2009-12-01

GPU-Accelerated Hierarchical Dense Correspondence for Real-Time Aerial Video Processing

Stephen Cluff
cluff@byu.edu

Bryan S. Morse
morse@byu.edu

Jonathan D. Cohen

Mark Duchaineau

Original Publication Citation

S. Cluff, B. Morse, M. Duchaineau, and J. Cohen, "GPU-accelerated hierarchical dense correspondence for real-time aerial video processing," in IEEE Workshop on Motion and Video Computing (WMVC), December 29.

GPU-Accelerated Hierarchical Dense Correspondence for Real-Time Aerial Video Processing

Stephen Cluff, Bryan S. Morse
Brigham Young University
Provo, UT 84602

Mark Duchaineau, Jonathan D. Cohen
Lawrence Livermore National Laboratory
Livermore, CA 94550

Abstract

Video from aerial surveillance can provide a rich source of data for many applications and can be enhanced for display and analysis through such methods as mosaic construction, super-resolution, and mover detection. All of these methods require accurate frame-to-frame registration, which for live use must be performed in real time. In many situations, scene parallax may make alignment using global transformations impossible or error-prone, limiting the performance of subsequent processing and applications. For these cases, dense (per-pixel) correspondence is required, but this can be computationally prohibitive. This paper presents a hierarchical dense correspondence *algorithm designed for implementation on graphics processing units (GPUs). Since the method does not rely on epipolar geometry, it is also suitable for use when there are uncorrected nonlinear lens distortions. A framework for using this dense correspondence to implement local mosaicking, super-resolution enhancement, and mover detection is also presented and demonstrated using examples of each of these types of enhancement and different types of video sources.*

1. Introduction

Over the last decade, video surveillance from Unmanned Aerial Vehicles (UAVs) has become increasingly feasible. UAV-based systems come in a range of sizes and prices. The more expensive systems, costing hundreds of thousands to millions of dollars, include multiple high-resolution cameras, GPS and other sensors, on-board data processing and storage, and state-of-the art transmission to ground. However, the high cost of such systems prohibits their widespread adoption for many important applications. For such applications 5–8 foot miniature UAVs (mini-UAVs), typically in the ten to fifty thousand dollar price range, may be more realistically and widely deployed.

Unfortunately, video from mini-UAVs suffers from a number of problems. Because of their small size, mini-UAVs are easily buffeted by the wind, making the video unstable and hard to watch. Providing sufficient resolution of objects on the ground results in a limited field of view, creating a "soda-straw" effect similar to that described in [5]. This makes it difficult for an observer to detect and identify objects and landmarks, let alone build a mental model of the enduring scene. Finally, the resolution of the video is restricted by both the limited payload capacity of the craft and the available bandwidth for transmission to ground.

A number of well-established vision techniques can be helpful in addressing these limitations. *Video mosaics*, even if created only for a relatively small number of frames (less than 200) can enhance the spatiotemporal display of the video and have been shown to improve detection of targets of interest [23]. Multi-frame *super-resolution* can be used to enhance the resolution of the aerial video while still allowing efficient wider-area coverage. For some applications, *mover detection* can also be used to identify targets of interest, but doing so requires separating intrinsic scene motion from that of the camera (egomotion).

All of these methods require accurate frame-to-frame registration, which for many aerial video applications must be done in real time. Real-time mosaicking can be accomplished in fairly straightforward ways using global transformations. However, for accurate super-resolution and mover detection when there is even modest parallax, dense (all-pixel) correspondence may be required in order to achieve acceptable results.

This paper presents a *hierarchical dense correspondence* (HDC) algorithm designed for implementation on graphics processing units (GPUs) and a framework for using this dense correspondence to implement local mosaicking, super-resolution enhancement, and mover detection. The use of dense correspondence between both adjacent and non-adjacent image pairs allows robust application even when there is significant parallax in the sequence. Furthermore, we can operate directly from uncalibrated cameras, even when there is significant uncorrected radial distortion from using wider-angle lenses to cover larger target areas. This achieves effective results even in cases where global transformation fitting may be difficult or error-prone.

We demonstrate our HDC-based algorithms on image and video data acquired in the field, including both mini-UAV video footage and small sequences acquired with an SLR in burst mode.

978-1-4244-5501-0/09/$25.00 ©2009 IEEE

2. Related Work

Many researchers have used aerial video for various applications (e.g., [8, 9, 18–20, 23, 31]). Some of these methods have been accelerated for real-time use using either custom hardware (e.g. [19]) or more recently GPUs (e.g. [32]). As is the case with our visualization framework, many of these applications rely on accurate registration of successive video frames, and in this work we focus on three applications of the HDC for image alignment: local mosaic construction, super resolution, and mover detection.

The creation of panoramas or mosaics from collections of images, including video, has been a well-studied area for more than 15 years [22, 24, 26, 27, 29]. (For an excellent survey of this broad area we recommend [28].) Mosaics have been used extensively as a means for summarizing aerial video sequences in ways that provide broader spatial context than individual frames [15, 18, 19]. They may also be used as a component in a larger strategy for visualizing video content [6, 11].

Once spatially aligned as with a mosaic, multiple video frames can also used to provide super-resolution, or greater resolution than a single source frame alone provides. Although there are a variety of approaches for aligning and warping the source images, modeling their respective point spread functions, and blending them [1, 2, 4, 7, 10, 12, 13, 16, 17, 25], a key component in all methods is likewise the accurate alignment of image content. Since most mosaicking approaches use a global transformation between the images, super-resolution from mosaics can often be limited by local deviations from the global transformation, typically the result of parallax or intrinsic motion in the scene. The HDC algorithm presented here, because it allows arbitrary per-pixel motions different from the general global transformation, can accurately align frames on a per-pixel basis and super-resolve the images even in the presence of parallax.

This per-pixel registration of images also facilitates the identification of intrinsic motion in the scene by separating out the frame-to-frame differences due to camera motion [3]. While the HDC algorithm allows minor deviations in motion due to parallax, larger motions can be detected and isolated to identify movement.

3. System Overview

The system in this paper relies on aligning sequences of video frames V in order to enhance them. Given a temporal window of $2W+1$ frames, we align frames $\{V_{t-W}, \ldots, V_t, \ldots, V_{t+W}\}$ with a designated *anchor frame* V_t.

Our system employs a combination of two types of $2D$ transformations: hierarchical dense correspondences and either perspective or affine transformations derived from these (see Figure 1). The dense correspondences produce pixel-level mappings between two images, often at sub-pixel accuracy. Without any direct computation of $3D$ depth or camera parameters, they can account for effects such as change of perspective and motion parallax. Perspective or affine transformations, on the other hand, provide a single, global mapping between two images that is compact to store and simple to compose to describe the transformation along a sequence of images. A useful tool in our approach is to compute the global transform that is the best least-squares fit to a given dense correspondence mapping.

Figure 2 illustrates the transformations that need to be computed to create a mosaic and enable both super-resolution and mover detection. HDC is used to compute the dense correspondence $D_{i\mapsto i+1}$ between frames V_i and V_{i+1}. This, in turn, is fit with affine approximation $A_{i\mapsto i+1}$. Each time our temporal window advances by a frame, we perform a single HDC and a single affine fit to it in order to incorporate the new frame into the sequence. Composing $A_{t\mapsto i}$ with $A_{i\mapsto i+1}$ produces $A_{t\mapsto i+1}$. These composited transformations may then, as with other mosaicking methods, be used to combined the new frame with other frames in a mosaic, as described in more detail in Section 5.1. Since we are continually updating the computations as new frames arrive, we do not use a global bundle adjustment [30].

Multi-frame mover detection and super-resolution require somewhat more computation. In particular, each requires a dense correspondence $D_{i\mapsto t}$ between each frame contributing to the detection or resolution enhancement and the anchor frame V_t. Although it may seem conceptually possible to compose dense correspondences in a manner similar to the affine transforms, there are two noteworthy obstacles. First, we wish to avoid any procedure that repeatedly resamples the images. Second, the correspondences are not easily composed without clipping to the boundaries of each successive image as the composition proceeds. Instead, we retain the previously described pipeline but use each affine estimate $A_{i\mapsto t}$ to seed the HDC algorithm for the computation of $D_{i\mapsto t}$. Having a good seed for the HDC computation can be important as source and target are farther apart in the sequence or in the spatial domain.

4. Hierarchical Dense Image Correspondence

The fundamental building block of our system is an efficient and robust algorithm for computing dense (per-pixel) correspondences between a source and a destination image. As described in Section 3, this enables the mapping of a set of images to the domain of a selected anchor image, and provides the ability to perform super-resolution image enhancement and background/mover segmentation.

As with other dense correspondence algorithms, such as optical flow and stereo matching, the HDC algorithm has a matching component that produces a motion field, and a regularization component that smooths the warp. One key

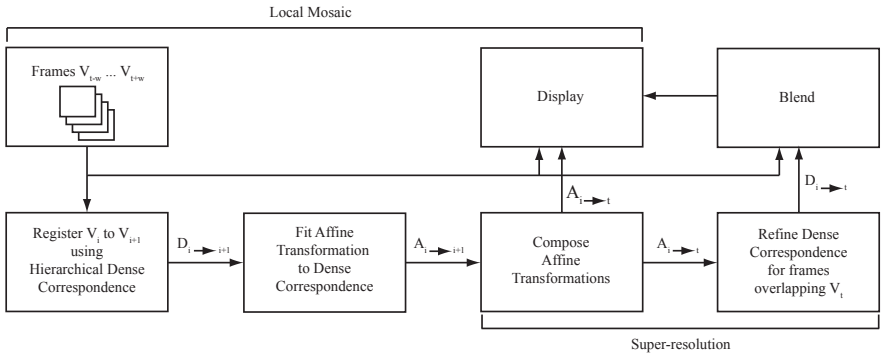

Figure 1. Overview of aerial video enhancement system. To create a temporally local mosaic each frame V_i is mapped to its successor V_{i+1} using HDC to produce $D_{i \mapsto i+1}$. An affine transformation $A_{i \mapsto i+1}$ (Section 5.1.1) is then fit to this and composed with $A_{t \mapsto i}$ to create an affine transformation that (approximately) maps frame V_{i+1} to the anchor frame V_t (Section 5.1.2). Each frame is then displayed using its corresponding affine transformation to create the mosaic. To achieve super-resolution each dense correspondence $D_{i \mapsto t}$ is seeded with $A_{i \mapsto t}$ and then refined with HDC to map each pixel that overlaps V_t. Once registered to sub-pixel accuracy, the source images are blended.

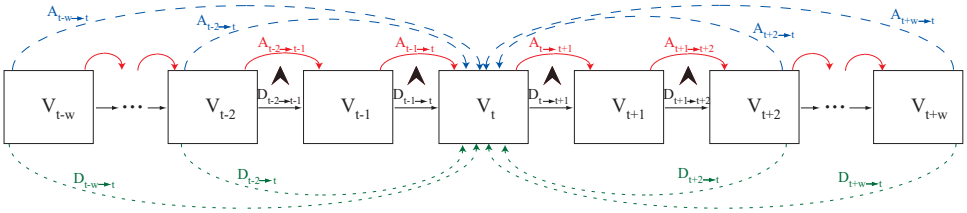

Figure 2. Summary of transformations. A dense correspondence is first calculated between each pair of neighboring frames (solid black arrows). An affine transformation (solid red arrows) is then fit to this and then composed together to create a direct transformation from each frame to the anchor frame (dashed blue arrows). These composed affine transformations are then used to seed the dense correspondence (dashed green arrows) used in super-resolution.

difference from existing algorithms is that the matching and regularization components are decoupled and occur in two distinct passes. The reason this decoupling is important is that it makes the algorithm amenable to GPU implementation where the required calculations are easily distributed to the processing cores to be calculated in parallel, providing a significant acceleration over a CPU-only implementation.

4.1. Hierarchical Algorithm

Using source and target images as input, we perform correspondence on (currently) greyscale imagery and assume that intensity levels across the sensor field have been properly calibrated. The output is a 2D coordinate for every source pixel indicating its forward correspondence into the target domain. In our current formulation, this warping function is presumed to be a bijection (i.e. the mesh of warped samples has no folds or gaps). This is of course not true in general, but we can get quite far under this assumption, especially for aerial imagery.

We begin by constructing a diamond hierarchy for the source and target images (Figure 3). Each successive level is low-pass filtered and contains half as many samples as the previous one, similar to Hwa et al. [14]. Notice that in our application, we use the hierarchy for the purpose of efficient computation and not just for visual display. As compared to a standard image pyramid, the diamond hierarchy improves the prediction quality as we advance from level to level, and reduces potential grid artifacts in the results.

After hierarchy construction, we apply an iterative, two-phase algorithm at successively finer levels of the hierarchy (Figure 4). Starting at the coarsest level, we initialize the transformation from the source to target pixel locations to either an identity transform or to an appropriate affine transform, if one is available. Then we perform some number of iterations, each of which consists of a *matching phase* followed by a *straightening phase* (Figure 5). The resulting transformation for level i of the hierarchy is used as a starting prediction at level $i+1$ using mesh refinement. Around ten iterations are typically required for convergence at the finer levels, and coarser levels can have a hundred or more very inexpensive iterations for robustness.

The matching phase is a gradient descent algorithm. For each pixel of the source image, we compute a local gradient at its current position in the target image. We use this to make a linear prediction of the direction and distance to move the source pixel in the target image to match its intensity. To promote robustness, the step size is clamped to a fraction of a target pixel to mitigate the effect that outliers have during the straightening phase. As the gradient mag-

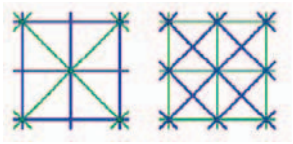

Figure 3. Parent (green) and child (blue) levels of a diamond hierarchy. Left: level i. Right: level $i+1$

```
HIERARCHICALDENSECORRESPOND(V_src, V_dst, A_src↦dst)
{
  for ℓ ← ℓ_coarse to ℓ_fine {
    if (ℓ = ℓ_coarse) {
      if (A_src↦dst is non null) D_ℓ ← sample(ℓ, A_src↦dst)
      else D_ℓ ← sample(ℓ, I)    // I = identity mapping
    }else D_ℓ ← split(D_ℓ−1)    // refine next coarser mapping

    for j ← 1 to j_max(ℓ) {

      // perform matching motion per pixel
      Move each source pixel's destination position in D_ℓ
      towards closest matching isoline in destination image
      V_ℓ,dst. Clamp motion to a fraction of a pixel width, and
      avoid motion for destination positions with small gradi-
      ents or missing/bad pixels.
      // straighten warp mesh
      Perform least-squares affine fit A_p for the 5 × 5 neigh-
      borhood centered at each source pixel p in D_ℓ. For each
      source pixel p′ store the average of the affine positions
      computed from the 5 × 5 neighborhood of local affine fits
      A_p centered around p′.
    }
  }
  return D_ℓ
}
```

Figure 4. Hierarchical Dense Correspondence.

nitude becomes small, the gradient direction becomes more noise than signal, so we disqualify such pixels, as well as those that go outside the bounds of the target image, from motion during the matching phase.

Whereas each source pixel moves independently in the matching phase, the straightening phase uses information about the current locations of source pixel neighborhoods to locally regularize the warp. For each source pixel, a local neighborhood of locations is used to compute an affine transformation from the source to target image (we find that a 5×5 neighborhood is sufficiently large). The new target location of each source pixel then becomes a weighted average of its target locations as predicted by all the local affine transformations to which it contributes.

4.2. GPU Implementation

The architecture of the commodity Graphics Processing Unit (GPU) is ideally suited to accelerate the dense correspondence algorithm. Modern GPUs provide hundreds of gigaflops of processing power, driven by a large number of floating point processing cores (e.g. 128 cores on the NVIDIA 8800 GTX). Furthermore, GPUs are designed to

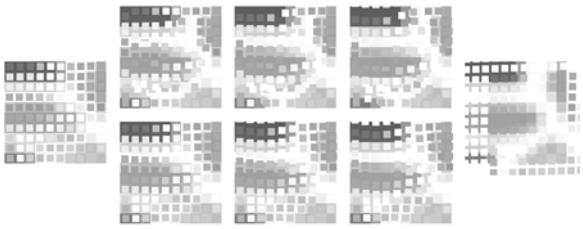

Figure 5. HDC stages. Left: source image (smaller squares) superimposed on destination image. Center: two phases of the HDC algorithm. The top row demonstrates how the pixels move independently during the matching phase, while the bottom row shows the pixel locations after the straightening phase. Right: after several iterations of alternating the matching and straightening phases, the images are aligned. The resulting locations are then propagated down to the next finer level, and the process repeats.

provide highly optimized memory access for programs employing 2D local access patterns. The correspondence algorithm is not significantly hindered by the limitations of the standard GPU programming model, such as reduced efficiency for incoherently branching code and lack of support for scattered write operations or inter-thread communication (though this is actually improving on the latest GPUs).

Both the initial hierarchy construction and the iterative correspondence algorithms map well to the fragment processing units on the GPU. Our implementation was written using CUDA and targeted for an NVIDIA Geforce 8800 GTX GPU. To compute the dense correspondence the source and destination images are first copied into GPU memory and a diamond hierarchy is created for each image using the grayscale intensity values. We also construct a diamond hierarchy of gradient values for the target image. Each hierarchy construction requires one "rendering pass" (computational kernel execution) for each hierarchy level.

The iterative algorithm is broken into passes for each phase of an iteration. The matching phase, requiring a single pass, maps a fragment thread to each source pixel. A thread reads a source pixel intensity and the current corresponding location in the target image to obtain the magnitude of the intensity difference. This is combined with the gradient of the target image at the current target position of the source pixel to compute the new $2D$ target location, which is output to a $2D$ texture via a frame buffer object.

The straightening phase requires two passes. The first maps a fragment thread to each source pixel. For neighboring pixels (using a 5×5 neighborhood), the current warp locations are read and contribute to a 6-parameter affine transformation, which is output from the first pass. The second pass also maps a fragment thread to each source pixel. The affine transformations computed in the first pass are read for each member of the source pixel's 5×5 neighborhood. Each of these affine transformations is applied to the thread's source pixel, and a weighted combination of the

results is used to compute the pixel's location after straightening, which is output from the fragment program.

We have benchmarked the two algorithm phases on an NVIDIA Geforce 8800 GTX GPU. The matching phase and straightening phases achieve rates of 145 million and 68 million pixel-iterations per second, respectively. This performance is over 230 times that of our unoptimized CPU implementation running on a single processor of an Intel dual-Xeon 3.0 GHz.

5. Application and Examples

We now demonstrate the HDC algorithm and accompanying framework (Figure 2) for implementing temporally local mosaics [23] super-resolution from multiple video frames, and detection of intrinsic scene motion.

5.1. Local Mosaics

Creating mosaicked sequences requires concatenating chains of correspondences that relate each frame to its neighbors. Although it is possible to compose the HDC mappings directly, this is computationally expensive and does not extrapolate well. To overcome these limitations—and for compositing long sequences only—we reduce dense correspondence from the HDC to a global transformation through least-squares fitting.

5.1.1 Affine Transform Estimation

The first step in creating a local mosaic is to create a transformation between each pair of neighboring frames in V. Because the current frame-to-frame transformation is typically similar to the previous one, we initialize the dense correspondence $D_{i\mapsto i+1}$ with the previous affine transformation $A_{i-1\mapsto i}$. (If $A_{i-1\mapsto i}$ has not yet been calculated, we initialize $D_{i\mapsto i+1}$ with the identity mapping.) We then run the HDC algorithm to get the dense correspondence $D_{i\mapsto i+1}$ between V_i and V_{i+1}.

We use a pseudoinverse technique to fit an affine transformation $A_{i\mapsto i+1}$ to $D_{i\mapsto i+1}$. By thresholding the gradient and correspondence error in the HDC phase, we fit the global transformation using only high-quality correspondences. The x and y components, A^x and A^y of the affine transformation $A_{i\mapsto i+1}$ are calculated independently, as

$$\begin{aligned} A^x_{i\mapsto i+1} &= D^x_{i\mapsto i+1} M(M^T M)^{-1} \\ A^y_{i\mapsto i+1} &= D^y_{i\mapsto i+1} M(M^T M)^{-1} \end{aligned} \tag{1}$$

where $D^x_{i\mapsto i+1}$ and $D^y_{i\mapsto i+1}$ are the x and y components of $D_{i\mapsto i+1}$, respectively, and M is a $N\times 3$ matrix of homogeneous pixel locations $(x_i, y_i, 1)$ giving us an affine transformation $A_{i\mapsto i+1}$ that maps each frame to its successor.

Even in situations when dense correspondence is not ultimately needed, first finding dense correspondence and then fitting global transformations to them can have advantages over directly computing global transformations alone. When there is significant parallax or uncorrected lens distortion in the sequence, a single global transformation alone cannot correctly match all of the points. This causes sparse feature-based approaches to be sensitive to the location of selected feature points, and dense area-based approaches to be driven by the mismatched areas.

5.1.2 Composing Affine Transformations

To correctly display all frames of the local mosaic, we compose the frame-to-frame transformations $A_{i\mapsto i+1}$ to create a transformation $A_{i\mapsto t}$ that maps each frame V_i directly to the anchor frame V_t.

Our goal in creating this temporally local mosaic is to provide additional context for the currently viewed frame while allowing features in the video to persist longer on the screen. When transformations are composed without doing a bundle adjustment, small errors in the transformations will accumulate and distort frames that are temporally distant from the anchor frame. While small distortions are not necessarily inconsistent with our goals, we would like to minimize distortions as much as possible, while maintaining the real-time display of the mosaic. Thus, we select the middle frame of the video sequence V to be the anchor frame, allowing both V_{t-W} and V_{t+W} to be as close to the anchor frame as possible. Since the affine transformations calculated above are only calculated in one direction, we need to invert the affine transformations $A_{i\mapsto t}$ where $i > t$ before doing the composition. We calculate $A_{i\mapsto t}$ from the previous composition, $A_{i+1\mapsto t}$, as

$$A_{i\mapsto t} = \begin{cases} A_{i+1\mapsto t} A_{i\mapsto i+1} & \text{if } i < t \\ A_{i-1\mapsto t} \hat{A}_{i-1\mapsto i} & \text{if } i > t \\ I & \text{otherwise} \end{cases} \tag{2}$$

where $\hat{A}_{i-1\mapsto i}$ is the inverse of $A_{i-1\mapsto i}$ and I is the identity transformation.

Each time t advances, we only need to compute one new frame-to-frame affine transformation in order to display the complete mosaic M_t. The frame-to-frame affine transformations are composed according to Equation 2 then displayed as described in the next section.

5.1.3 Mosaic Display

While mosaics can aid in building of a mental model of the scene, artifacts may appear during the mosaic process, especially in frames far from the anchor frame. In order to ensure that each frame appears undistorted at some point during the playback of the mosaic we keep the anchor frame undistorted at all times and display the composited mosaic from frames $\{V_{t-W}, \ldots, V_t, \ldots, V_{t+W}\}$ relative to it (Figure 6). For live video, of course, this requires buffering the

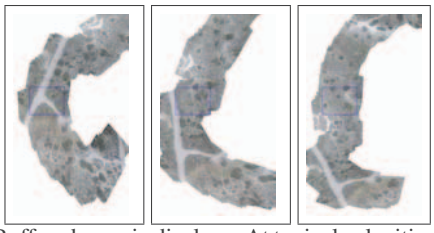

Figure 6. Buffered mosaic displays. At typical velocities, the scene visible in a single displayed frame passes by in approximately one second. Mosaicing dramatically extends the visible lifetime of the images. By keeping the anchor frame fixed and warping the rest of the local mosaic, rather than warping each new frame, the anchor frame remains undistorted. By buffering the video stream, the mosaic can show not only what has already passed through as the undistorted anchor frame but what is approaching.

display by $2W+1$ frames between the entering new frame, the undistorted anchor frame, and the exiting frame.

We can filter out unwanted motions by allowing the anchor frame to translate. Experience has shown that it can also be helpful to unwind the rotations caused by the turning of the plane by allowing the anchor frame (and hence the mosaic) to counter-rotate. Thus, we can maintain a consistent orientation.

Users are also allowed to zoom and pan the mosaic as desired. As the user zooms in or out, we adjust the frame window size W accordingly. When the user zooms in far enough so that the anchor frame is occupying more screen real estate than its original dimensions, we then transition to displaying the mosaic in super-resolution.

5.2. Super-Resolution

Many applications require covering a target area as quickly and as thoroughly as possible. Increasing the area that the mini-UAV covers on each pass while still maintaining the resolution needed to identify ground features can shorten the time to search a given area. Super-resolution techniques can enable the mini-UAV to fly at higher altitudes (or use wider-angle lenses) while providing the same resolution as if it was flying at a lower altitude.

As with other multi-image super-resolution methods, we combine information from overlapping video frames. The movement of the mini-UAV causes each frame to be captured from a slightly different viewpoint. As a result, pixels in each source image are integrating over a finite area of the scene that is slightly offset from the other images.

Super-resolution requires calculating very accurate alignment for each of the source images that overlap the anchor frame, allowing us to take advantage of additional information gained from repeatedly sampling the scene. Significant parallax due to the movement of the mini-UAV and depth variation in the scene can cause alignment problems for global registration techniques. The local warps that the HDC approach allows, however, can accommodate parallax

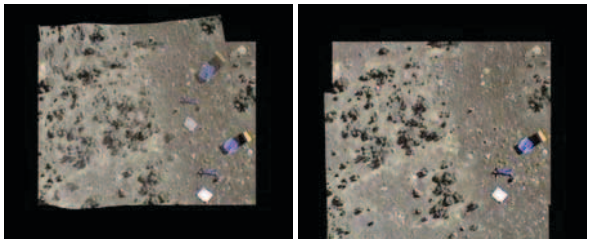

Figure 7. Dense correspondences computed directly between distant frames with limited overlap (left) and seeded with composed affine approximations (right).

Figure 8. 61-frame super-resolution example. Left: one image. Right: 61 images warped and super-resolved.

motion and give us the needed accuracy in alignment.

To generate a resolution-enhanced anchor frame V_t, we collect surrounding frames and warp these into the anchor frame's coordinate system using dense correspondence. The affine transformations that we calculated during the mosaic construction ($A_{i \mapsto t}$) are used to initialize the HDC warps ($D_{i \mapsto t}$), allowing us to robustly warp frames that only slightly overlap the anchor frame. We then refine the registration using the HDC algorithm to align the images to sub-pixel accuracy ($D_{i \mapsto t}$). Figure 7 shows an example of dense correspondence between distant frames seeded using composited affine transformations compared to dense correspondence calculated directly.

Once all of the correspondences are calculated we combine the source images to create a super-resolution image. Each pixel of each source image is displayed on screen at their floating-point coordinates.Each screen pixel then selects for its color the source pixel it is closest to (nearest neighbor) or a blending of the source pixels that are close by (weighted average). This blending process is accelerated using the GPU to keep the display of the super-resolution image at interactive speeds.

Many methods for super-resolution frame it instead as an inverse problem [4, 17, 25]. These approaches also require accurate pixel alignment and would also benefit from HDC's provision of dense corresponded. We use a forward- rather than backward-projection approach simply for efficiency and to further make use of the GPU.

One indicator of the accuracy of the dense correspondence computation is the ability to create super-resolution images by warping multiple images to a single frame of reference and compositing, as shown in Figures 8 and 9. Figure 10 compares the results of super-resolution for both global and dense correspondence when parallax occurs.

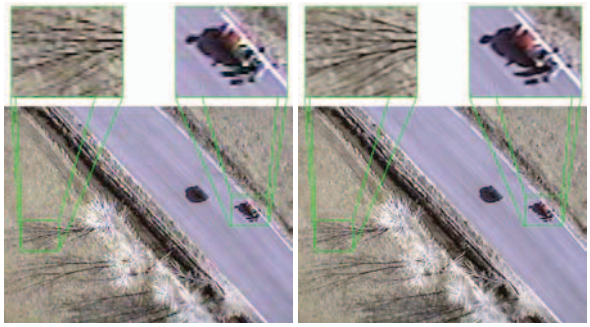

Figure 9. 15-frame super-resolution example. Left: a single input frame. Right: super-resolution output. Notice the additional detail in the shadows of the trees and around the red car, as shown in the magnified insets. Also notice how color artifacts caused by Bayer sampling in the camera sensor are removed in the super-resolution version, especially around areas that transition from light to dark.

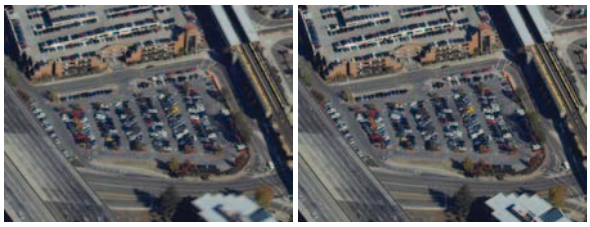

Figure 10. Affine warp (left) versus dense correspondence (right) for super-resolution. Notice how parallax causes misalignment (blurring) of the taller, nearer building when using global correspondence, but the corresponding result for dense correspondence remains sharp.

5.3. Mover Detection

Segmentation of moving objects is performed in two phases: detection and completion. In the detection phase, the goal is to have at least a single pixel of positive detection per moving object, not including parallax motion of buildings, trees, or other tall structures. In the completion phase, pixels for the entire object are determined. Extra "guard pixels" surrounding the object are also labeled as part of the mover to obtain a conservative segmentation. Both phases could be accomplished with well-known but expensive search and correlation strategies [8, 21], but for real-time processing on small sensor platforms, fast, local computations are desired.

The detection phase uses five consecutive frames $\{V_{t-2} \ldots V_{t+2}\}$ and their dense correspondences to the anchor frame $\{D_{t-2 \mapsto t} \ldots D_{t+2 \mapsto t}\}$. The median value is obtained for each pixel over these frames. The detection is then the difference between the current frame's value and the least different value from a small neighborhood of the next frame. A small amount of "soft erosion" is performed on the results. This was found to be very reliable at detecting some pixels per mover and was not as sensitive to noise or to building edge parallax as single-frame differencing.

After detecting a set of pixels from each mover, a conservative region of pixels is determined around each complete mover. Using the results of the five-frame detection, these are good starting points to seed a search for all mover pixels. We start with the core pixel of each connected component, defined as the last pixel to be deleted for the component by repeated erosion operations that are restricted to not break connected components. From this starting point, we flood fill to all pixels that are near pixels with frame-to-frame differences with magnitudes above a specified threshold.

Figure 11. Affine warp versus dense correspondence. Top: source and target images. Middle: affine warp with difference image. Bottom: dense correspondence warp with difference image (significant differences are movers).

Figure 11 shows the differences in mover detection accuracy using a dense correspondence registration versus an affine warp. The dense correspondence matches the background scene much better and eliminates the vast majority of false alarms caused by parallax motion when using a global transformation alone.

6. Conclusion

In this paper we have presented a novel method hierarchical dense correspondence algorithm designed for imple-

mentation on GPUs. We have also presented a framework for using these dense correspondences for such aerial video processing applications as local mosaic creation, super-resolution, and mover detection. This approach, based on a combination of hierarchical dense correspondences and affine transformations fit to those correspondences, performs robustly even on mini-UAV video with no extrasensory information and in jittery flight conditions. The same framework permits not only basic mosaics but fast computation of super-resolution imagery on the GPU, as well as mover detection. The use of dense correspondences means the method is robust even when there is parallax from 3D structure in the scene or when using uncorrected video with non-linear lens distortions.

7. Acknowledgments

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. It was partially supported by the National Science Foundation under grant number 0534736.

References

[1] S. Baker and T. Kanade. Super resolution optical flow. Technical Report CMU-RI-TR-99-36, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, October 1999.

[2] S. Baker and T. Kanade. Limits on super-resolution and how to break them. In *IEEE CVPR*, volume 2, page 2372, 2000.

[3] K. S. Bhat, M. Saptharishi, and P. K. Khosla. Motion detection and segmentation using image mosaics. In *IEEE International Conf. on Multimedia and Expo*, volume 3, pages 1577–1580, 2000.

[4] T. E. Boult and G. Wolberg. Local image reconstruction and sub-pixel restoration algorithms. *CVGIP: Graphical Models and Image Processing*, 55(1):63–77, 1993.

[5] J. Casper and R. R. Murphy. Human-robot interactions during the robot-assisted urban search and rescue response at the world trade center. *IEEE Trans. Systems, Man, and Cybernetics, Part B*, 33:367–385, 2003.

[6] M. Chen, R. Botchen, , R. Hashim, D. Weiskopf, T. Ertl, and I. Thornton. Visual signatures in video visualization. *IEEE Trans. Visualization and Computer Graphics*, 12(5):1093–1100, 2006.

[7] M.-C. Chiang and T. E. Boult. Efficient image warping and super-resolution. In *IEEE WACV*, pages 56–61, December 1996.

[8] R. Collins, A. Lipton, and T. Kanade. A system for video surveillance and monitoring. In *American Nuclear Society 8th Internal Topical Meeting on Robotics and Remote Systems*, 1999.

[9] R. T. Collins, Y.-Q. Cheng, C. O. Jaynes, F. Stolle, X. Wang, A. R. Hanson, and E. M. Riseman. Site model acquisition and extension from aerial images. In *IEEE ICCV*, pages 888–893, 1995.

[10] N. Damera-Venkata and N. L. Chang. Realizing super-resolution with superimposed projection. In *IEEE CVPR*, pages 1–8, 2007.

[11] G. Daniel and M. Chen. Video visualization. In *VIS '03: Proceedings of the 14th IEEE Visualization*, pages 409–416, Washington, DC, USA, 2003. IEEE Computer Society.

[12] R. Fransens, C. Strecha, and L. V. Gool. Optical flow based super-resolution: A probabilistic approach. *Computer Vision and Image Understanding*, 106(1):106–115, 2007.

[13] T. Huang and R. Tsai. Multi-frame image restoration and registration. In *Advances in Computer Vision and Image Processing*, volume 1, pages 317–339. 1984.

[14] L. M. Hwa, M. A. Duchaineau, and K. I. Joy. Real-time optimal adaptation for planetary geometry and texture: 4-8 tile hierarchies. *IEEE Trans. Vis. and Comp. Graphics*, 11(4):355–368, 2005.

[15] M. Irani, P. Anandan, and S. Hsu. Mosaic based representations of video sequences and their applications. In *IEEE ICCV*, pages 605–611, 1995.

[16] M. Irani and S. Peleg. Improving resolution by image registration. *Graphical Models and Image Processing*, 53:231–239, 1991.

[17] D. Keren, S. Peleg, and R. Brada. Image sequence enhancement using sub-pixel displacements. In *IEEE CVPR*, pages 742–746, June 1988.

[18] R. Kumar, H. Sawhney, S. Samarasekera, S. Hsu, H. Tao, Y. Guo, K. Hanna, A. Pose, R. Wildes, D. Hirvonen, M. Hansen, and P. Burt. Aerial video surveillance and exploitation. *Proceedings of the IEEE: Special Issue on Third Generation Surveillance Systems*, 89(10):1518–1539, October 2001.

[19] R. Kumar, H. S. Sawhney, J. C. Asmuth, A. Pope, and S. Hsu. Registration of video to geo-referenced imagery. *IEEE CVPR*, pages 54–62, August 1998.

[20] Y. Lin and G. Medioni. Map-enhanced UAV image sequence registration and synchronization of multiple image sequences. In *IEEE CVPR*, pages 1–7, 2007.

[21] A. J. Lipton, H. Fujiyoshi, and R. S. Patil. Moving target classification and tracking from real-time video. In *IEEE WACV*, pages 8–14, 1998.

[22] S. Mann and R. W. Picard. Virtual bellows: constructing high quality stills from video. In *IEEE ICIP*, volume 1, pages 363–367, 1994.

[23] B. Morse, D. Gerhardt, C. Engh, M. A. Goodrich, N. Rasmussen, D. Thornton, and D. Eggett. Application and evaluation of spatiotemporal enhancement of live aerial video using temporally local mosaics. In *IEEE CVPR*, June 2008.

[24] S. Peleg and J. Herman. Panoramic mosaics by manifold projection. In *IEEE CVPR*, pages 338–343, 1997.

[25] S. Peleg, D. Keren, and L. Schweitzer. Improving image resolution using subpixel motion. *Pattern Rec. Letters*, 5(3):223–226, 1987.

[26] D. Steedly, C. Pal, and R. Szeliski. Efficiently registering video into panoramic mosaics. In *IEEE ICCV*, pages 1300–1307, 2005.

[27] R. Szeliski. Video mosaics for virtual environments. *IEEE Computer Graphics and Applications*, 16(2):22–30, March 1996.

[28] R. Szeliski. Image alignment and stitching: A tutorial. *Foundations and Trends in Computer Graphics and Vision*, 2(1):1–109, 2006.

[29] R. Szeliski and H.-Y. Shum. Creating full view panoramic image mosaics and texture-mapped models. In *ACM SIGGRAPH*, pages 251–258, August 1997.

[30] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment—a modern synthesis. In B. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice*, volume 1883 of *Lecture Notes in Computer Science*, pages 298–372. Springer-Verlag, 2000.

[31] R. P. Wildes, D. J. Hirvonen, S. C. Hsu, R. Kumar, W. B. Lehman, B. Matei, and W. Y. Zhao. Video georegistration: algorithm and quantitative evaluation. In *IEEE ICCV*, pages 343–350, 2001.

[32] Q. Yu and G. Medioni. A gpu-based implementation of motion detection from a moving platform. In *IEEE Workshop on Computer Vision on GPU*, pages 1–6, June 2008.