Optimization of the Earth-Mover's Distance algorithm for a GPU-based real-time multispectral computer vision system

Andrew Shaffer
aps148@arl.psu.edu
Applied Research Laboratory, Penn State University, State College, PA 16801

Introduction

Color-based target detection and tracking systems operate by monitoring a scene and finding regions of interest with color distributions that closely match the color distribution of a target that is to be tracked. When the level of color resolution provided by a standard tri-chromatic camera is insufficient to differentiate a target from its background, such as when the target is deliberately camouflaged, multispectral cameras can increase the number of color channels that are sampled to provide a more precise description of the colors in the scene and restore the distinction between a target and its background. Once precise color signatures for a scene are available, success in locating and tracking a target becomes primarily a question of the tracker algorithm's ability to accurately determine the similarity between the color signatures in the scene and the color signature of the target.

The Earth-Mover's Distance (EMD) [1] provides a highly robust measure for determining the similarity between two multispectral color signatures. It supports color distance computation by finding the minimum cost to move all of the flow through a flow network in which the sources are defined by the bin weights of one histogram, the sinks are defined by the bin weights of the other histogram, and every source is connected to every sink by an edge with a cost corresponding to the distance between the two histogram bins being connected. Since the distance between any two histogram bins is specified individually by the programmer, the EMD provides great flexibility in defining image similarity metrics. However, because the solution of a network flow problem typically requires the computation of a linear programming problem, the use of EMD is computationally expensive. EMD's high computational cost has precluded it from being widely adopted in previous real-time computer vision systems.

In this paper, we present an approach to overcome the high computational cost of the EMD algorithm by using a GPU, and we demonstrate a GPU-based system for solving large numbers of EMD computations in parallel at very high speed. Our system is designed to support real-time multispectral object detection and tracking in the context of a fixed-position video camera monitoring a scene, and it provides support for performing an EMD computation on every pixel of each frame of a video stream as the frames arrive in sequence. Using simulated multispectral input data with up to 11 color channels and a single NVIDIA C2050 GPU, we demonstrate that our system is capable of processing high resolution multispectral video feeds at frame rates sufficient to support many real-world computer vision applications.

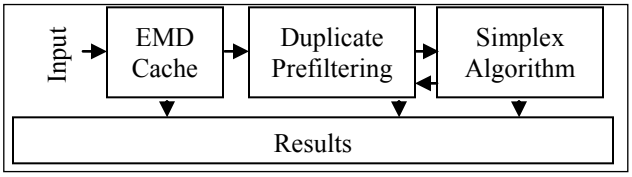

Figure 1: System Design

System Design

Our system achieves high performance by processing each video frame using a three step process:

1. Look up cached results for EMD problems that were solved in previous frames. A scene monitored by a fixed-location video camera does not typically change dramatically between frames, so many of the pixel colors in any given frame will have already been processed in a preceding frame. The C2050 GPU's large on-card memory provides an ideal location in which to cache previously computed EMD results.

2. Using atomic instructions to support a shared global hash table, identify groups of duplicate EMD problems within the current frame's set of unsolved EMD problems. Objects in a scene often correspond to multiple pixels that share the same color histogram. Since each color histogram corresponds to a unique EMD result, it is only necessary to compute the EMD between each color histogram and the target color histogram once – the result from the single EMD computation can be assigned to all of the pixels that share the same color histogram.

3. Find the EMD for any remaining color histograms in parallel using an optimized implementation of the two-phase Simplex algorithm presented in [2].

Simplex Algorithm Optimizations

As noted in [1], the EMD problem is an instance of the transportation problem, for which it is known that the entries of the Simplex tableaux will only ever assume the values -1, 0, or 1 [3]. This property allows us to replace most of the arithmetic operations needed for pivoting the Simplex tableaux with logical operations, and also to store most of the tableaux contents in a densely packed bitmap format where each thread of a GPU warp can store and update the values for up to 32 columns of a row of the Simplex tableaux using only two unsigned integer variables. For EMD problems that compare the histogram values for 11 or fewer color channels, which is typical for modern multispectral cameras, the use of this packed bitmap storage format makes it possible for each 32-thread warp to store and process 2 or more EMD problems at the same time with all of the tableaux data stored directly in the GPU register file. Furthermore, because each problem

is solved entirely within a single warp, we are able to avoid all overhead due to inter-warp synchronization.

Additionally, we are able to simplify and improve the performance of our algorithm by bypassing the entire first phase of the two-phase Simplex algorithm. This is accomplished by modifying the flow network for our EMD problem to contain an auxiliary sink that can be reached from every source but which costs more to access than any other path in the standard EMD flow network, and then initially assigning all of the flow from every source to this auxiliary sink in the initial tableaux setup. Since this initial setup assigns all of the flow from every source in the flow network to a sink, the criterion for ending phase 1 pivoting is already met and our algorithm can begin phase 2 pivoting immediately. We rely on the fact that the Simplex algorithm minimizes the cost of moving the flow through the network to ensure that the algorithm will reassign all of the flow that is initially assigned to the expensive auxiliary sink and that it will obtain the optimal solution. The elimination of phase 1 pivoting simplifies our implementation, reduces GPU register pressure, and typically results in a significant decrease in the number of pivot operations needed to find the optimal solution.

Finally, our implementation is designed to maximize the effective utilization of the GPU by overlapping memory transfers with computation whenever possible, by splitting the cache lookup and duplicate problem detection operations into a separate kernel from the EMD solver kernel to increase GPU occupancy, and by implementing several problem-size dependent optimizations to replace control loops and streamline shared variable reductions for small problem sizes that support these optimizations.

Methodology

Due to the high cost of obtaining a multispectral video camera and the difficulty of obtaining existing multispectral video data, we simulate multispectral input data by computing intensity distribution histograms for each pixel of a region at the upper left corner of Matlab's grayscale image 'concordorthophoto.png' using an 11x11 pixel window to compute the intensity distribution histograms. Tests were run using a 1280x720 pixel region and a 1920x1080 pixel region. Each bin of the intensity distribution histogram for each pixel corresponds to one color channel of the simulated multispectral video input for that pixel. Similarly, a random multispectral target color histogram is computed by generating an intensity distribution histogram over the entirety of Matlab's 'pout.tif' image. For each number of color channels considered, the EMD between the color histograms for every pixel in the input image and the color histogram of the target is then computed twice. First, an initial call to our EMD solver is made with the EMD cache empty to determine the typical performance that could be achieved in a video stream in which every pixel is constantly changing. Second, after the initial call is completed and its results have been stored in the EMD cache, the same set of EMD problems is solved a second time to approximate the typical best case performance that might be achieved if none of the pixels in the input video stream changed at all between two frames. We expect that the actual real-world performance of our system will fall somewhere between these two values, with the exact performance varying according to the amount of the scene that changes in each frame. Our timing results for both tests include the time necessary to transfer data from main memory to the GPU and back, but exclude the time needed for GPU setup and cleanup, since data transfer must be performed for each frame but setup and cleanup only need to be performed once at startup and shutdown.

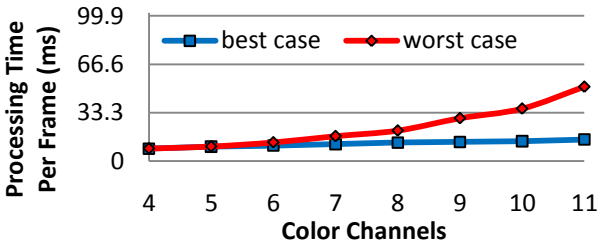

Figure 2: Processing Time vs. Color Channels (1280x720)

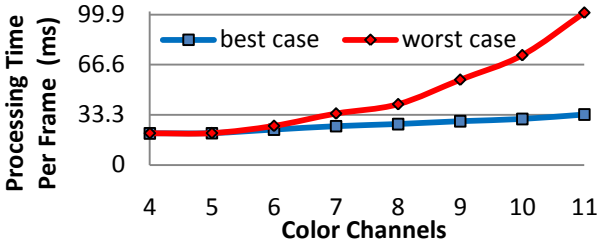

Figure 3: Processing Time vs. Color Channels (1920x1080)

Results

Our simulation results demonstrate that our EMD solver system is capable of processing high resolution multispectral video feeds with up to 11 color channels at frame rates between 9.9-117.1 FPS, depending on the number of color channels used and the variability of the scene being observed. These frame rates are sufficient for many current computer vision applications, and could be improved upon by adding logic to the tracker algorithm to limit the regions of the video frames on which EMD computations must be performed.

Conclusion

The excellent performance of the GPU in solving low-dimensional EMD problems indicates the feasibility of using the GPU to perform large-scale EMD processing in real-time environments. Since prior CPU-based systems have not generally been able to use EMD as a distance metric due to its high computational cost, our system adds this versatile algorithm to the system developer's software development toolkit.

References

[1] Y. Rubner, C. Tomasi, and L. Guibas. "The Earth Mover's Distance as a Metric for Image Retrieval," *International Journal of Computer Vision,* Vol. 40, No. 2, Nov. 2000.

[2] S. Warner and S. Costenoble, *Finite Mathematics and Applied Calculus,* Fourth Edition, Ch. 4, Brooks/Cole, 2007.

[3] H. Nagler, "On a Property of the Simpex Tableaux in the Transportation Problem," *Numerische Mathematik,* Vol. 4, No. 1, Dec. 1962.