ABSTRACT

Time-of-flight point cloud acquisition systems have grown in precision and robustness over the past few years. However, even subtle motion can induce significant distortions due to the long acquisition time. In contrast, there exists sensors that produce depth maps at a higher frame rate, but they suffer from low resolution and accuracy. In this paper, we correct distortions produced by small motions in time-of-flight acquisitions and even output a corrected animated sequence by combining a slow but high-resolution time-of-flight LiDAR system and a fast but low-resolution consumer depth sensor. We cast the problem as a curve-to-volume registration, by seeing a LiDAR point cloud as a curve in a 4-dimensional spacetime and the captured low-resolution depth video as a 4-dimensional spacetime volume. Our approach starts by registering both captured sequences in 4D, in a coarse-to-fine approach. It then computes an optical flow between the low-resolution frames and finally transfers high-resolution details by advecting along the flow. We demonstrate the efficiency of our approach on both synthetic data, on which we can compute registration errors, and real data.

CCS CONCEPTS
• Computing methodologies → 3D imaging; Motion capture; Point-based models;

KEYWORDS
3D Video, Dynamic Point Sets, Detail transfer

1 INTRODUCTION

Capturing accurate 3D geometries is a powerful way for artists to design sceneries, for historians to reconstruct old monuments, for real-estate agents to communicate their products or for navigation systems to provide context. It has become a widespread need, and, when it comes to static environments, is now mostly successfully performed using laser technologies such as LiDaR, that capture environments at sub-millimeter accuracy. When it comes to slightly moving, let alone fully animated scenes, this technology breaks. In fact, capturing a single frame can take tens of seconds, which makes any motion problematic. Even small motions manifest as distortions (Fig. 1) altering the reconstructed point cloud. For dynamic scenes, one often resorts to much less accurate systems such as infrared sensors (Microsoft Kinect or Creative Senz3D), or structure-from-motion using multiple video cameras. These systems allow for capturing rough depth of an entire field of view at 30-60 frames per second, albeit at low resolution, with an accuracy of centimeters and numerous outliers. These setups gained popularity with interactive console 3D games for which neither precision nor accuracy is crucial. For accurate dynamic scene reconstruction, no satisfactory solutions exist and one often resorts to high-resolution templates deformed to match rough motions. This
is particularly the case for facial animation, but does not generalize well to different geometries.

In this paper, we provide the insight that using both laser and infrared technologies at the same time, one can undistort and partly recover a more accurate geometry in the presence of moderate motions. We design a method that registers an accurate LiDAR point cloud captured at a low temporal framerate to a coarse spatio-temporal depth-sensor point cloud captured at interactive framerate, without any preliminary device cross-calibration. We formulate this problem as a spacetime curve-to-volume rigid correspondence problem efficiently solved using a Hough transform and Iterative Closest Point algorithm. Our intuition is that, due to the capture time, a LiDAR point cloud can be seen as a curve in a 4-dimensional spacetime, in which points are each identified by a single \((x, y, z, t)\) value, while a real-time depth video camera produces \((x, y, z)\) dense slices for individual timestamps at a high frame-rate, resulting in a sliced 4-dimensional spacetime volume. We then transfer details from the LiDAR to the Kinect point cloud and advect them across frames, leading to a detailed animated model. We validate our method on synthetic data, and demonstrate it by recovering high-resolution dynamic geometries under moderate motion.

2 RELATED WORK

*Acquisition of high resolution dynamic shapes.* has been tackled using stereo and active light projection systems [Konincx and Gool 2006], such as fringe projection and time shifting [Zhang 2010]. In the special case of facial motion capture, Zhang et al. [2004] propose to use synchronized video cameras and structured light projector and fit a highly detailed template to the resulting geometry to get a high resolution facial animation. Bradley et al. [2010] avoids templates by using a high-resolution multi-camera setup to reconstruct detailed facial geometries. Weise et al. [Weise et al. 2011] use a consumer depth sensor to animate a face template. A combination of active light and stereo was also proposed for capturing scenes in real time with motion compensation [Weise et al. 2007]. More generally, high resolution spacetime capture can be performed using multiview stereo techniques [Davis et al. 2005; Zhang et al. 2003]. Yet the resolution is often limited, depending on the number of views and size of captured objects. Sensors also need synchronization and often, heavy calibration, which can make them difficult to use in practice.

*Static point set super-resolution.* has been tackled by Kil et al. [2006] where several nearby scans are registered and merged together to obtain a high resolution point cloud. More recently, Hamdi-Cherif et al. [2018] nonlocally merge self similar patches of a LiDAR point scan to improve its resolution. In a quite different setting, Haefner et al. [2018] proposed to perform single frame super-resolution from a Kinect scan by using shape from shading to solve this ill-posed problem.

*Texture synthesis and transfer.* High resolution and detailed animations synthesis is a hot topic in computer generated animation research. Rohmer et al. [2010] generate detailed wrinkles on an animated mesh to make it look more realistic, Bertiken et al. [2017] propose a way to transfer details from similar areas of one shape to another, using metric learning.

3 OVERVIEW

As input, our method takes two 3D point clouds of the same dynamic scene, under small to moderate motion: a set of low-resolution point clouds obtained at 30fps from a structured-light infrared depth sensor (such as Kinect) during the motion, and a single accurate but distorted time-of-flight laser LiDAR point cloud taken during the same period of time. While the former provides a low-resolution point set at regular time intervals, the latter provides a highly accurate point set but at a single time stamp \(t\) for each point. We will refer to the structured-light frames as LR frames, and similarly, to the time-of-flight data as HR data. In practice, both setups capture depth values of the scene with respect to the device. Due to motion in the captured scene, the HR point cloud appears distorted (Figure 1) but each point is precisely captured. The LR frames do not suffer from such time-distortion but exhibit a poor quality: spatially inaccurate and quantized depth values with large noise at a low 0.3 mega-pixel resolution. Our core idea is thus to un-distort the HR data that is accurate in space based on the motion captured by the LR data that is accurate in time. The process if summarized in Figure 2.

Our goal is to resample HR points in time to obtain a high-resolution point set for each time frame. This is achieved through three steps. First, we estimate a motion field between the depth sensor LR frames using an off-the-shelf RGB-D optical flow technique. Second, we observe that, up to missing data and noise, the HR point cloud can be exactly registered to the LR data via a rigid transform. In the 4D spacetime continuum, we see the LR frames as a set of 3D spatial “slices” taken at regular time intervals, while the HR point cloud is seen as a time-parameterized curve in the 4D volume as each captured 3D point corresponds to a unique time stamp (Figure 2). This registration step hence amounts to finding a curve pattern within a 4D volume. We robustly perform this step using a global Iterative Closest Point algorithm initialized via an adaptation of a coarse generalized Hough transform. Finally, we...
use the motion field to advect details from the registered HR data across depth sensor frames.

4 SCENE FLOW ESTIMATION

To be able to transfer details from the time-of-flight HR acquisition to the LR depth video frames, it is necessary to track 3D points of the LR depth frames through time. A solution to this problem would be to work solely on the depth information and use an algorithm to register Dynamic Point Sets without computing point to point correspondences [Mitra et al. 2007]. However, in our case, the depth sensor provides color information which, if properly registered to the depth frames, gives valuable information. The flow obtained from RGBD images is called scene flow. It is a variant of the more common optical flow between color images (e.g. [Fleet and Weiss 2006]), with the additional challenge that the flow should also account for depth information, and not only the color. In recent years, several approaches have been proposed to solve this problem (e.g. [Herbst et al. 2013; Huguet and Devernay 2007; Quiroga et al. 2014]). We use the approach of Quiroga et al. [2014] that extends variational optical flow estimation from color image sequences to RGBD videos. This method favors a piecewise smooth scene flow by modeling motions as twists and introducing a total variation regularization.

The computed scene flow provides a way to track a point across all frames until the end of the sequence or until it becomes occluded. It will later be used to advect details from the HR dataset.

5 REGISTRATION IN THE 4D SPACE

In the 4-dimensional spacetime volume, the LR data provides a set of regularly spaced 3D hyperplanes while HR data provides a single curve parameterized by the time \( t \) (see Section 3). We adopt a two-step procedure to align the curve to the 3D hyperplanes. We see this operation as the problem of searching for a pattern within a point cloud. We initialize this search using a coarse, discretized, Hough transform, which we fine-tune in a second step using an ICP. Because of efficiency concerns, we first register the data in the 3D space only, disregarding the time component completely, where we interpret the two point clouds as their projections along the time axis to the 3D space. Then, we perform both the coarse and fine registration steps again, this time on 4-dimensional point sets. This section describes these steps in more details.

5.1 Problem formulation

If both the LR and HR sensors are located at the same place, share the exact same field of view and the capture starts at the same time \( t_0 \), then all captured points are completely aligned in spacetime and share the same spacetime coordinate frame. However, this is never the case as both cameras capture different portions of the scene and are hard to synchronize. In addition, the raw HR data do not directly include a time-stamp for each captured 3D point, instead we roughly estimate the time-stamp using the total acquisition time and the scanline pattern of the acquisition. The first operation we perform is thus a registration procedure that brings both datasets to the same space. This amounts to estimating a rotation and translation in space, together with some translation and possible scale in time, to match one dataset to the other. Moreover, due to the inaccuracy of the LR frames, adjusting a scale in space is also necessary for a good alignment. The whole registration thus corresponds to the search for a single global 3D spatial rotation \( R = (\theta_x, \theta_y, \theta_z) \), a 4D spacetime translation \( T = (T_s, t_t) \) and a 4D spacetime scale \( s = (s_s, s_t) \).

We parameterize \( R \) by three Euler angles, the translation \( T_s \) by three coordinates, and \( t_t, s_s \) and \( s_t \) are three scalars. This registration procedure amounts to estimating 9 parameters. We will denote the entire 9-d transformation \( T \). Denoting \( H \) the HR point set (resp. \( L \) the low-resolution structured light data) in 4D, we formulate the registration problem as the minimization:

\[
\min_T \sum_p ||T_p - q||^2,
\]

where \( p \in H \) and \( q \in L \) is the closest LR point to \( p \).

5.2 Coarse initialization

Rough estimates of these parameters can be obtained using a voting scheme akin to the generalized Hough transform traditionally used for detecting shapes in images, by discretizing a well-chosen parameter space, and computing the scores of each set of parameters. Historically, the Hough transform was first introduced to detect straight lines on images by discretizing line parameters and scoring them [Duda and Hart 1972]. It was later extended to more general shapes by discretizing a space of shape template transforms [Ballard 1981].
In our case, we detect our HR curve on the LR volume by discretizing the 9-dimensional space of transformation parameters described in Sec. 5.1. However, given the sheer amount of data and the curse of dimensionality affecting our 9-dimensional space, some adaptations are needed.

First, it’s easy to see that by placing both the consumer depth sensor and the LiDaR system in an upright position facing the scene \( \theta_x \) and \( \theta_y \) become negligibly small for the purpose of the coarse registration step, so they can be safely omitted. Then, scaling in space is only included to make up for the inaccuracy of the LR data and thus it should be close enough to 1 not to alter the results of the Hough Transform applied at such a low resolution. Considering these observations the 9-dimensional parameter space \( \mathbb{P} \) is first safely reduced to a 6-dimensional space \( \mathbb{P}' \).

As mentioned previously, the registration procedure is divided into an initial phase computing only the transformation \( T_x \) in space and a final one computing the translation \( t_s \) and scale \( s_t \) in time. In the context of the Hough Transform this means that \( \mathbb{P}' \) can be further divided into two separate parameter spaces \( \mathbb{P}_x(\theta_y, t_x, t_y, t_z) \) and \( \mathbb{P}_t(t_s, t_f) \) which are discretized at a given resolution.

For each set of parameters of the form \( T_x = (\theta_y, t_x, t_y, t_z) \) and \( T_t = (t_s, t_f) \), corresponding to a bin in \( \mathbb{P}_x \) and \( \mathbb{P}_t \) respectively, we compute a score which represents how well the transformed point cloud \( T_x H \) in space and \( T_t H \) in 4D spacetime are registered to \( L \) and seek to maximize it. When working on \( \mathbb{P}_x \), the LR data is converted, at a given rough resolution \( L \), to a voxel grid \( V_L \) with each voxel \( v \) storing the number of points lying in it. Then the HR data is transformed using \( T_x \) and discretized into a grid \( V_{T_x H} \) using the same resolution as \( L \). The score of \( T_x \) is computed as \( \sum_v \min(V_L(v), V_{T_x H}(v)) \). The final solution is found as the transformation \( T_x \) with the highest score.

In the case of time registration, 4-dimensional voxels would not allow a high enough resolution neither in the parameter space \( \mathbb{P}_t \) nor for the voxel box. Our alternative solution is to search for a transformation \( T_t \) that minimizes the point-wise distance between \( T_t H \) and \( L \).

### 5.3 Fine registration

The solution of the coarse registration step is only known up to the precision of the parameter space discretization. This is insufficient as we aim at transferring millimeter-scale details. Hence, in a second step, we refine the rough estimate both in the 3D case first and then in the 4D spacetime. As the coarse solution is assumed to be close to the optimal solution, we can now resort to a local optimization, namely an ICP [Besl and McKay 1992], to refine \( T \). Let \( H' = T H \), the new transformation \( T' \) is found by iterating the classical two steps: 1) assign to each point \( p_i \) in \( H' \) its closest point \( q_i \) in \( L \) (if no point is found closer to a given threshold then the point is simply omitted). 2) Find the transform \( T' \) minimizing:

\[
\sum_i ||T' p_i - q_i||^2.
\]

which is solved using Kabsch algorithm for the translation in 4D and rotation in space. The scale in time is solved by computing the standard deviation of the time stamps of the matched points \( \sigma_p \) and \( \sigma_q \) of \( H' \) and \( L \) and deriving the scale as \( s_t = \frac{\sigma_q}{\sigma_p} \). Conversely, the scale in space is computed by averaging the ratios of the 3D Euclidean distances between HR point pairs and their LR counterparts as follows:

\[
s_t = \frac{1}{|H'|^2} \sum_{i,j} \frac{||p_i - p_j||}{||q_i - q_j||}.
\]

At the end of the iterations the HR points \( p_i \) with no sufficiently close LR counterpart \( q_i \) are considered occluded from the point of view of the LR sensor and are discarded.

The result of the spacetime registration can be seen in Figure 3: for different time values \( t \), we display the LR and HR points that lie in a small temporal neighborhood around \( t \), showing that our spacetime registration matches well the datasets despite the difference in resolution.

### 6 DETAIL TRANSFER

Now that both the high and low resolution point clouds share the same spacetime coordinate frame, the last step of our algorithm advects the HR point cloud to enrich all frames of the LR point cloud. To do so, we rely on our estimate of the scene flow between consecutive LR frames (see Section 4), and advect HR points accordingly.

Let us consider a point \( p \in H \) and \( F^t \) its closest LR frame in time. A search in the 3D space finds \( Q^t_p \subset F^t \) as the set of its nearest neighbors in frame \( t \). The motion estimated at each point \( q_i \in Q^t_p \) is then interpolated in space and time to bring \( p \) to the exact timestamp \( t \) of \( F^t \). Next \( p \) is advanced by one frame in time based on the scene flow of the points \( Q^t_p \). This process is performed iteratively until no sufficiently close nearest neighbor can be found which means \( p \) is occluded.

### 7 RESULTS

We validate our approach using two datasets. First, using synthetic data, we make sure our method allows to transfer details with sufficient accuracy, and evaluate any reconstruction error. Second, we showcase our method on real data and show it to be of sufficient accuracy to be used to undistort small to moderate motions.
7.1 Simulated Data

Our simulated data consists of a single character undergoing a rigid transformation \( T = (T_s, R_s) \), raytraced from different viewpoints using parameters similar to LR and HR devices. This synthetic data allows us to synthesize a distortion-free dynamic point set, serving as a groundtruth, and compare our result to it.

To simulate the noise introduced by the commercial depth sensor we generated depth-bins increasing in size proportionally to the distance from the camera mimicking the quantization errors introduced by the sensor and matched the computed depth values to the depth-bins. However we do not simulate any depth sensor calibration error.

Table 1 shows the accuracy of the different steps of our method. By using the known values of the parameters of the 4D registration and the precomputed motion field we assess separately the noise introduced only by the detail transfer. On the contrary, by using no prior knowledge of the scene the average distances indicate the accumulated error introduced throughout the steps of our method.

Table 1: Average point-wise distances between reconstructed and generated ground truth point clouds using either known (GT) registration parameters and motion field or computing them using our method (E). All errors are below 2% of the shape height (2.5 units)

<table>
<thead>
<tr>
<th>Used methods</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>GT registration and GT motion field (1)</td>
<td>0.01271</td>
</tr>
<tr>
<td>E registration and GT motion field (2)</td>
<td>0.01463</td>
</tr>
<tr>
<td>GT registration and E motion field (3)</td>
<td>0.04368</td>
</tr>
<tr>
<td>E registration and E motion field (4)</td>
<td>0.04479</td>
</tr>
</tbody>
</table>

7.2 Real Data

We now turn to the more difficult case of real data. Our experimental setup is the following. A LiDaR and an ASUS Xtion sensor acquire the same animated scene yielding respectively a high resolution depth data (corresponding to \( H \)) and a low-resolution dense depth video (corresponding to \( L \)). Our system does not require any manual calibration: both acquisition systems are only roughly synchronized and we only need them to start roughly at the same time (in practice, the Asus depth sensor capture often starts several seconds before the LiDaR as the LiDaR performs an automatic self-calibration procedure at the start of each capture). A picture of the acquisition system is shown on Figure 5.

**Figure 5: Our experimental setup: a Kinect and a LiDaR acquire the same scene from different viewpoints. No calibration is required.**

The Xtion captor captures 30 depth frames per second with a resolution of \( 640 \times 480 \), the capture is performed using the OpenNi2 library [organization 2017], removing distortion and yielding the final 4D point cloud in millimeters and seconds. In practice, we observed an accuracy of roughly \( \pm 5 \text{cm} \) at 1.5m, which corresponds to our capture distance. To compensate for noisy depth values in the LR sequence, we filter depth values using a bilateral filter with standard deviations in space and values: \( \sigma_s = 1.16 \) and \( \sigma_v = 64 \).

The LiDaR scanner is a FARO Laser Scanner Focus\(^5\) X 330. It was set to capture one point every 50mm at a 2m distance. A laser ray is emitted and reflected by a mirror that directs the beam towards the scene and controls the angle of the ray. By rotating around its axis, this mirror allows for a complete rotation of the laser ray, however it only measures the time of flight on a given angular range.
Figure 6: Undistorting LiDAR scans (top rows, left) from consumer depth camera sequences (top rows, right). Our result (bottom rows) show higher spatial accuracy than the consumer depth camera while respecting the global motion. Video results can be seen in supplemental materials.

<table>
<thead>
<tr>
<th>LiDAR scan</th>
<th>frame A</th>
<th>frame B</th>
<th>frame C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>Reconstruction</td>
<td>Input</td>
<td>Reconstruction</td>
</tr>
<tr>
<td>Input</td>
<td>Reconstruction</td>
<td>Input</td>
<td>Reconstruction</td>
</tr>
<tr>
<td>Input</td>
<td>Reconstruction</td>
<td>Input</td>
<td>Reconstruction</td>
</tr>
</tbody>
</table>

(Continued...)


