# Video Stitching for Linear Camera rrays

Wei-Sheng Lai[1,2]
wlai24@ucmerced.edu
Orazio Gallo[2]
ogallo@nvidia.com
Jinwei Gu[2]
gujinwei@gmail.com
Deqing Sun[2]
deqing.sun@gmail.com
Ming-Hsuan Yang[1]
mhyang@ucmerced.edu
Jan Kautz[2]
jkautz@nvidia.com

[1] University of California Merced
[2] NVIDIA

**bstract**

Despite the long history of image and video stitching research, existing academic and commercial solutions still produce strong artifacts. In this work, we propose a wide-baseline video stitching algorithm for linear camera arrays that is temporally stable and tolerant to strong parallax. Our key insight is that stitching can be cast as a problem of learning a smooth spatial interpolation between the input videos. To solve this problem, inspired by pushbroom cameras, we introduce a fast pushbroom interpolation layer and propose a novel pushbroom stitching network, which learns a dense flow field to smoothly align the multiple input videos for spatial interpolation. Our approach outperforms the state-of-the-art by a significant margin, as we show with a user study, and has immediate applications in many areas such as virtual reality, immersive telepresence, autonomous driving, and video surveillance.



(a) Our stitching result (b) [19] (c) [3] (d) [12] (e) Ours

Figure 1: **Examples of video stitching.** Inspired by pushbroom cameras, we propose a deep pushbroom stitching network to stitch multiple wide-baseline videos of dynamic scenes into a single panoramic video. The proposed learning-based algorithm outperforms prior work with minimal mis-alignment artifacts (*e.g.*, ghosting and broken objects). More video results are presented in the supplementary material.

# 1 Introduction

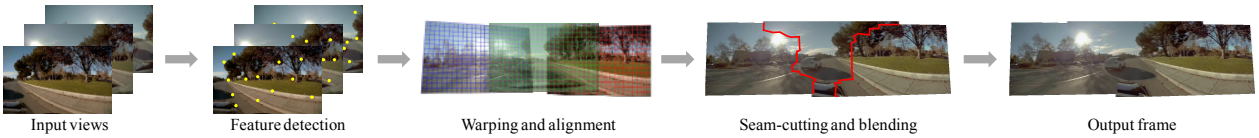
Due to sensor resolution and optics limitations, the field of view (FOV) of most cameras is too narrow for applications such as autonomous driving and virtual reality. A common solution is to stitch the outputs of multiple cameras into a panoramic video, effectively extending the FOV. When the optical centers of these cameras are nearly co-located, stitching can be solved with a simple homography transformation. However, in many applications, such as autonomous driving, remote video conference, and video surveillance, multiple cameras have to be placed with *wide baselines*, either to increase view coverage or due to some physical constraints. In these cases, even state-of-the-art methods [12, 21] and current commercial solutions (*e.g.*, VideoStitch Studio [27], AutoPano Video [3], and NVIDIA VRWorks [19]) struggle to produce artifact-free videos, as shown in Figure 1.

One main challenge for video stitching with wide baselines is *parallax*, *i.e.* the apparent displacement of an object in multiple input videos due to camera translation. Parallax varies with object depth, which makes it impossible to properly align objects without knowing dense 3D information. In addition, occlusions, dis-occlusions, and limited overlap between the FOVs also cause a significant amount of stitching artifacts. To obtain better alignment, existing image stitching algorithms perform content-aware local warping [6, 30] or find optimal seams around objects to mitigate artifacts at the transition from one view to the other [9, 31]. Applying these strategies to process a video frame-by-frame inevitably produces noticeable jittering or wobbling artifacts. On the other hand, algorithms that explicitly enforce temporal consistency, such as spatio-temporal mesh warping with a large-scale optimization [12], are computationally expensive. In fact, commercial video stitching software often adopts simple seam cutting and multi-band blending [5]. These methods, however, often cause severe artifacts, such as ghosting or misalignment, as shown in Figure 1. Moreover, seams can cause objects to be cut off or completely disappear from stitched images—a particularly dangerous outcome for use cases such as autonomous driving.

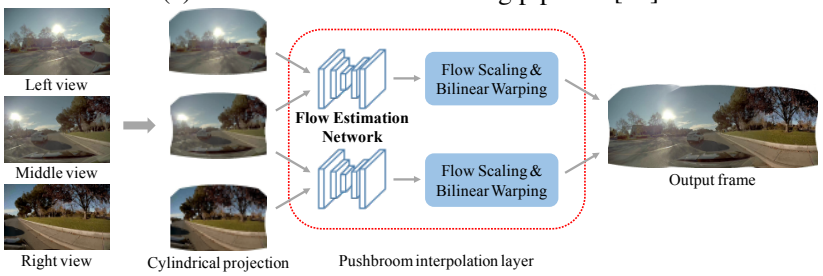
We propose a video stitching solution for linear cameras arrays that produces a panoramic video. We identify three desirable properties in the output video: (1) Artifacts, such as ghosting, should not appear. (2) Objects may be distorted, but should not be cut off or disappear in any frame. (3) The stitched video needs to be temporally stable. With these three desiderata in mind, we formulate video stitching as a spatial view interpolation problem. Specifically, we take inspiration from the pushbroom camera, which concatenates vertical image slices that are captured while the camera translates [10]. We propose a pushbroom stitching network based on deep convolutional neural networks (CNNs). Specifically, we first project the input views onto a common cylindrical surface. We then estimate bi-directional optical flow, with which we *simulate* a pushbroom camera by interpolating all the intermediate views between the input views. Instead of generating all the intermediate views (which requires multiple bilinear warping steps on the entire image), we develop a *pushbroom interpolation layer* to generate the interpolated view in a single feed-forward pass. Figure 2 shows an overview of the conventional video stitching pipeline and our proposed method. Our method yields results that are visually superior to existing solutions, both academic and commercial, as we show with an extensive user study.

# 2 Related Work

**Image stitching.** Existing image stitching methods often build on the conventional pipeline




(a) Conventional video stitching pipeline [12]




(b) Proposed pushbroom stitching network

Figure 2: **lgorithm overview.** (a) Conventional video stitching algorithms [12] use spatio-temporal local mesh warping and 3D graph cut to align the entire video, which are often sensitive to scene content and computationally expensive. (b) The proposed pushbroom stitching network adopts a pushbroom interpolation layer to gradually align the input views, and outperforms prior work and commercial software.

of Brown and Lowe [4], which first estimates a 2D transformation (*e.g.*, homography) for alignment and then stitches the images by defining seams [9] and using multi-band blending [5]. However, ghosting artifacts and mis-alignment still exist, especially when input images have large parallax. To account for parallax, several methods adopt spatially varying local warping based on the affine [18] or projective [30] transformations. Zhang *et al*. [31] integrate the content-preserving warping and seam-cutting algorithms to handle parallax while avoiding local distortions. More recent methods combine the homography and similarity transforms [6, 16] to reduce the projective distortion (*i.e.*, stretched shapes) or adopt a global similarity prior [7] to preserve the global shape of the resulting stitched images.

While these methods perform well on still images, applying them to videos frame-by-frame results in strong temporal instability. In contrast, our algorithm, also single-frame, generates videos that are spatio-temporally coherent, because our pushbroom layer only operates on the overlapping regions, while the rest is directly taken from the inputs.

**Video stitching.** Due to computational efficiency, it is not straightforward to enforce spatio-temporal consistency in existing image stitching algorithms. Commercial software, *e.g.*, VideoStitch Studio [27] or AutoPano Video [3], often finds a fixed transformation (with camera calibration) to align all the frames, but cannot align local content well. Recent methods integrate local warping and optical flow [21] or find a spatio-temporal content-preserving warping [12] to stitch videos, which are computationally expensive. Lin *et al*. [17] stitch videos captured from hand-held cameras based on 3D scene reconstruction, which is also time-consuming. On the other hand, several approaches, *e.g.*, Rich360 [15] and Google Jump [2], create 360° videos from multiple videos captured on a structured rig. Recently, NVIDIA released VRWorks [19], a toolkit to efficiently stitch videos based on depth and motion estimation. Still, as shown in Figure 1(b), several artifacts, *e.g.*, broken objects and ghosting, are visible in the stitched video.

In contrast to existing video stitching methods, our algorithm learns local warping flow fields based on a deep CNN to effectively and efficiently align the input views. The flow is learned to optimize the quality of the stitched video in an end-to-end fashion.

**Pushbroom panorama.** Linear pushbroom cameras [10] are common for satellite imagery:

while a satellite moves along its orbit, they capture multiple 1D images, which can be concatenated into a full image. A similar approach has also been used to capture street view images [24]. However, when the scene is not planar, or cannot be approximated as such, they introduce artifacts, such as stretched or compressed objects. Several methods handle this issue by estimating scene depth [22], finding a cutting-seam on the space-time volume [29], or optimizing the viewpoint for each pixel [1]. The proposed method is a software simulation of a pushbroom camera which creates panoramas by concatenating vertical slices that are spatially interpolated between the input views. We note that the method of Jin *et al*. [13] addresses a similar problem of view morphing, which aims at synthesizing intermediate views along a circular path. However, they focus on synthesizing a single object, *e.g*. a person or a car, and do not consider the background. Instead, our method synthesizes intermediate views for the entire scene.

# 3 Stitching as Spatial Interpolation

Our method produces a temporally stable stitched video from wide-baseline inputs of dynamic scenes. While the proposed approach is suitable for a generic linear camera array configuration, here we describe it with reference to the automotive use case. Unlike other applications of structured camera arrays, in the automotive case, objects can come arbitrarily close to the cameras, thus requiring the stitching algorithm to tolerate large parallax.

For the purpose of describing the method, we define the camera setup as shown in Figure 3(a), which consists of three fisheye cameras whose baseline spans the entire car's width. Figures 4(a)-(c) show typical images captured under this configuration, and underscore some of the challenges we face: strong parallax, large exposure differences, as well as geometric distortion. To minimize the appearance change between the three views and to represent the wide FOV of the stitched frames, we first adopt a camera pose transformation to warp $C_L^i$ and $C_R^i$ to the position of $C_L^o$ and $C_R^o$, respectively. Therefore, the new origin is set at the center camera $C_M$. Then, we apply a cylindrical projection (by approximating the scene to be at infinity) to warp all the views onto a common viewing cylinder, as shown in Figure 3(a). However, even after camera calibration, exposure compensation, fisheye distortion correction, and cylindrical projection, parallax still causes significant misalignment, which results in severe ghosting artifacts, as shown in Figure 4(d).

## 3.1 Formulation

In this work, we cast video stitching as a problem of spatial interpolation between the side views and the center view. We denote the output view by $\mathcal{O}$, and the input views (projected onto the output cylinder) by $I_L$, $I_M$, and $I_R$, respectively. Note that $I_L$, $I_M$, and $I_R$ are in the same coordinate system and have the same resolution. We define a *transition region* as part of the overlapping region between a pair of inputs (see the yellow regions in Figure 3(b)). Within the transition region, we progressively warp $K$ vertical slices from both images to create a smooth transition from one camera to another. Outside the transition region, we directly take the pixel values from the input images without modifying them.

For presentation clarity, here we focus only on $I_L$ and $I_M$. Our goal is to generate $K$ intermediate frames, $\hat{I}_L^{k)}$, which smoothly transition between $I_L$ and $I_M$. We first compute the bidirectional optical flows, $F_{L\to M}$ and $F_{M\to L}$, and then generate warped frames $\hat{I}_L^{k)} = \mathcal{W}\ I_L,\ \ _k \cdot F_{L\to M})$ and $\hat{I}_M^{k)} = \mathcal{W}\ I_M,\ 1 \quad _k) \cdot F_{M\to L})$, where $\mathcal{W}\ I,F)$ is a function