Pocket Time-Lapse

ERIC M. CHEN, MIT CSAIL, Cornell University, USA
ŽIGA KOVAČIČ, Cornell University, USA
MADHAV AGGARWAL, Cornell University, USA
ABE DAVIS, Cornell University, USA

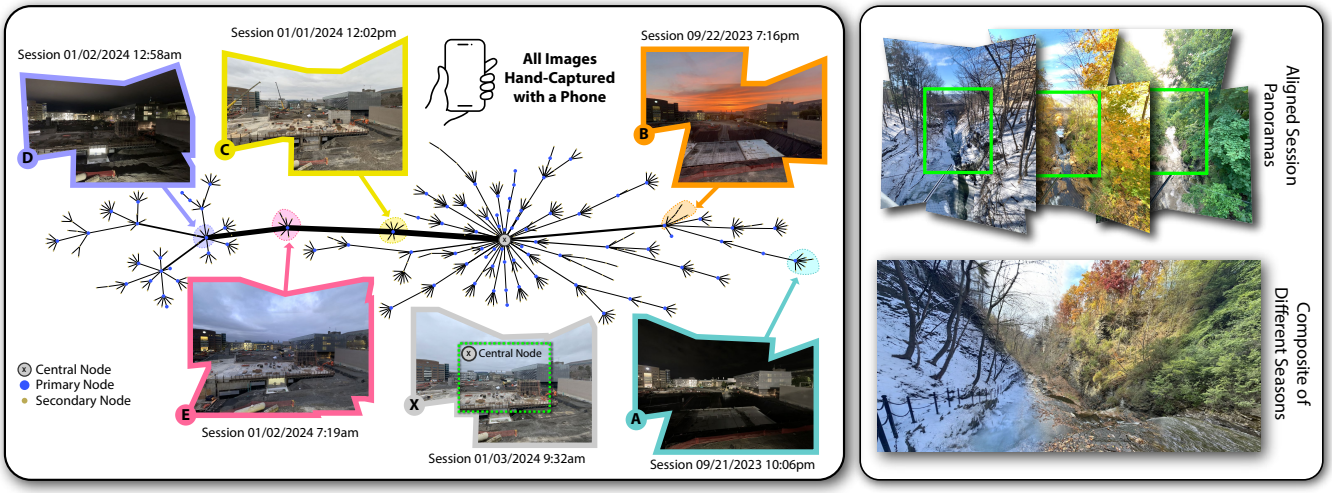

Fig. 1. We present a system for capturing, registering, and visualizing panoramic time-lapse in uncontrolled settings using a hand-held mobile phone. **Left:** a visualization (described more in Section 4) of the Alignment Graph for thousands of construction site images taken over more than a year. The central node, marked X, defines an image plane for our time-lapse, to which all other images are registered by chaining together high-confidence homographies between image pairs. The width of the edges visualized here is proportional to the number of images that use a given homography in registration. Our method successfully registers images with few or no correspondences by finding paths through other images. For example, session D is taken at night with snowfall but registers with X by chaining together matches to sessions taken before and after twilight and during the day (C and E). **Right (top):** Three aligned panoramas taken of one hiking trail over three different seasons. **Right (bottom):** a composite of panoramas for a different hiking trail, where the composite transitions from Winter on the left through Fall and Spring to Summer on the right.

This paper explores how to record, explore, and visualize long-term changes in an environment—at the scale of days, months, and even years—based on data that a single user can conveniently capture using the mobile phone they already carry. Our strategy involves making the data capture process as quick and convenient as possible so that it is easy to integrate into daily routines. This strategy yields large unstructured panoramic image datasets, which we process using novel registration and scene reconstruction approaches. Our central contribution lies in demonstrating pocket time-lapse as a novel application, made possible through several key technical contributions. These include a novel method for quickly and robustly registering thousands of unstructured panoramic images, a novel reconstruction technique for rendering time-lapse and performing state-of-the-art intrinsic image decomposition, and several large hand-captured datasets that span multiple years of data collection, totaling over 6k separate capture sessions and 50k images.

SIGGRAPH Conference Papers '25, August 10–14, 2025, Vancouver, BC, Canada

ACM ISBN 979-8-4007-1540-2/2025/08.
https://doi.org/10.1145/3721238.3730594

CCS Concepts: • **Computing methodologies** → **Computational photography**; *Computer vision*; • **Human-centered computing** → Visualization;

ACM Reference Format:
Eric M. Chen, Žiga Kovačič, Madhav Aggarwal, and Abe Davis. 2025. Pocket Time-Lapse. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference Papers '25), August 10–14, 2025, Vancouver, BC, Canada.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3721238.3730594

1 INTRODUCTION

Most smartphone users carry a pocket-sized camera with them almost everywhere they go. However, most of the pictures we take focus on sudden salient events, leaving slower, more subtle visual changes in our environment comparatively under-sampled. This paper focuses on a simple but powerful idea: How can we leverage the mobile phone we already carry to capture and visualize slower, long-term changes in our environment? Answering this question has broad implications—not just for personal photography, but for many other applications that rely on long-term visual observation in uncontrolled settings, including scientific fieldwork, agriculture,

construction, and structural health monitoring. With this broad range of applications in mind, we spent several years collecting what we call ***pocket time-lapse***. The idea of pocket time-lapse is to use a mobile phone to capture consistent panoramic viewpoints of a scene over long periods by integrating quick and convenient data capture into users' daily routines. Over time, we collect and register this data to reconstruct a time-lapse, offering a powerful way to visualize long-term changes in the scene.

1.1 Challenges

At first glance, pocket time-lapse may seem like a straightforward extension of panoramic image capture. Were this true, one might expect casually hand-captured time-lapse to be common, especially in public or uncontrolled environments where setting up a fixed camera or tripod is impossible. However, hand-captured time-lapse is extremely rare, and most examples are limited to popular landmarks based on data harvested from large Internet photo collections (e.g. Lin et al. [2023]; Martin-Brualla et al. [2015b]). Our work aims to make it easy for anyone to capture time-lapse of environments they frequently visit using devices they already carry. This proves to be challenging for several reasons:

- **Long-Term Data Capture:** To record long-term phenomena, we need to capture data regularly and frequently for months or even years at a time, requiring a significant longitudinal effort.
- **Registration:** Pocket time-lapse data is difficult to register for several reasons. First, the number of images that need to be registered is orders of magnitude greater than what existing panorama tools are designed for. Second, panoramic data makes feature triangulation impossible, precluding strategies used by more scalable 3D registration tools like COLMAP [Schönberger and Frahm 2016] to filtering image matches. Third, sudden and significant visual changes (e.g., night and day, sudden snow cover) frequently result in unreliable matches even between consecutive capture sessions, causing sequential registration strategies to fail.
- **Reconstruction & Visualization:** Pocket time-lapse sampling tends to be much sparser and less uniform than traditional time-lapse, and different visual phenomena of interest often happen at very different timescales. All of this makes reconstruction and visualization especially difficult.

These challenges are tightly interconnected. For example, registration would be easier with multi-view data, and reconstruction would be easier with denser temporal sampling, but both would require additional effort to capture—effort that may need to be sustained for years on end. The potential and nature of using foundation models in reconstruction also presents a tradeoff between fidelity and potential accuracy. To navigate these tradeoffs, it helps to keep clear goals and potential applications in mind.

1.2 Applications & Contributions

Pocket time-lapse has many potential applications beyond personal photography, with many of its most compelling use cases being in fields where data collection cannot (or should not) be replaced with plausible hallucination. For example, in construction, structural health monitoring, agriculture, or environmental science, where manual data collection is already common and accuracy is crucial.

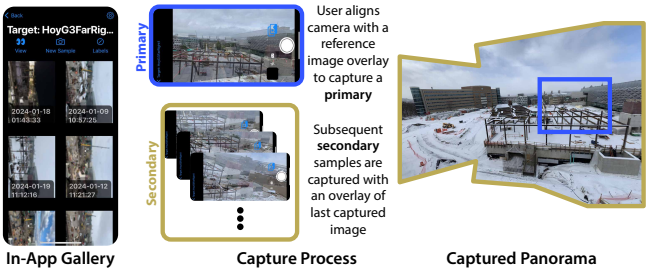

Fig. 2. **Capture Process:** We use a custom panoramic capture app to acquire data. The user selects their target from a gallery to begin capture. A reference view is displayed as an overlay on the screen to help capture an initial primary sample image. Each time they capture a photo, the overlaid image is replaced with the newly captured one to help gauge overlap between consecutive photos. Our registration pipeline later aligns and stitches these images offline. The image on the right shows a stitched panorama, with the region taken from the primary sample marked in blue.

In this paper, we focus on the collection, registration, and visualization of data that supports these types of applications, noting that generative reconstruction conditioned on scarcer data offers increasingly compelling alternative tradeoffs for use cases that focus more exclusively on visual appeal.

Our work represents an extensive effort spanning several years. To support regular and frequent capture without placing an unreasonable burden on users, we developed a simple mobile application to quickly capture panoramic data from consistent viewpoints. We used this application to routinely capture data of several subjects over a roughly three-year period, collecting over 50k images of 17 subjects spread over 6k unique capture sessions. Parallel to this effort, we developed a scalable registration pipeline that can quickly register thousands of panoramic images from uncontrolled dynamic environments (e.g., construction sites and hiking trails) while adjusting to significant changes in lighting and weather. Finally, we built tools and techniques for exploring and visualizing this unstructured data, as well as performing intrinsic image decomposition on captured subjects, which we demonstrate on a wide range of difficult and unique scenes. Our app, code and data will be made available at https://pocket-timelapse.github.io.

2 RELATED WORK

2.1 Traditional & Computational Time-Lapse

Time-lapse videos are traditionally recorded by taking numerous photos at short, regular intervals with a static camera, typically fixed on a tripod throughout the duration of the scene capture. Time-lapse captured in this way is considered a gold standard for data quality, as it provides pre-aligned images with dense, uniform temporal sampling. However, it can be impractical or expensive due to the need to leave a camera in place for extended periods, which is often impossible in public settings. Traditional time-lapse also tends to exhibit significant temporal aliasing. Previous work has focused on removing this aliasing by filtering out high frequencies from the time-lapse Rubinstein et al. [2011]. Matusik et al. [2004] use time-lapse data to represent images as a product of the reflectance

field and the incoming illumination but cannot disentangle the two. Sunkavalli et al. [2007] can implicitly remove transient objects (e.g., people) but is limited to fixed viewpoints and clear-sky conditions.

2.2 Internet Photo Collections

Matzen and Snavely [2014], Martin-Brualla et al. [2015a,b], and Lin et al. [2023], reconstruct 3D scenes over time from heavily captured Internet photo collections. Reliance on Internet photo collections limits such techniques almost exclusively to popular tourist attractions, for which many photos can be found online. These methods rely heavily on large 3D reconstructions and assume that scene geometry is constant over time. Finally, these methods require multi-view data and rely on 3D reconstructions that are much slower and more expensive to compute than our approach.

2.3 Generative Models

In recent years, incredible progress has been made in generative models. Results produced by these methods are visually impressive but favor plausibility over accuracy, making them prone to hallucination. TLGAN [Härkönen et al. 2022] uses a GAN conditioned on time to disentangle time-lapse sequences in a way that allows control of overall trends, seasons, and day-night cycles. However, as we show, TLGAN struggles to reconstruct plausible shadows and the correct season, especially with sparse data. Methods such as Geyer et al. [2024] and Chen et al. [2023] use diffusion models for video generation. However, these approaches also hallucinate, and consistency is difficult to enforce. Preventing hallucination in generative models remains an open problem.

2.4 Guided Photography Systems

With specific capture goals in mind, researchers have studied AR-based guidance for creative photography. For example, Adams et al. [2008] interactively visualizes a coverage map made from panoramic images in viewfinders. Recent works from the HCI community, such as E et al. [2020, 2021]; Kim and Lee [2019], have explored ways to improve image composition with picture-in-picture, image overlays, edge visualization, or virtual frames. In line with our goal, Yan et al. [2022] addresses the challenge of handheld time-lapse photography using AR guidance. However, their work focused on capture and did not explore reconstruction. In comparison, our work addresses panoramic capture over a very long timescale and deals with 1-2 orders of magnitude more images per scene. This makes capture and registration significantly more challenging.

3 DATA CAPTURE & ORGANIZATION

We explored different capture procedures over the years and found that the best balance of convenience and data quality involved capturing 2D panoramic data with a custom iOS app we designed to make data collection as efficient as possible. The user lines up an initial shot based on an overlay visualizing some reference view of the time-lapse, then they optionally rotate their phone in different directions to capture some number of additional images that can be stitched into a panorama later on (see Figure 2). A single session of capture can be completed in just a few seconds with one hand, making it easy to integrate into regular routines. This speed and convenience was critical to sustaining frequent use over three years of data collection.

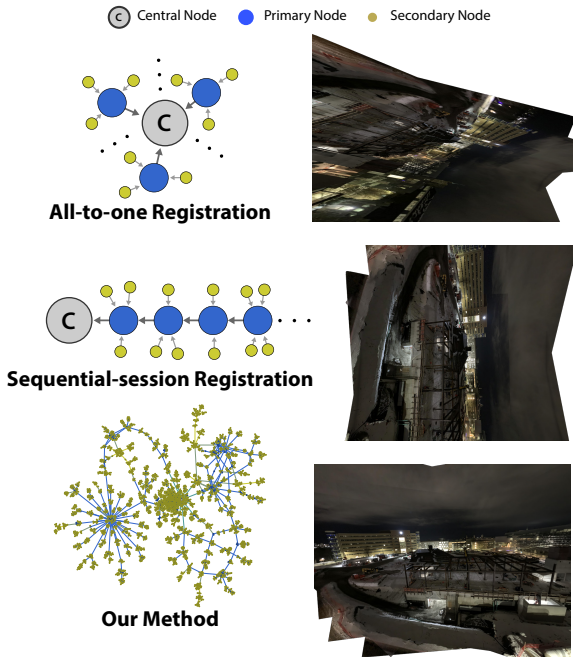

Fig. 3. **Registration Methods: Left:** We show the graph structure of each registration baseline compared to the structure computed by our method. **Right:** Alignments of the same session — captured at night after snowfall, when few of the central node's features could be matched — registered using each approach. All-to-one (top) fails to find accurate feature matches, while sequential-session (middle) has accumulated drift error from the reference image. Our method (bottom) correctly registers the session by chaining high-confidence transformations derived from other images in the dataset.

We call the sequence of images taken between one opening and closing of the application's camera view a ***session***. Our overlay guidance helps ensure that every session's first image covers roughly the same field of view in a scene. We call this first image the ***primary*** sample for that session, and each subsequent image a ***secondary*** image. Section 4 describes a way to derive primary samples for data not captured using this process based on node centrality. Following this pattern is simple, helps ensure data quality, and allows for very efficient computation of feature matches during registration.

A significant advantage of pocket time-lapse over Internet photos is that collecting the data oneself yields more reliable labels. In addition to a timestamp for every image, we use GPS to pull local weather information and calculate the sun's angle during capture. We can use this to, for example, filter for images taken on sunny days, and subsequently train a reconstruction of the scene conditioned on the sun's angle, enabling us to control shadows (see Figure 6).

4 GRAPH BASED REGISTRATION & STITCHING

4.1 Existing Approaches

Our approach to registration is easiest to understand by first examining how existing solutions fail. Notably, we did not set out

to make registration a technical contribution to this work. Instead, we began using existing panoramic registration tools before discovering their limitations as our datasets grew more extensive and varied. For time-lapse, we cannot simply stitch each session into a panorama independently, as this results in a different frame of reference for every session. To create a single time-lapse, we need to register all images across all sessions with a common frame, which is the source of our scaling problem. Software like Adobe Photoshop [Adobe Inc. 2023] offers no way to constrain panorama stitching to a specific image plane, and its automatic stitching pipeline appears to have quadratic complexity, causing it to crash when given about 200 images (far short of the thousands we need). SfM tools like COLMAP [Schönberger and Frahm 2016] are designed to scale for multi-view data but rely on 3D feature triangulation to filter correspondences, causing them to fail on panoramic datasets. We also tried Hugin / Panorama Tools but found it frequently failed across different sessions. The best-performing baselines to converge on our data were two different strategies based on OpenCV. Both strategies use the image plane of the first primary sample as the reference coordinate system for the time-lapse:

- ***All-to-one registration***: All primary samples are aligned with the first primary. All secondary samples are then aligned with their respective aligned primaries.
- ***Sequential-session registration***: We iterate through the sessions chronologically. Each session's primary is registered against the last session to return an alignment successfully. Each secondary is then aligned with its aligned primary.

Each of these strategies has problems. All-to-one registration frequently fails due to insufficient matches between a later session and the reference session, often due to scene or lighting changes. Sequential-session registration suffers from accumulated drift error over many sessions. We introduce a technique derived from analysis of an *alignment graph* built over captured images, which can efficiently and accurately align $> 10k$ images, clearly outperforming existing methods.

4.2 The Alignment Graph

We can represent each of our OpenCV-based strategies as a graph structure, where the alignment of two images corresponds to a path between two nodes (see Figure 3). Our registration strategy builds on this graph interpretation to overcome the problems observed in our baseline approaches. This approach is inspired by graph-based registration strategies used for multi-view data in SfM tools like COLMAP, but we are the first to our knowledge to apply such an approach to scaling panoramic registration.

Given n images $\{I_1, I_2, ...I_n\}$, our task is to compute the transformed images $\{I_1^c, I_2^c, ...I_n^c\}$ expressed in some common coordinate system I_c. For notation, superscripts indicate the coordinate system each image is expressed in, so input images with a matching superscript and subscript are equivalent. For panoramic data, we can represent the transformation $H_i^j : I_i^i \mapsto I_i^j$ as a homography. Our registration method works by first building an ***alignment graph*** with nodes representing input images and edges representing homographies between those images. As shown in Fig. 1, our Alignment Graph spans across a variety of illuminations and seasons.

4.2.1 Graph Construction. We start by running feature tracking and matching in COLMAP. In practice, we customize the matching process for speed (see below), but this customization is optional. The results of geometric verification are stored as a `two_view_geometries` table in the COLMAP database, which we load and decode as the input for graph construction. We begin by adding an edge for every image pair classified as a homography. We then assign a match score to each edge equal to its inlier count divided by the maximum number of inliers across all edges. We treat the reciprocal of this match score as an edge length for further analysis.

4.2.2 Reference Frame Selection & Registration: Registration benefits from selecting a reference frame I_c that matches a large proportion of the dataset well. We estimate this as the node with the lowest average minimal path length to other nodes in the graph, which we calculate by finding the maximum closeness centrality among all nodes [Freeman 1978]. We then register each individual image by finding the shortest path from it to our central node in the graph and taking the ordered product of homographies corresponding to each edge in this path. Fig. 1 visualizes the sub-network of edges that contribute to the registration of nodes in our Garage A dataset. The topology of this sub-network shows how our alignment graph leverages indirect correspondences to reliably register images even when they share little or no reliable matches.

4.2.3 Cutsom Matching. Our registration pipeline does not require labeling primary and secondary images, but having this information can help make registration much faster by employing a strategy analogous to sequential matching for video in Schönberger and Frahm [2016]. We leverage known temporal relationships between images to keep the number of image pairs computed linear in the total number of images rather than quadratic. Firstly, we include all matches between images from the same capture session (across our data, roughly eight samples per session on average). Next, we include matches between each sample and a neighborhood of samples in time: each primary is matched with the primaries of 20 previous and subsequent sessions, and each secondary is matched with all the samples from the last and subsequent four sessions. Finally, we select a uniform subsampling of 25 primary images that are matched with all the primaries. This selection of matches could likely be reduced further, but these settings are already quite effective. For example, on our largest dataset (Balcony 1), they result in computing less than one percent of possible image pairs. Prior to this strategy, we used COLMAP's vocabulary tree-based matching, which starts by indexing input images for a nearest-neighbor search. In our experiments, our custom matching finished in less time than it took to build this index.

4.3 Stitching

We stitch one panorama for each capture session. The main challenges we encounter here are fairly standard: variable exposure and white balance in scenes with high dynamic range and some issues with lens distortion and vignetting. We address these in two ways. The first is by calibrating our camera with a standard radial distortion model and undistorting images prior to processing. The second has to do with how we blend images. Most of the captured

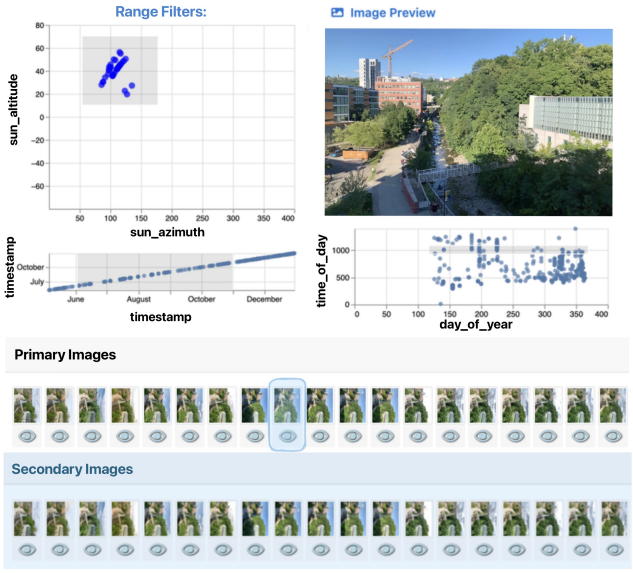

Fig. 4. **Data Exploration Widget:** Our Jupyter Notebook widget allows one to visualize a time-lapse's factors of variation by axes such as sun angle, time of day, and day of the year. It allows users to add custom labels like weather, divide subsets of data.

field of view is seen by pixels from more than one input image, which we can take advantage of for exposure and blending. We add a linear encoding of the transformed images within a session, one at a time, in the order they were captured. Each time we add a new image, we calculate an adjustment for exposure and white balance based on the average per-channel ratio of incoming pixel values to any previously added values that overlap within the output frame. Pixels closer to the optical axis of the lens tend to have less distortion and vignetting. To take advantage of this, we apply a radial falloff from the optical center of each image in RGBA space before adding it to the frame. Once all images have been added, we divide each pixel by its alpha channel to homogenize the resulting image. Doing this ensures that our output image will give greater weight to input pixels with less distortion. As a final step, we splat an extra copy of the primary sample with a substantial radial falloff in alpha using an over operator to minimize the chances of blending artifacts near the center of our panoramic field of view. While it is simple and heuristic, the speed and robustness of our approach are a great strength when stitching tens of thousands of images.

5 EXPLORING CONVENIENCE SAMPLED DATA

To capture frequent data over very long periods of time, pocket time-lapse works best when integrated into a user's daily routines. For example, our Garage A dataset was captured by a user from the garage where they park their car during work hours. As they primarily capture data when arriving or leaving work, most samples are separated by a gap of 8-15 hours. On the other hand, our Baseball Field 1 dataset was captured by a user from a window between their office, a nearby kitchenette, and the bathroom. This capture intensity yielded a much better sampling of different sun angles, as they moved between these places at various times throughout most days. We built a custom interactive Jupyter widget to help visualize and explore such patterns, shown in Figure 4. This interface lets us visualize the distribution of samples along various axes like season and time of day and filter by specific criteria to create a time-lapse focused on specific visual changes.

6 RECONSTRUCTION

The range of visual changes present in multiple years of data can be quite extensive, with many changes happening simultaneously and at very different rates. Our challenge here is to balance temporal filtering against the preservation of well-sampled changes. One solution is to use nearest-neighbor interpolation, which preserves all captured detail at the expense of significant temporal aliasing. Another is to apply a low-pass temporal filter to blur under-sampled phenomena in the scene. We propose a third option, inspired by 3D Gaussian splatting [Kerbl et al. 2023], that improves on these two basic strategies in a few key ways:

- Rather than choosing a single temporal filter per image, we fit local Gaussians over our labels to local parts of the output field of view. This lets us adapt temporal filtering locally to patterns in different parts of a scene.
- We reconstruct our scene over periodic domains of season and sun angle. For subjects that do not undergo significant long-term geometric changes, this lets us combine samples from multiple years to reconstruct daily and seasonal changes. We also use local weather data pulled for each capture session to condition on estimated cloud cover, which strongly impacts the presence of hard shadows.
- We additionally solve for a factorization of the scene into reflectance and shading, which can provide useful information for both reconstruction and analysis.

We call our reconstruction method ***time splatting***.

6.1 Time Splatting

6.1.1 Intrinsic Images. To separate a scene into geometry and shading changes, we turn to the classic approach of intrinsic image decomposition from image sequences [Barrow and Tenenbaum 1978; Weiss 2001]. Given an image at time step t, denoted as $I(t)$, the decomposition can be represented as a per-pixel product of an RGB reflectance $R(t)$ and a single-channel shading image $S(t)$.

$$I(t) = R(t) \cdot S(t) \tag{1}$$

Ideally, $R(t)$ should contain geometry changes, such as falling leaves or construction, while $S(t)$ should contain illumination changes, such as shadows.

6.1.2 Label Conditioning. To better constrain our problem, we condition S on additional labels. First, since the shading image varies with the direction of lighting, we condition it on the sun's angle—represented using azimuth and altitude (θ_t, ϕ_t), derived from the timestamp t and the GPS location. Prior works have demonstrated the usefulness of sun angle for analyzing outdoor scenes [Lalonde et al. 2010; Liu et al. 2020]. Optionally, we can also condition S on an estimated percentage of cloud cover, w_t, which we pull from a local weather API for each session. Our intrinsic image

decomposition is then formulated by the equation:

$$I(t) = R(t) \cdot S(t, \theta_t, \phi_t, w_t). \tag{2}$$

6.1.3 3D Gaussian Splatting (Prior Work). Building on the 3D Gaussian splatting code provided by Kerbl et al. [2023], we proceed by fitting R and S with 2D Gaussians such that their product equals our input image, as shown in Figure 5. For 3D view-synthesis, Kerbl et al. [2023] reconstructs images with a set of 3D Gaussians projected into the synthesized view. Using an over operator, images are reconstructed by alpha-compositing the projected Gaussians according to their depth. The projected Gaussians in their work are described by a mean $\mu_{\mathbf{x}} \in \mathbb{R}^2$, and covariance $\Sigma_{\mathbf{x}} \in \mathbb{R}^{2\times2}$ in image space, with alpha values given in terms of Gaussian G for pixel coordinates $\mathbf{x} \in \mathbb{R}^2$:

$$\Delta\mathbf{x} = \mathbf{x} - \mu_{\mathbf{x}}, \quad G(\Delta\mathbf{x}) = \exp\left(-\frac{1}{2}\Delta\mathbf{x}^T\Sigma_{\mathbf{x}}^{-1}\Delta\mathbf{x}\right). \tag{3}$$

During training, they use gradient-based optimization to fit the Gaussians to input images. Gaussians are split, cloned, and pruned adaptively based on a set of heuristics.

6.1.4 Time Splatting (Our Method). Our method starts by forgoing the initial 3D reconstruction and projection of Kerbl et al. [2023] and directly initializes Gaussians in image space. To extend the domain of these Gaussians to include a set $\mathbf{t} \in \mathbb{R}^d$ of additional labels (including time at a minimum), we redefine each Gaussian as the product of an image-space Gaussian and a Gaussian with one dimension for each newly added label:

$$\Delta\mathbf{x} = \mathbf{x} - \mu_{\mathbf{x}}, \quad \Delta\mathbf{t} = \mathbf{t} - \mu_{\mathbf{t}} \tag{4}$$

$$G(\Delta\mathbf{x}, \Delta\mathbf{t}) = \exp\left(-\frac{1}{2}\Delta\mathbf{x}^T\Sigma_{\mathbf{x}}^{-1}\Delta\mathbf{x}\right)\exp\left(-\frac{1}{2}\Delta\mathbf{t}^T\Sigma_{\mathbf{t}}^{-1}\Delta\mathbf{t}\right). \tag{5}$$

For the reflectance image, we set $\mathbf{t}$ as a normalized scalar representing absolute timestamps $t \in [0, 1]$. For the shading image, we use the label vector $\mathbf{t} = (t, \theta_t, \phi_t, w_t) \in \mathbb{R}^d$. We parameterize the d-dimensional covariance $\Sigma_{\mathbf{t}}$ efficiently using the $\Sigma_{\mathbf{t}} = LDL^T$ decomposition, where L is a unitriangular matrix, and D is a diagonal matrix with positive entries.

6.1.5 Change-Driven Densification. To adapt to varying rates of change in the scene, we initialize the amount of Gaussians in pixel and time space proportional to the rate of change. To address potential label noise, we add normally-distributed noise to the labels during training, with variance proportional to the distances between each sample and its nearest neighbors. This reduces aliasing artifacts from elongated or over-smoothed Gaussians.

6.1.6 Tone Mapping. Finally, similar to Martin-Brualla et al. [2021]; Rückert et al. [2021], to control for changes in auto-exposure and white balance we add an additional per-camera exposure term which maps a camera embedding c_i to a scalar: $L_{exp}(c_i)$, and a three-channel white balance term, which is a function of other labels: $L_{wb}(t, \theta_t, \phi_t, w_t)$. During test time, we set L_{exp} to be the average camera embedding. Our final image is given by the equation:

$$I(t, c_i) = R(t) \cdot S(t, \theta_t, \phi_t, w_t) \cdot L_{wb}(t, \theta_t, \phi_t, w_t) \cdot L_{exp}(c_i). \tag{6}$$

We implement L_{wb} and L_{exp} with small two-layer neural networks and initialize their outputs to 1 before training.

Table 1. **Compute Times for the Entire Registration and Stitching Process.** We recorded the compute times for running our method's registration and stitching process for four of our major datasets. All data was processed on a consumer laptop (M3 MacBook Pro).

Dataset	No. Images	Compute Time
Balcony 1	10140	6 h 47 min
Balcony 2	5570	3 h 8 min
Baseball Field 1	7986	5 h 5 min
Stream 1	791	0 h 28 min

6.1.7 Scene Relighting. The intrinsic decomposition offered by time splatting lets us relight any time-lapse by combining the shading reconstructed for one set of labels with the reflectance reconstructed for others. For example, in Fig 6, our method can generate faithful one-day time lapses as the time changes from sunrise to sunset. As shown in Fig 9, our method can even factor out reflectance and shading from nighttime photos.

6.1.8 Real-Time Rendering. Time splatting can be trained completely on a consumer laptop. After training, a user can interactively explore the time-lapses along each label dimension. On a Dell XPS 15 with a Nvidia GTX 1050Ti Mobile GPU, time splatting takes 20 minutes to train for 5000 iterations and can be rendered at 25-30 FPS.

7 RESULTS

7.1 Registration

Image registration is challenging for pocket time-lapse both because of the number of images and the dramatic changes in appearance over time. As stated in Section 4, we were unable to scale existing registration tools like Photoshop to even our smaller datasets. Instead, we compare our alignment-graph strategy with the two other OpenCV-based matching baselines described in Section 4: all-to-one registration, and sequential-session registration. Each method is quantitatively evaluated for how many sessions it is able to register.

On the Garage A dataset, with all-to-one registration, only 53% of sessions are successfully registered due to significant changes in the scene over time. With sequential-session registration, 98% of sessions register, but these suffer from significant drift error, which causes distortion and a large rotation of the scene over the course of the time-lapse (see supplemental). Our alignment graph strategy successfully registers > 99% of sessions with minimal drift error, leading to significantly better alignment. Visualizing the alignment graph topology in Figure 1, we see that this is accomplished by chaining together high-confidence alignments from non-sequential sessions. For example, night sessions (D and E) are registered with day sessions (C and X) by way of sessions taken at twilight, sunrise or sunset (E and B). This leads to robust and efficient registration even with $10k$ images spanning dramatic changes of appearance. To our knowledge, our approach is the only one capable of successfully registering data of this difficulty and scale.

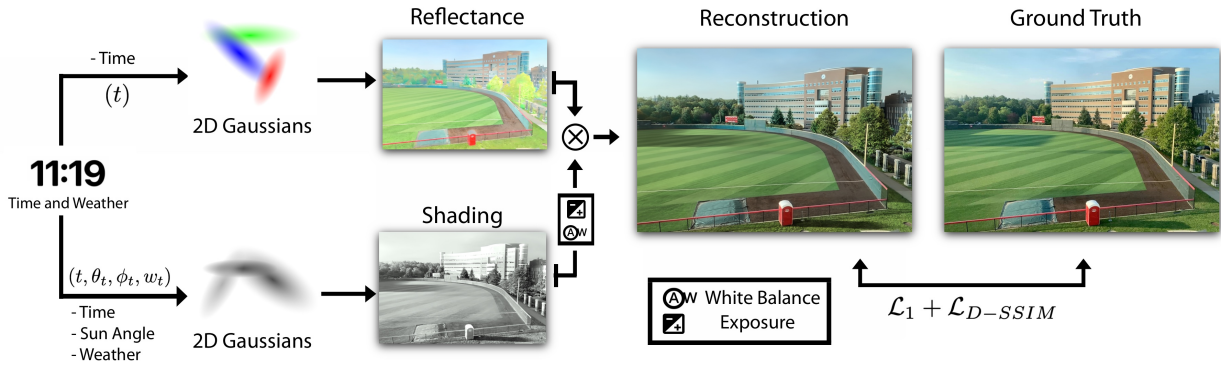

Fig. 5. **Time Splatting:** We reconstruct a time-lapse from sparse samples by fitting a reflectance image and a shading image with 2D Gaussians. Their product is optimized to reconstruct the original samples. To control for changes in appearance over time, we condition the reflectance image on a time step. To control for changes in shading, we condition the shading image on a vector of the time step, sun angle, and cloudiness. Finally, we apply a white balance and exposure adjustment to the predicted (grayscale) shading image to account for differences in auto-exposure and lighting. As shown above, our method is able to contain lighting phenomena such as shadows entirely in the shading image.

Table 2. **Reconstruction Metrics.** We compare the reconstruction quality for unseen test images between time splatting and TLGAN [Härkönen et al. 2022]. Our method outperforms TLGAN across all metrics. Because TLGAN was not designed to interpolate such sparse data, it can often hallucinate, reducing its reconstruction quality. On the other hand, time splatting stays true to training data, as demonstrated by the better reconstruction metrics.

Dataset	**Baseball Field Subset (70 Images)**			**Balcony 1 (815 Images)**		
Metric	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Ours	**20.49**	**0.74**	**0.22**	**18.85**	**0.74**	**0.27**
TLGAN	15.52	0.41	0.36	16.47	0.50	0.31

7.2 Time Splatting

7.2.1 Reconstruction. We first compare how well our time-splatting method can reconstruct unseen images in a time-lapse against TLGAN [Härkönen et al. 2022]. Even when trained on only 70 images, we observe that our method produces realistic reconstructions. TLGAN, on the other hand, as seen in Figure 7, incorrectly hallucinates shadows (top row) and seasons (bottom row), generating reconstructions of winter instead of summer. Highlighted in yellow, TLGAN also synthesizes the construction incorrectly. time splatting reconstructions stay close to ground truth images, accurately preserving the season as well as the positions of shadows. Quantitatively, as seen in Table 2, we outperform TLGAN across all reconstruction metrics. These results hold true for both a small "mini time-lapse" of 70 images and a larger one of 815 images.

7.2.2 Intrinsic Images. A unique property of time splatting is how it can obtain intrinsic images for scene relighting. We compare intrinsic images obtained from time splatting with predictions from Ordinal Shading [Careaga and Aksoy 2023], a state-of-the-art intrinsic image decomposition technique. As seen in the top row of Figure 10, Ordinal Shading incorrectly predicts the reflectance image of nighttime photos. On the other hand, our method succeeds in predicting a sunlight-independent reflectance. Additionally, as seen in the bottom row, our method completely removes shadows from the reflectance image. Contrary to this, the Ordinal Shading approach leaves a strong residue of shadows in the reflectance image.

We tested time splatting on a dozen datasets that each vary in the size and content they present. In Figure 11, we show decompositions of sample images, which demonstrate that our method is able to successfully tackle challenging reconstruction scenarios such as flowing water, heavy foliage, hard shadows, and low light or night photography. Specifically, we can separate most of the illumination information into the shading image and predict a faithful reflectance. Our method can even reconstruct bloom effects, as shown in the Balcony example in Figure 11.

7.3 Limitations

Our work helps reduce the inconvenience of capturing and registering data, but pocket time-lapse still requires prolonged effort to create, and the value of collected data is susceptible to gaps in user behavior. Better support for collaborative capture could help with these challenges.

Time splatting amounts to locally-adaptive temporal filtering, which can help reduce distracting temporal aliasing. This strategy visualizes uncertainty caused by undersampling as blur, which may be useful in some contexts, but is not very visually compelling. The gap in visual fidelity between this approach and one that builds on visual foundation models is likely to grow with time, motivating the exploration of more hybrid approaches that strike different balances between uncertainty and visual fidelity.

8 CONCLUSION

This paper represents the prolonged exploration of two powerful ideas: pocket time-lapse as a type of photography, and convenience sampling as a strategy for realizing that mode of photography. The scale and duration of data, captured on at least 565 unique days, provides a fascinating examination of how pocket time-lapse can integrate with daily routines.

In the process of this exploration, we developed several tools that make these applications more compelling, and we anticipate that

these tools will find many uses in the photography and computer graphics communities. Our alignment graph registration method can be used to efficiently register thousands of time-lapse images, and time splatting allows one to explore time-lapse data in a new way. That being said, recent progress in generative models points to many new opportunities to explore with pocket time-lapse data.

Nonetheless, a major takeaway from our exploration is how pocket time-lapse offers a unique and often surprising picture of the photographer's environment. With this in mind, we are particularly excited to see what others will do with these tools and ideas in the future.

ACKNOWLEDGMENTS

EC is supported by the NSF Graduate Research Fellowship Program. This work was partially supported by a National Science Foundation Faculty Early Career Development Grant under award #2340448 and by a generous gift from Meta.

REFERENCES

Andrew Adams, Natasha Gelfand, and Kari Pulli. 2008. Viewfinder alignment. In *Computer Graphics Forum*, Vol. 27. Wiley Online Library, 597–606. https://doi.org/10.1111/j.1467-8659.2008.01157.x

Adobe Inc. 2023. *Adobe Photoshop.* https://www.adobe.com/products/photoshop.html

Harry Barrow and J. Tenenbaum. 1978. Recovering Intrinsic Scene Characteristics from Images. *Recovering Intrinsic Scene Characteristics from Images* (01 1978).

Chris Careaga and Yağız Aksoy. 2023. Intrinsic Image Decomposition via Ordinal Shading. *ACM Trans. Graph.* (2023).

Weifeng Chen, Jie Wu, Pan Xie, Hefeng Wu, Jiashi Li, Xin Xia, Xuefeng Xiao, and Liang Lin. 2023. Control-A-Video: Controllable Text-to-Video Generation with Diffusion Models. arXiv:2305.13840 [cs.CV]

Partha Das, Sezer Karaoglu, and Theo Gevers. 2022. PIE-Net: Photometric Invariant Edge Guided Network for Intrinsic Image Decomposition. In *IEEE Conf. Comput. Vis. Pattern Recog.*

Jane L. E, Ohad Fried, Jingwan Lu, Jianming Zhang, Radomír Měch, Jose Echevarria, Pat Hanrahan, and James A. Landay. 2020. Adaptive Photographic Composition Guidance. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20)*. ACM, 1–13. https://doi.org/10.1145/3313831.3376635

Jane L. E, Kevin Y. Zhai, Jose Echevarria, Ohad Fried, Pat Hanrahan, and James A. Landay. 2021. Dynamic Guidance for Decluttering Photographic Compositions. In *Proceedings of the 34th Annual ACM Symposium on User Interface Software and Technology (UIST '21)*. ACM. https://doi.org/10.1145/3472749.3474755

Linton C. Freeman. 1978. Centrality in social networks conceptual clarification. *Social Networks* 1, 3 (1978), 215–239. https://doi.org/10.1016/0378-8733(78)90021-7

Michal Geyer, Omer Bar-Tal, Shai Bagon, and Tali Dekel. 2024. TokenFlow: Consistent Diffusion Features for Consistent Video Editing. *ICLR.*

Erik Härkönen, Miika Aittala, Tuomas Kynkäänniemi, Samuli Laine, Timo Aila, and Jaakko Lehtinen. 2022. Disentangling Random and Cyclic Effects in Time-Lapse Sequences. *ACM Trans. Graph.* 41, 4 (2022).

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics* 42, 4 (July 2023). https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/

Minju Kim and Jungjin Lee. 2019. PicMe: Interactive Visual Guidance for Taking Requested Photo Composition. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3290605.3300625

Jean-François Lalonde, Srinivasa G. Narasimhan, and Alexei A. Efros. 2010. What Do the Sun and the Sky Tell Us About the Camera? *International Journal of Computer Vision* 88 (2010), 24–51. https://api.semanticscholar.org/CorpusID:1378771

Zhengqi Li and Noah Snavely. 2018. Learning Intrinsic Image Decomposition from Watching the World. In *Computer Vision and Pattern Recognition (CVPR).*

Haotong Lin, Qianqian Wang, Ruojin Cai, Sida Peng, Hadar Averbuch-Elor, Xiaowei Zhou, and Noah Snavely. 2023. Neural Scene Chronology. In *CVPR.*

Andrew Liu, Shiry Ginosar, Tinghui Zhou, Alexei A. Efros, and Noah Snavely. 2020. Learning to Factorize and Relight a City. In *ECCV.*

Ricardo Martin-Brualla, David Gallup, and Steven M. Seitz. 2015a. 3D Time-lapse Reconstruction from Internet Photos. In *Computer Vision (ICCV), 2015 IEEE International Conference on.*

Ricardo Martin-Brualla, David Gallup, and Steven M. Seitz. 2015b. Time-lapse Mining from Internet Photos. *ACM Trans. Graph.* 34, 4, Article 62 (July 2015), 8 pages. https://doi.org/10.1145/2766903

Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. 2021. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *CVPR.*

Wojciech Matusik, Matthew Loper, and Hanspeter Pfister. 2004. Progressively-Refined Reflectance Functions from natural Illumination. *Rendering Techniques* 1, 2 (2004).

Kevin Matzen and Noah Snavely. 2014. Scene Chronology. In *Proc. European Conf. on Computer Vision.*

Radu Paul Mihail, Scott Workman, Zachary Bessinger, and Nathan Jacobs. 2016. Sky segmentation in the wild: An empirical study. *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)* (2016), 1–6. https://api.semanticscholar.org/CorpusID:11483611

Michael Rubinstein, Ce Liu, Peter Sand, Frédo Durand, and William T. Freeman. 2011. Motion denoising with application to time-lapse photography. *CVPR 2011* (2011), 313–320. https://api.semanticscholar.org/CorpusID:490931

Darius Rückert, Linus Franke, and Marc Stamminger. 2021. ADOP. *ACM Transactions on Graphics (TOG)* 41 (2021), 1 – 14. https://api.semanticscholar.org/CorpusID:238744020

Johannes Lutz Schönberger and Jan-Michael Frahm. 2016. Structure-from-Motion Revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR).*

Kalyan Sunkavalli, Wojciech Matusik, Hanspeter Pfister, and Szymon Rusinkiewicz. 2007. Factored time-lapse video. In *ACM SIGGRAPH 2007 papers*. 101–es.

Y. Weiss. 2001. Deriving intrinsic images from image sequences. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, Vol. 2. 68–75 vol.2. https://doi.org/10.1109/ICCV.2001.937606

Ruyu Yan, Jiatian Sun, Longxiulin Deng, and Abe Davis. 2022. ReCapture: AR-Guided Time-lapse Photography. In *ACM Symposium on User Interface Software and Technology (UIST).* https://doi.org/10.1145/3526113.3545641

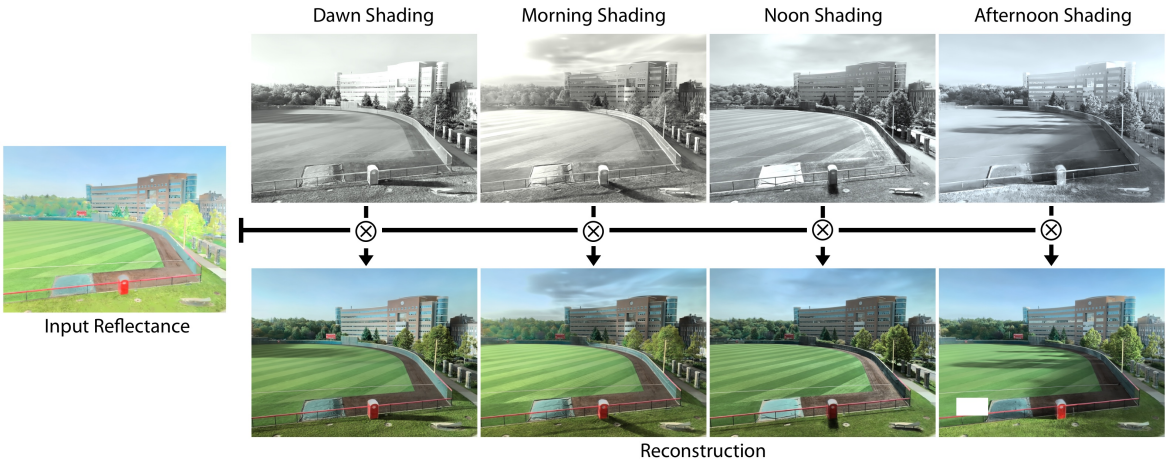

Fig. 6. **Relighting a scene across a single day:** By multiplying a single reflectance image with shading images predicted for various times of day, we can change the shading and shadows of the scene. This demonstrates that most of the shading information is in the shading image instead of the reflectance. Because we control for white balance, we can also synthesize lighting effects like the golden hour, as shown in the Morning Shading example.

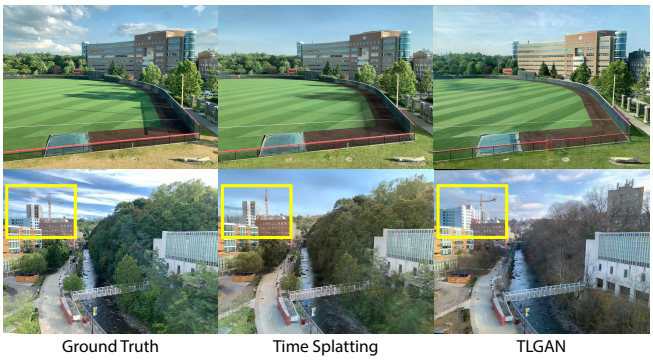

Fig. 7. **Reconstruction Comparison:** Our method is able to closely reconstruct ground truth images from an input time lapse. On the other hand, generative methods such as TLGAN [Härkönen et al. 2022] may hallucinate the training data. In the top example, TLGAN struggles to reconstruct the shadow. In the bottom example, TLGAN synthesizes the wrong season, as shown by trees without leaves. Highlighted in yellow, TLGAN also does not synthesize the construction correctly.

Fig. 8. **Time Splatting for Traditional Time-Lapse:** To demonstrate the applicability of time splatting to traditional time-lapses, we perform intrinsic image decompositions on the SkyFinder [Mihail et al. 2016] dataset. Time splatting separates seasonal information like snow in the reflectance image and lighting effects in the shading image.

Fig. 9. **Panoramic Intrinsic Image Decomposition:** Time splatting decomposes a time-lapse into intrinsic images, allowing one to visualize how reflectance and shading change through time. These results are generated from training on 166 images across 4 months.

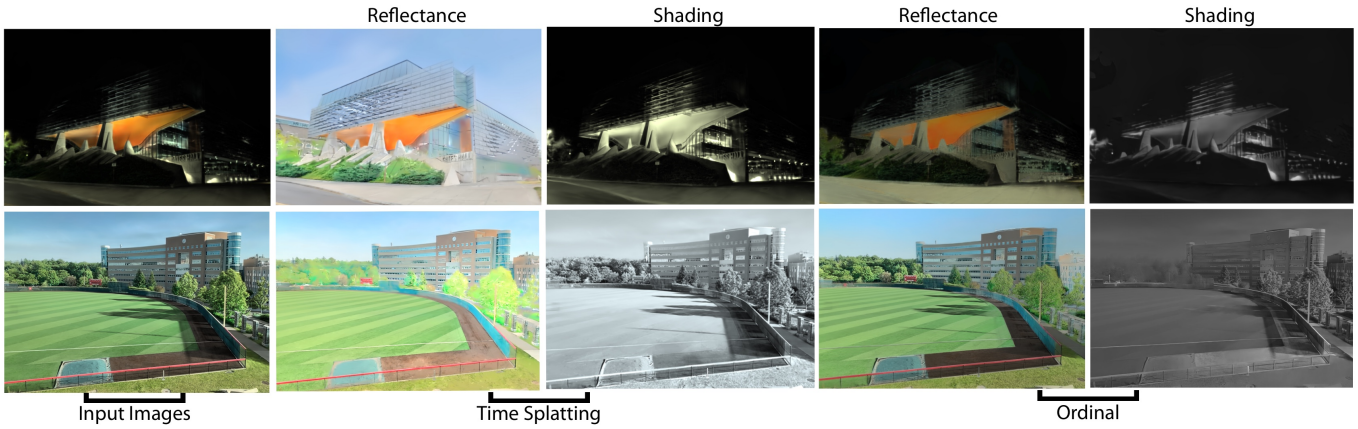

Fig. 10. **Intrinsic Image Decomposition Comparison.** *Top:* Compared to Ordinal Shading [Careaga and Aksoy 2023], a state-of-the-art intrinsic decomposition method, our method faithfully predicts the reflectance of night time photos. *Bottom:* Time splatting also manages to predict a reflectance image with close to no encoded shadow information, which [Careaga and Aksoy 2023] fails to achieve. **Note:** We provide additional comparisons to [Li and Snavely 2018] and [Das et al. 2022] in the supplemental material, both of which our method outperforms.

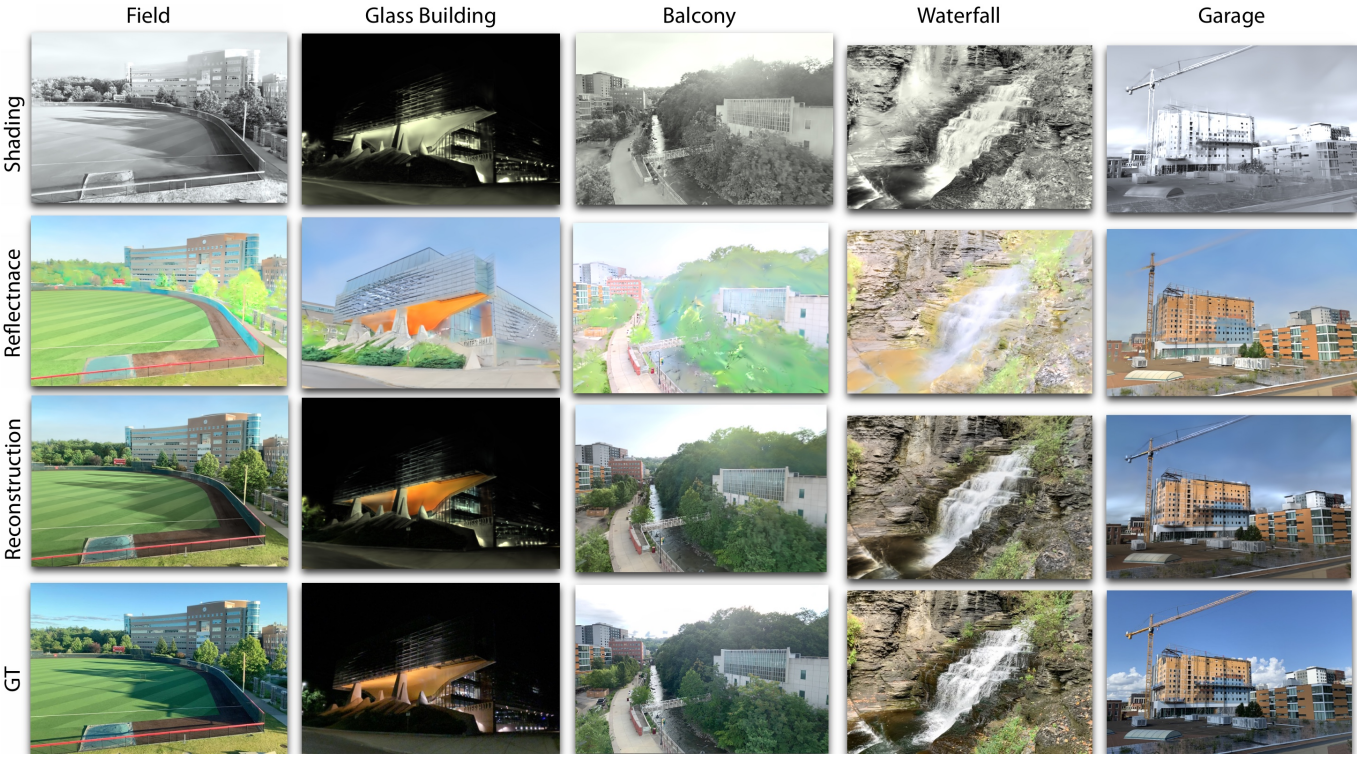

Fig. 11. **Intrinsic Images Across Several Datasets:** We have extensively tested time splatting on a dozen of the collected datasets that vary in size from around 70 images to over 1,000 images. Our results demonstrate that time splatting performs well in a variety of challenging conditions such as shadows, low light, heavy foliage, and running water.

S1 THE POCKET TIME-LAPSE DATASETS

One of our main contributions is presenting a diverse collection of datasets amounting to over 50k time-lapse images. To additionally enrich the datasets, we use our image registration method to combine the primaries and secondaries into panoramic images, which amount to almost 6k photos. Further statistics about the datasets can be seen in Table S3. The datasets were captured by a single individual using their mobile phone, demonstrating how useful everyday activities and habits can be in capturing data of commonly visited scenes. These routines can be used to identify trends in the distribution of data captures, which we can use to visualize various changes and progressions in the data.

The datasets represent diverse scenery ranging from modern glass buildings to balcony views and waterfalls. Each represents unique challenges when attempting to visualize specific trends in the data, such as seasonal changes, construction site progressions, and daily shadow movements. Combined with our time splatting method, which closely reconstructs accurate time-lapse data and can effectively isolate changes in structure and lighting, we use sampling patterns in our captured datasets to isolate and reconstruct smooth and accurate time-lapses of desired trends/changes.

S2 CONVENIENCE SAMPLING

S2.1 "Mini Time-Lapse"

Another less common type of convenience sampling that does not rely on routine is one we informally call "mini time-lapse," where the user captures several sessions during a period where they happen to be near a subject with the intent of seeing what those sessions look like when played back in sequence. This pattern was relatively uncommon for *User A*, with the most notable exceptions found in the Balcony 1 and Balcony 2 datasets, which were captured from the balcony of *User A*'s former apartment. There were a small number of occasions, always on weekends, where *User A* spent a significant amount of time on the balcony engaged in some other activity but

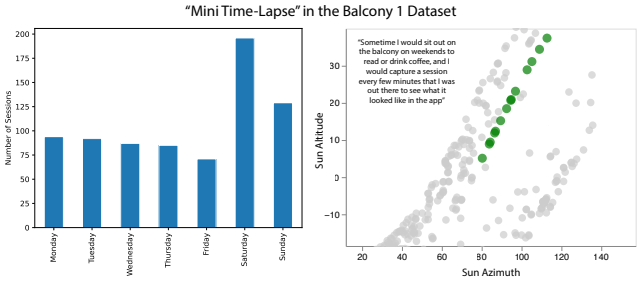

Fig. S12. **"Mini Time-Lapse"** When a user decides to capture a cluster of sessions in a short period with the intent of seeing what they look like played in sequence.

stopping periodically to capture a new session every few minutes. These occasions yielded some of our data's densest and cleanest sampling. Taken as separate subsets and registered with our tool, these image clusters make for a relatively compelling time-lapse even when shown in sequence with no reconstruction. We did not initially intend to capture this type of data, as it felt very similar to that of a traditional time-lapse. However, these short bursts of data yield interesting opportunities for reconstruction when combined with the intrinsic image decompositions we describe in Section 6.

S3 ADDITIONAL RESULTS

S3.1 Panorama Stitching

As stated in the main text, Hugin, an open-source panorama stitching tool, often distorted images or created blurry results. Contrary to Hugin, our method correctly registers the correspondences between the images. Our alignment graph-based panorama stitching method performs significantly better than previous methods under challenging conditions.

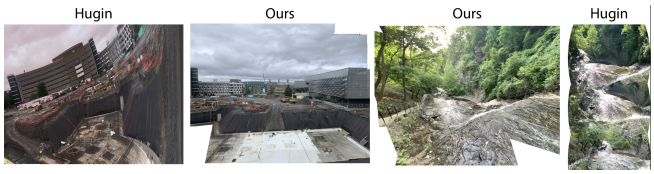

Fig. S13. **Panorama Stitching with Hugin:** Our new alignment graph-based panorama stitching method performs significantly better than previous methods under challenging conditions. Contrary to Hugin, our method correctly registers the correspondences between the images.

S3.2 Traditional Time-Lapses

Time splatting can be applied to not only pocket time-lapses, but traditional time-lapses as well. In Figure 8, we use time splatting to create an intrinsic image decomposition on a scene from the SkyFinder dataset [Mihail et al. 2016]. The reflectance images contain seasonal information like snow, while the shading images contain lighting effects from the sky and lamps. For a video, please refer to the included HTML gallery.

Scene	No. Primaries	No. Secondaries	No. Panos	Total Images	Day Range	Start Date	End Date
Baseball Field 1	1197	6789	1144	7986	665	Mar. 2022	Present
Baseball Field 2	577	3458	577	4035	424	Nov. 2022	Present
Baseball Field 3	618	3933	618	4551	607	May. 2022	Present
Balcony 1	934	9206	934	10140	482	May. 2022	Aug. 2023
Balcony 2	754	4816	752	5570	482	May. 2022	Aug. 2023
Waterfall 1	104	1224	104	1328	597	May. 2022	Present
Garage A	223	2162	529	2385	270	Apr. 2023	Present
Garage E	101	1045	93	1146	185	Jul. 2023	Present
Home Garage	166	1499	166	1665	117	Jun. 2023	Oct. 2023
Glass Building	372	2730	372	3102	556	Jul. 2022	Present
Trail 1	79	1008	79	1087	599	May. 2022	Present
Trail 2	93	997	92	1090	581	May. 2022	Dec. 2023
Trail 3	94	1070	92	1164	581	May. 2022	Dec. 2023
Stream 1	102	689	102	791	605	May. 2022	Present
Stream 2	103	802	103	905	605	May. 2022	Present
Tree Petals	269	1558	268	1827	607	May. 2022	Present
Construction	280	2132	279	2412	521	Apr. 2022	Oct. 2023
Total	6078	45118	5897	51192			

Table S3. **Dataset Statistics:** We plan to share over 50k time-lapse images across diverse subjects. All images were captured by a single person on a mobile phone. Each dataset consists of *primary images*, which are all aligned to the center of a subject, and *secondary images*, which are images taken around the primary images. Using our image registration method, we align the primaries and secondaries into panoramic time-lapses. The majority of the datasets span over one year, allowing one to visualize cycles of seasons.

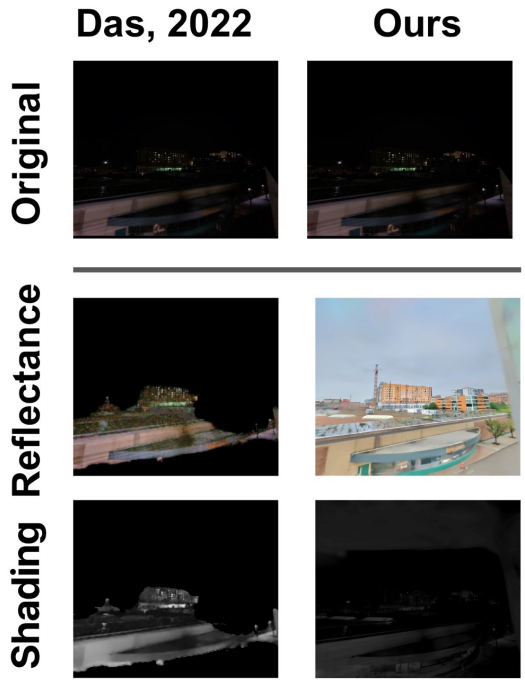

Fig. S14. [Das et al. 2022] also fails to predict faithful reflectance images from night photos. In contrast, our method is able to extract a reflectance image that is essentially the same as the one extracted from a daytime photo.

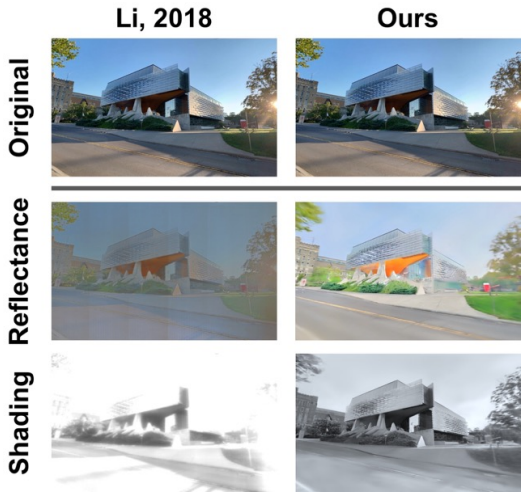

Fig. S15. Our method is able to correctly disentangle shadows from the reflectance image in different lighting and geometry conditions too.

S3.3 More Intrinsic Image Decomposition Baselines

In addition to comparing time splatting to the state-of-the-art Ordinal Shading image decomposition method [Careaga and Aksoy 2023], we provide comparisons with [Li and Snavely 2018] and [Das et al. 2022] on the image decomposition task, two older CNN-based intrinsic image decomposition methods. Neither method can fully disentangle shadows from the shading images nor decompose nighttime photos well.

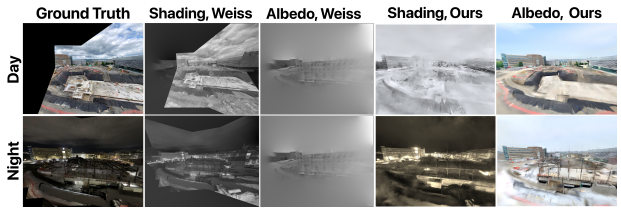

Fig. S18. [Weiss, 2001] is limited to predicting single channel intrinsics and can only predict one albedo for a given sequence of images. On a dataset with high geometric variability it fails to predict meaningful albedos. Our method is able to predict time-varying albedos, avoiding that problem.

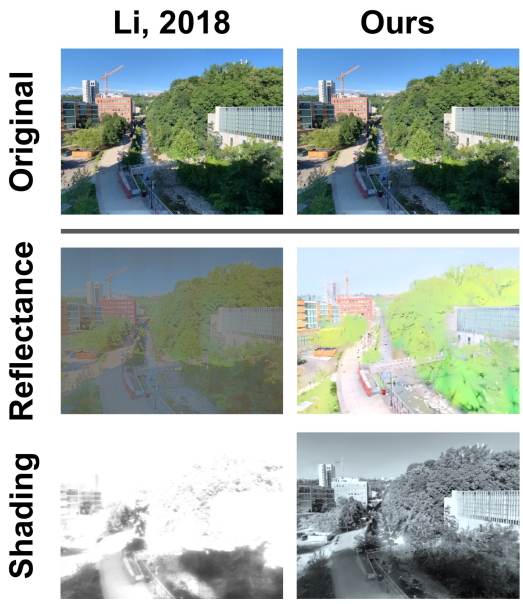

Fig. S16. Our image is able to correctly disentangle even very hard shadows from the reflectance image, whereas [Li and Snavely 2018] does not.

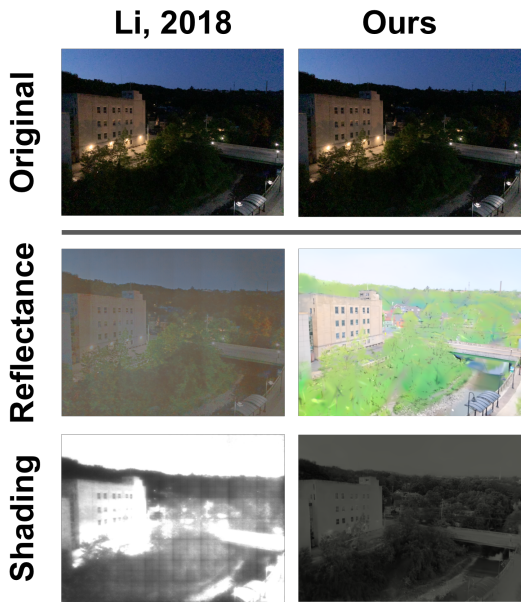

Fig. S17. Our method correctly predicts the reflectance image from a night-time image, whereas [Li and Snavely 2018] is unable to disentangle the dark from the reflectance image.

We also compare our method to a classical baseline, [Weiss 2001], which performs intrinsic image decomposition on image sequences but only performs gray-scale images. The major difference between [Weiss 2001] and our method is that [Weiss 2001] can only predict one albedo image for the entire image sequence, while we can predict albedos that vary in time. This causes [Weiss 2001] to fail for scenes with changing geometry, such as construction.

S3.4 HTML Video Gallery

The supplementary material also includes an HTML gallery of linear interpolations through the time lapses and reconstruction results from time splatting. The linear interpolations are generated by interpolating the panoramas sequentially with a Gaussian weight. The time splatting results demonstrate how we can interpolate across different axes in time, such as through seasons and through sunrise to sunset on a single day.

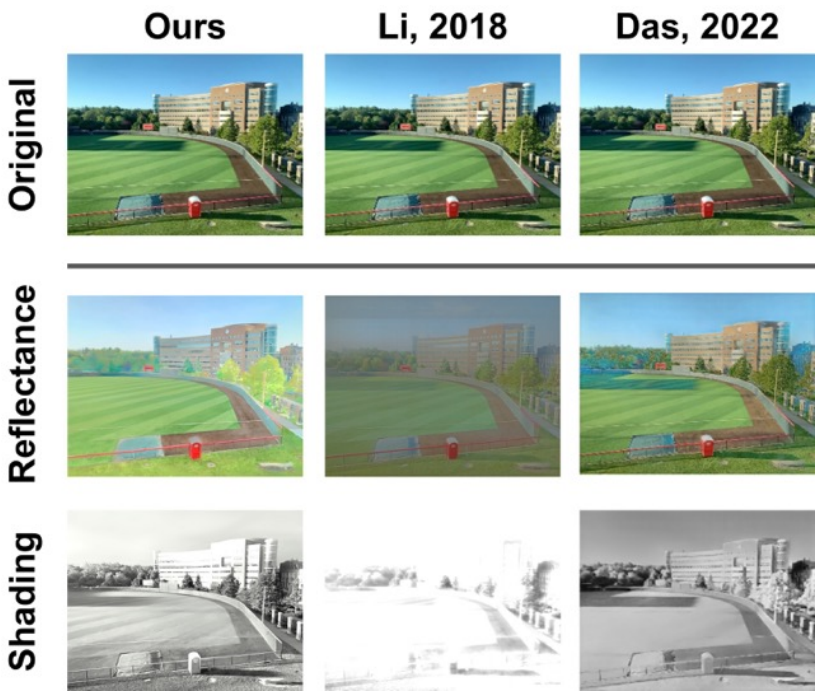

Fig. S19. Our method is able to correctly disentangle shadows from the reflectance image, completely containing it in the shading image. [Li and Snavely 2018] and [Das et al. 2022] fail to disentangle shadows, with the reflectance image containing visible shadows.