve the robustness of mask calculation by using another graph cuts optimization process. However in practice we found this is not necessary as we do not require accurate object and hand masks. The unoccluded object mask is mainly used for motion estimation, thus small errors do not affect its performance. For the hand mask $M^{hand}_j$, we intentionally dilate it after computing it to remove shadow artifacts around the hand regions. The per-pixel operations can give the user fast visual feedback, which we found more important for a fluent workflow. In case the object casts strong shadows to the scene (e.g., the Rubik’s Cube in Figure 1), which would appear in $M^{obj}$ and possibly make optical flow less robust, the user can roughly exclude the object shadow regions using a paint brush.

4.4 Video Speed Adjustment

Since we use a video camera to capture the animation, motion blur is inevitable when the object moves too fast. As motion blur usually does not exist in traditional stop motion animation, to mimic the same visual effect, our system requires the user to move the object slowly. This will also benefit motion tracking and image completion, and greatly reduce the amount of required user assistance.

Traditional stop motion animation has the sense of motion discontinuity between adjacent frames. Simply playing our processed video at its original speed may not produce the same effect. We thus sample the video for every 5 or more frames to create a artistically more pleasing animation. The user can also fine tune the sampling rate at different parts of the video for more personalized results in post-production.

5 RESULTS AND DISCUSSION

We have applied our technique to animating a variety of everyday objects in stop motion style. All the animations presented in the paper and the accompanying video were created by amateur users who have little experience in making either digital animation or stop motion. All the examples were made in a daily working environment instead of a professional studio with special lighting setup and blue/green screens for background replacement in post production.

In traditional stop motion production, even for a simple example like moving a computer mouse on the table (see the accompanying video) to mimic the motion of a real mouse, animators have to carefully plan and perform the motion in increments. Using our system, the animator performs and captures the desired motion in a continuous fashion. Our workflow essentially separates motion performance (Phase I),
which might require an experienced animator, from keyframe-based interactive rendering (Phase II), which requires some amount of user input but can be done by amateur users. In other words, it is not necessary to have the same animator involved in both phases. On the other hand, the flexibility of our system is at the cost of the lack of precise control (e.g., for achieving carefully crafted ease in/out effects), possibly requiring motion editing in post production.

With our tool, the animation of pouring water from a cup (Figure 7 (a)) can be produced in its natural way. In contrast, this type of animation is difficult to achieve even using complicated rigs or other supporting tools in traditional stop motion production. We asked two artists majored in animation to make a similar animation with whatever hardware and software they like. Although they finally managed to produce a similar animation using the DragonFrame software and hardware tools like transparent adhesive tape and wires, they failed to reproduce the motion of water in their still shots. The two artists spent around 2 hours producing an animation with 30 still shots only (cf. with our tool a single amateur produced the animation with 512 frames within 2.5 hours). The artists also commented that animations involving complex 3D rotation, like our examples of walking stool (Figure 7 (d)), are challenging to make even with the help of rigs, since rig-object configurations have to be changed frequently to accommodate complex motions. In contrast, making such animations is easy by changing hand placement during the first capturing phase in our system.

In Figure 7 (c), we show an example of plant pruning by a scissor which involves multi-body motion. Two body parts were simultaneously manipulated and animated by two hands during the first capturing phase. Therefore, this example requires two ROIs, which are processed sequentially in Phase II. Figure 1 shows another example involving multi-body motion. Here, this example also shows easy integration of our technique with traditional stop motion production for the rotation of individual faces of the Rubik’s Cube.

With direct hand manipulation, it is easy to animate relatively heavy objects like walking stool (Figure 7 (d)). The flipping book example shown in Figure 7 (b) involves continuous rotation around a fixed axis. Given that such motions have only one degree of freedom, it is possible to obtain exactly the same trajectory during the second capturing phase, though the moving speed might be different. Hence, although our keyframe-based alignment and capturing approach still works, a more efficient approach might be to first capture the motion in Phase II as a video as well and then synchronize the two videos by establishing temporal correspondence between their frames [12].

**Hand placement.** Generally the bigger the object is, the more flexible the placement of the hands can be, since the hands only occlude a small portion of the object. On the other hand, our system may become less reliable when the object is small and the majority of the object is occluded, since motion tracking is difficult in this case, and reconstructing a large portion of the object may introduce noticeable artifacts. The right solution for animating small objects is to use suitable tools (e.g., a scissor used in the typing keyboard example) to drive the animation, which can still be removed using our system.

**Number of keyframes.** Table 1 shows the number of keyframes used in some of our examples. Generally speaking, dense keyframes are needed for fast or complex motion, such as 3D rotation. On average, one keyframe was used for every 12 frames. For objects that are hard to track due to the lack of features or fast motion (e.g., the walking stool), more keyframes are needed to generate satisfactory interpolation results. In the worst case, such as animating an almost transparent object that is difficult to track even using the best computer vision techniques, a keyframe is needed for every two or three frames. In this case, the total amount of user interaction needed may be comparable to that in traditional stop motion production. However we believe that some users may still prefer our system given its flexibility of direct hand manipulation.

**Timings.** As shown in Table 1, the examples presented in this paper took from 1-2 hours to half a day to produce, including the time for motion planning and post production. The total user time largely depends on the number of keyframes extracted in each example. It took from dozens of seconds to one or two minutes per keyframe for object alignment in Phase II. For interactive hand removal in each keyframe, only a small number of strokes on a few keyframes were needed to achieve reasonably good results. On average the user spent about half a minute on each keyframe for interactive hand removal. For the temporal interpolation approach described in Section 4.3.2, currently our unoptimized implementation achieved a few seconds per-frame (depending on the ROI size), which can be further accelerated for a more rapid visual feedback.

With traditional stop motion techniques, it would typically take a much longer time to create animations similar to those shown in the paper and the accompanying video. This has been confirmed by two professional animators we interviewed, who teach animation courses related to stop motion in a prestigious art school. For instance, they expected that at least 2 days (cf. half a day in our case) are needed even for professionals to complete an animation similar to our walking stool example. They also felt it is difficult to reproduce the continuous motion of water by still shots.

### 6 Conclusion and Future Work

We have demonstrated a video-based interface for creating stop motion animations of everyday objects. Our system provides a fluent workflow with direct hand manipulation, which is particularly beneficial for amateur users, who typically have difficulties in performing a desired motion by integrating small motion increments, and in making and using rigs for object

<table>
<thead>
<tr>
<th>Examples</th>
<th>#Frames</th>
<th>#Keyframes</th>
<th>Time (hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rubik’s Cube</td>
<td>503</td>
<td>33</td>
<td>~3</td>
</tr>
<tr>
<td>Flipping book</td>
<td>201</td>
<td>21</td>
<td>~0.5</td>
</tr>
<tr>
<td>Water pouring</td>
<td>512</td>
<td>43</td>
<td>~2.5</td>
</tr>
<tr>
<td>Moving mouse</td>
<td>341</td>
<td>21</td>
<td>~1</td>
</tr>
<tr>
<td>Walking stool</td>
<td>310</td>
<td>70</td>
<td>~6</td>
</tr>
</tbody>
</table>

**TABLE 1:** Statistics of keyframes and timings.
control. Our tool can also be used together with traditional stop motion techniques for making professional animations.

We speculate that the number of keyframes used is far from optimal. A better solution might be: starting from a very sparse set of keyframes, let the user recursively insert a new keyframe between two keyframes that produce an unsatisfactory temporal interpolation result. This requires a faster implementation of the temporal propagation algorithm for real-time feedback.

Our system is of course not designed for animating every possible object, and it has a number of limitations that the user should be aware of. Our system is based on the key assumption that occluded object parts in the first capturing phase are available in the second phase. However this is not true for highly deformable objects such as clothes, which are easily distorted by direct hand manipulation. Our system may have difficulties with this type of objects.

Our automatic temporal propagation step first uses optical flow to estimate the motion of unoccluded parts and then extrapolates the flow to the occluded regions as a warp field. Thus the problems due to either optical flow or flow extrapolation would cause artifacts. Optical flow is rather weak at estimating out-of-plane rotation and performs poorly for regions with low texture (Figure 8). Our smoothness-based extrapolation may fail to respect objects features, e.g., straight edges as shown in Figure 8. Such problems can be alleviated by inserting more keyframes in our current system. A better solution might be to impose rigid constraints on the formulation of optical flow or extrapolation, e.g., in a sense of structure-preserving image deformation [13]. An alternative solution is to employ roughly reconstructed 3D models (e.g., by using an RGB-D camera like the Microsoft Kinect [2] instead of our traditional webcam), which afford much more robust (3D) object tracking, thus leading to more accurate warp fields.

Camera movement and/or dynamic background can further complicate the problem and make the system less efficient. In these cases a clean background plate is not available, thus the user has to provide more manual input for ROI tracking and keyframe segmentation, and also use denser keyframes to avoid temporal interpolation errors. Furthermore, the system has to employ video hole filling techniques [5] to fill in the background regions that are occluded by the user’s hands as well. For a moving camera with static background, another possible solution is to roughly reposition the camera in Phase II with respect to the camera movement in the first video, and then resort to advanced video matching algorithms [12] that are able to bring two videos following nearly identical trajectories through space into spatiotemporal alignment.

In case of changing lighting conditions, images captured from Phase II may have significantly different appearances than the video captured in Phase I, which will make a number of components that rely on computing color difference fail. A possible solution is to examine how the lighting changes in common regions of the scene captured in the two phases, and apply relighting techniques to cancel out the effect of lighting difference before applying our system.

Even with onion skinning, keyframe-based object alignment (needed in Phase II) becomes more difficult when desired object motion involves more degrees of freedom. In particular, it is the most tricky when an object is completely floating in the air and involves both 3D translation and rotation (i.e., 6-DOF), since there is no physical reference for alignment in this case. To alleviate this problem, the user is suggested to continuously and sequentially perform object alignment keyframe by keyframe, as object movement across adjacent keyframes is typically small, thus requiring only slight movement of the object to get aligned. However, this would easily cause fatigue for animating heavy objects like the walking stool. Furthermore, it is observed that object alignment is less intuitive when the user is operating in front the camera (Figure 1) rather than at the same side as the camera. In such cases, although the live frame can be horizontally flipped to get a mirrored effect, which the user might be more familiar with, rotation/translation towards or away from the camera is rather counterintuitive to control.

Although our system is limited in a number of cases discussed above, we have demonstrated in this paper that our system is already useful and efficient for creating stop motion animations for a wide variety of objects, and has advanced the state-of-the-art in low-budget stop motion production. As a future work we plan to address these limitations for building a more robust system.

ACKNOWLEDGEMENTS

We thank the reviewers for their constructive comments, Michael Brown for video narration, Lok Man Fung and Wai Yue Pang for experimenting traditional stop motion production, and Tamas Pal Waliczky and Hiu Ming Eddie Leung for their professional comments. This work was partially supported by grants from the Research Grants Council of HKSAR (No. CityU113610, No. CityU113513), the City University of Hong Kong (No. 7002925 and No. 7002776), the National Natural Science Foundation of China (No. 61222206), and the National Basic Research Program of China (No. 2011CB302400).

REFERENCES


Y. Boykov and O. Veksler, Graph cuts in vision and graphics: theories and applications. Springer, 2005, ch. 5.


Xiaoguang Han is a Research Associate in the School of Creative Media, City University of Hong Kong. He received his M.Sc. in Applied Mathematics (2011) from Zhejiang University, and his B.S. in Information & Computer Science (2009) from Nanjing University of Aeronautics and Astronautics, China. His research interests include computer graphics and computer vision.

Hongbo Fu is currently an Assistant Professor in the School of Creative Media, City University of Hong Kong. Before joining CityU, he had postdoctoral research trainings at the Imager Lab, University of British Columbia, Canada and the Department of Computer Graphics, Max-Planck-Institut Informatik, Germany. He received the PhD degree in computer science from the Hong Kong University of Science and Technology in 2007 and the BS degree in information sciences from Peking University, China, in 2002. His primary research interests fall in the field of computer graphics with an emphasis on digital geometry processing.

Hanlin Zheng is a Ph.D. candidate of applied mathematics in Zhengjiang University of China. He received his B.Sc. in applied mathematics in 2008 from Zhejiang University, China. He paid an academic visit to the City University of Hong Kong during 2011 and 2012. His research interests include computer graphics and digital geometry processing.

Ligang Liu is a professor at the Graphics&Geometric Computing Laboratory, University of Science and Technology of China. He received his B.Sc. in applied mathematics (1996) and his Ph.D. in computer graphics (2001) from Zhejiang University, China. Between 2001 and 2004, he worked at Microsoft Research Asia as an associate researcher. Then he worked at Zhejiang University as an associate professor and professor during 2004 and 2012. He paid an academic visit to Harvard University during 2009 and 2011. His research interests include digital geometric processing, computer graphics, and image processing. His research works could be found at his research website: http://staff.ustc.edu.cn/~ligliu

Jue Wang is currently a Senior Research Scientist at Adobe Research. He received his B.E. (2000) and M.Sc. (2003) from Department of Automation, Tsinghua University, Beijing, China, and his Ph.D (2007) in Electrical Engineering from the University of Washington, Seattle, WA, USA. He then joined Adobe Research in 2007 as a research scientist. His research interests include image and video processing, computational photography, computer graphics and vision. He is a senior member of IEEE and a member of ACM.