Toward Fast Computation of Dense Image Correspondence on the GPU

Mark Duchaineau Jonathan Cohen Sheila Vaidya
Lawrence Livermore National Laboratory
{duchaineau1, jcohen, vaidya1}@llnl.gov

Introduction

Large-scale video processing systems are needed to support human analysis of massive collections of image streams. Video, from both current small-format and future large-format camera systems, constitutes the single largest data source of the near future, dwarfing the output of all other data sources combined.

A critical component to further advances in the processing and analysis of such video streams is the ability to register successive video frames into a common coordinate system at the pixel level. This capability enables further downstream processing, such as background/mover segmentation, 3D model extraction, and compression.

We present here our recent work on computing these correspondences. We employ coarse-to-fine hierarchical approach, matching pixels from the domain of a source image to the domain of a target image at successively higher resolutions. Our diamond-style image hierarchy, with total pixel counts increasing by only a factor of two at each level, improves the prediction quality as we advance from level to level, and reduces potential grid artifacts in the results.

We demonstrate the quality our approach on real aerial city imagery. We find that registration accuracy is generally on the order of one quarter of a pixel. We also benchmark the fundamental processing kernels on the GPU to show the promise of the approach for real-time video processing applications.

Dense Correspondence Algorithm

The input to the correspondence algorithm is a source and target image. We currently perform correspondence on greyscale imagery and assume that intensity levels across the sensor field have had proper calibration applied. The output of the algorithm 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 (see Figure 1). Each successive level is low-pass filtered and contains half as many samples as the previous one, as described in [1].

We next apply an iterative, two-phase algorithm at successively finer levels of the hierarchy. Starting at the coarsest level, we initialize to an identity transformation from the source to target pixel locations. Then we perform some number of iterations, each of which consists of a *matching phase* followed by a *straightening phase*. The resulting transformation for level *i* of the hierarchy is used as a starting prediction at level *i*+1 using mesh refinement.

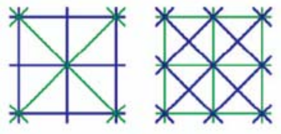

Figure 1: Parent (green) and child (blue) levels of a diamond hierarchy. **Left:** level *i* **Right:** level *i*+1

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 the gradient 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 taken is clamped to a fraction of a target pixel size, and this is further modified according to the magnitude of the gradient. As the gradient magnitude becomes small, the gradient direction becomes more noise than signal, and we ultimately disqualify such pixels from match-phase motion.

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 (e.g., 5x5 block) of locations is used to compute an affine transformation from the source to target image. The updated location of each source pixel is a weighted average of its target locations as predicted by all the local affine transformations to which it contributes.

Mapping to GPU

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 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. The hierarchy construction requires one "rendering pass" (computational kernel execution) for each level of the hierarchy.

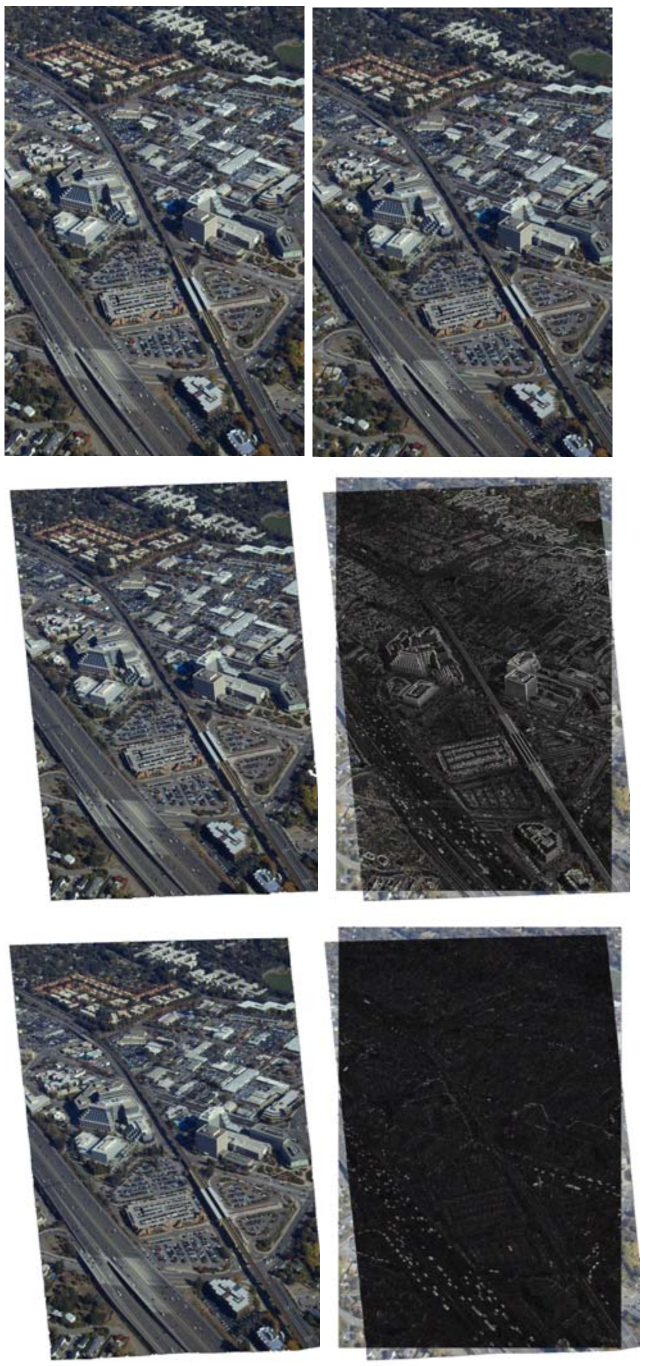

Figure 2: Affine warp versus dense correspondence. **Top:** source and target images. **Middle:** affine warp with difference image. **Bottom:** dense correspondence warp with difference image (significant differences are movers).

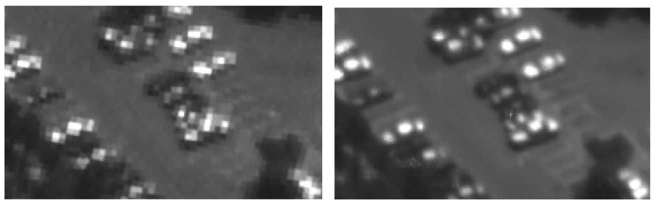

Figure 3: Super-resolution. **Left:** one image. **Right:** 61 images warped and composited into a high-resolution image.

The iterative algorithm requires one pass for each phase of an iteration. The matching phase reads and interpolates a gradient from the target data and outputs a 2D coordinate for each pixel. The straightening phase reads the 2D coordinates from each source pixel and outputs a straightened 2D coordinate. Breaking the phases into rendering passes avoids the need for inter-thread communication to compute the straightened coordinates.

We have benchmarked the computational kernels on the registration of two 1024x1024-resolution images. The straightening phase has been simplified somewhat, and just performs a 3x3 averaging of source pixel locations, repeated three times. On NVIDIA Quadro 5500, a previous-generation (G70-class) GPU, we can perform 74 iterations/second. On a current-generation (G80-class) GPU, the NVIDIA Geforce 8800 GTX, we achieve 219 iterations/second. Typically 15 iterations are needed on each level of the hierarchy. With performance proportional to the number of pixels on each level, we can expect over 7 fully corresponded frames per second on images of this resolution. This is about 3000 times faster than our completely unoptimized CPU implementation running on a single core.

Analysis and Conclusions

In Figure 2, we explore the benefit of our dense correspondence registration versus a simpler, affine warp. The dense correspondence matches the background scene much better, which eliminates the vast majority of false alarms in mover detection.

Another 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. Visual perturbation analysis indicates that registration errors are less than 0.25 pixel widths.

The dense correspondence algorithm presented provides high-quality image registration at multiple frames per second on a GPU. As ongoing work, we are investigating identification of and support for discontinuous warps as well as applications such as 3D model extraction.

Acknowledgements

We would like to thank the Sonoma Project and Brett Wayne for the video imagery used in our experimentation. This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

References

[1] Lok M. Hwa, Mark A. Duchaineau and Kenneth I. Joy. “Real-time Optimal Adaptation for Planetary Geometry and Texture: 4-8 Tile Hierarchies.” *IEEE Transactions on Visualization and Computer Graphics*. 11 (4). pp. 355-368. 2005.