A 203 FPS VLSI ARCHITECTURE OF IMPROVED DENSE TRAJECTORIES FOR REAL-TIME HUMAN ACTION RECOGNITION

Zhi-Yi Lin[⋆†] *Jia-Lin Chen*[⋆] *Liang-Gee Chen*[†]

[⋆†] zylin@video.ee.ntu.edu.tw [⋆]jocelyn@video.ee.ntu.edu.tw [†] lgchen@ntu.edu.tw
DSP/IC Design Lab, Graduate Institute of Electronics Engineering
National Taiwan University

ABSTRACT

This paper introduces architecture with high throughput, low on-chip memory, and efficient data access for Improved Dense Trajectories (iDT) as video representations for real-time action recognition. The iDT feature can capture long-term motion cues better than any existing deep feature, which makes it crucial in state-of-the-art action recognition systems. There are three major features in our architecture design, including a low bandwidth frame-wise feature extraction, low on-chip memory architecture for point tracking, and two-stage trajectory pruning architecture for low bandwidth. Using TSMC 40nm technology, our chip area is 3.1 mm^2, and the size of on-chip memory is 40.8 kB. The chip can support videos in resolution of 320×240 with a throughput of 203 fps under 215 MHz, which is a 81.2 times speedup compared with CPU. Under the same operating frequency, it can also provide feature extraction for six windows of size 320×240 in higher resolution videos with a throughput of 34 fps.

Index Terms— real-time action recognition, Improved Dense Trajectories, ASIC, chip design, feature extraction architecture

1. INTRODUCTION

Action recognition receives a lot of attentions from researchers in computer vision field because of its wide spectrum of applications. It is important to response in real time for applications related to interaction or security issues, such as surveillance system, robots, and self-driving cars. However, few methods can really achieve real-time performance with CPUs, let alone action prediction, which is the ultimate goal of action recognition to predict actions that are about to happen like humans do. Therefore, hardware acceleration is a must in real applications of action recognition.

There are plenty of features proposed to represent actions in video sequences. Trajectory-based features [1, 2, 3, 4] are the trend of hand-crafted features since they can capture information in longer time segments, which is proved to be crucial for action recognition. The Improved Dense Trajectories (iDT) [2], belonging to trajectory-based features, is the most efficient action features used in state-of-the-art action recognition systems [3, 4, 5, 6, 7, 8, 9, 10, 11].

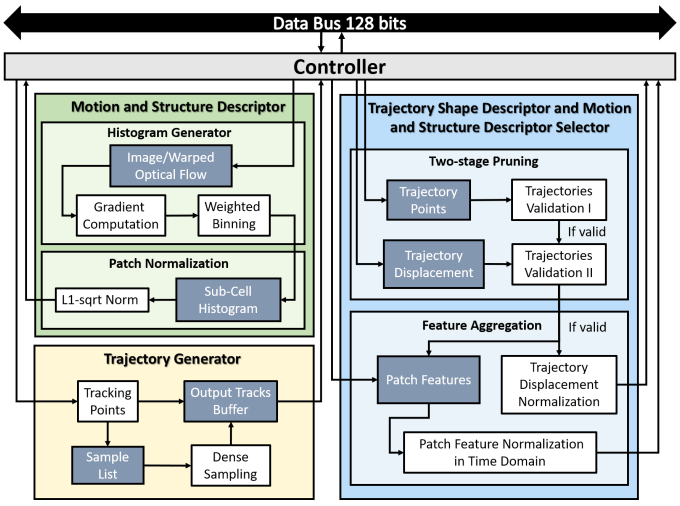


Fig. 1. Proposed architecture of Improved Dense Trajectories.

Deep features [5, 6, 7, 8, 9, 10, 11, 12] are also used for action recognition in recent years. However, the performance of deep features does not outperform hand-crafted features significantly because they are weak at extracting information in time domain. Though many deep features are combined with hand-crafted features for better performance, they can only provide 3.6% mAP improvement compared with using hand-crafted features only. Meanwhile, the computation is 35 times as much, and the power consumption is 10 times as large as hand-crafted features [13]. Hence, deep-learning-based features have not taken down the field of action recognition.

Some hardware architectures were proposed for action recognition [14, 15, 16]. Since features used in these works can only encode motion information spanning at most two frames, none of them can achieve high-performance action recognition in benchmarks with more action classes and more realistic environment (e.g. HMDB51, UCF101, and Hollywood2 datasets).

In this work, considering both performance and hardware

implementation cost, we explore the possibility to implement a trajectory-based feature, iDT, in ASIC. We are the first to propose a hardware-oriented algorithm and hardware architecture of iDT. Our architecture (Fig. 1) can achieve a throughput of 203 fps with only 40.8 kB on-chip memory. Furthermore, it can support not only QVGA videos but also six windows of QVGA resolution in higher resolution videos in real time.

The remainder of this paper is organized as follow. Section 2 overviews the Improved Dense Trajectories algorithm and shows our reconstructed experiment results. Based on iDT, we propose a hardware-oriented algorithm in Section 3, and Section 4 describes the details of our architecture design. In Section 5, we discuss our implementation result. Section 6 concludes the paper.

2. OVERVIEW OF IMPROVED DENSE TRAJECTORIES

Improved Dense Trajectories (iDT) is proposed in [2]. It becomes the state-of-the-art in contributing to its four core concepts, which equip iDT with the strong ability to capture long-term motion information.

Dense Sampling. Feature points are sampled densely in both spatial and temporal domains to ensure that feature points are equally spread over all spatial positions and scales, and motion appearing at any time can be captured.

Trajectories and Local Spatio-temporal features. All feature points will be tracked by dense optical flow filtered by median filter. Points tracked along 15 frames will be concatenated to form a trajectory. Local spatio-temporal features are extracted from the patches centered at the points along each trajectory, including HOG, HOF, MBHx, and MBHy. In the following of this paper, we will call the combination of these four types of local spatio-temporal feature as patch feature.

Camera Motion Elimination. Camera motion affects the correctness of optical flow, and also generates trajectories in the background region. iDT estimates the homography matrix of two consecutive frames, and warps the second frame by the matrix. The optical flow between these two consecutive frames will be re-computed to represent the motion correctly.

Trajectory Pruning Trajectories that cannot represent actions will be pruned since they will introduce erroneous information and degrade the performance.

Our reconstructed experiment uses the HMDB51 dataset, the mean average precision (mAP) is 54.14%. The throughput using single-core CPU is **2.5 fps**, which is almost twelve times less than the minimum requirement (30 fps) for real-time computation. The whole flow of iDT includes the computation of optical flow, warping, trajectories, and descriptors. Since some architecture [17, 18, 19] has been proposed to implement the most time-consuming step (40% of total operating time), optical flow computation, we focus on the architecture design to accelerate the computation of trajectories and

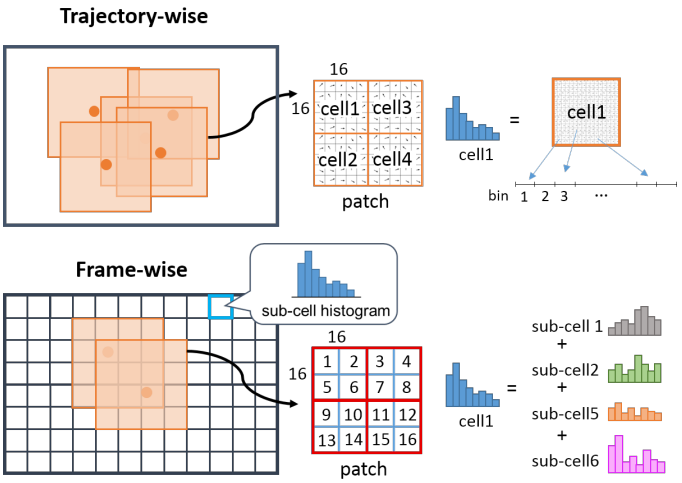


Fig. 2. Illustration of trajectory-wise and frame-wise feature extractions.

descriptors, which has not been addressed before. These two computations are also the core of iDT, and compose the second most time-consuming part (24% of total operating time) of all operations.

3. HARDWARE-ORIENTED ALGORITHM

The most challenging problem in iDT is that local features are extracted based on the trajectories, whose locations are not predictable. This is not suitable for low-cost hardware implementation. We propose a hardware-oriented feature extraction algorithm in frame-wise scheme to take the place of trajectory-wise scheme. The details are introduced in the following subsection.

3.1. Frame-Wise Feature Extraction

The original feature extraction algorithm of iDT is in trajectory-wise scheme. The local spatio-temporal features are extracted along the trajectories. The patches centered at the sample points in a frame heavily overlap, and the locations of the patches are randomly spread. Therefore, the average number of the pixels that need to be processed is almost twenty times the number of pixels in one frame, which introduces high bandwidth and repeated computation in hardware design.

To reduce the bandwidth and repeated computations, we design a histogram reuse scheme based on the commonly seen computation flow of HOG. We first split a frame into a grid of 8×8 sub-cells, and then extract each sub-cell histogram. Once a tracking point shows up, the nearest 16 sub-cells will be used to derive patch histogram, which is composed of four concatenated cell histograms. The derivation of each cell histogram is to simply sum four sub-cell histograms as shown in the bottom of Figure 2. Finally, patch feature can be obtained

after applying L1-sqrt norm on patch histogram. In this way, each pixel histogram needs to be extracted only once, and there will be no repeated data access for overlapping patches. In our implementation, to minimize the bandwidth and computation, we decide to extract all patch features of each frame first and then store them in off-chip memory. These patch features will be loaded in the feature aggregation stage. This can reduce 82% of data I/O and 98% of computation time compared with storing sub-cell histograms in off-chip memory. Extracting all patch features first is also efficient since 12 sub-cell histograms can be reused by neighboring patches.

In our experiment results, there are only 76,800 (320×240) pixels by frame-wise feature extraction and 1,509,011 pixels by trajectory-wise feature extraction to be processed per frame in QVGA videos. Frame-wise feature extraction can reduce the bandwidth usage and save the computation time to 94% while only causes a 0.24% mAP loss compared with trajectory-wise feature extraction.

4. ARCHITECTURE DESIGN

Our proposed architecture is shown in Figure 1, and is composed of three parts. The first part deals with Motion and Structure Descriptor (MSD), including HOG, HOF, MBHx, and MBHy. The second part is Trajectory Generator (TG), which handles point tracking and dense sampling. The third part deals with Trajectory Shape Descriptor and Motion and Structure Descriptors Selector (TSD-MSDS). In TSD-MSDS, erroneous trajectories will be pruned and patch features along 15 frames will be requested and do normalization in time domain for final output features of iDT.

4.1. Low On-chip Memory for Point Tracking

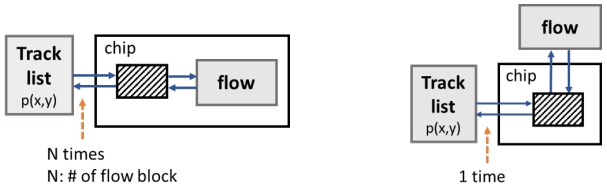
When doing point tracking, all trajectory points that need to be tracked require 141 kB of memory for storage, and the flow data of a frame costs 153.6 kB. There are two types of architecture for point tracking, described below.

The concept of the first architecture is to fully reuse input flow data by splitting flow data in one frame into N blocks. It only need to store one block's flow data on chip, and the memory size depends on N (Fig. 3(a)). However, to track all points, we need to load all trajectory points N times since we have to search for target flow in each block of flow data. On the contrary, in the second architecture, both flow data and trajectory points are stored in off-chip memory (Fig. 3(b)). The flow data will be loaded from off-chip memory for tracking according to the trajectory points' location. The drawback of this architecture is that flow data may be loaded repeatedly since they may belong to more than one trajectory.

To see which architecture can achieve higher throughput and lower bandwidth, we do some numeric analysis shown in Table 1. The analysis focuses on the number of cycles spent on reading and writing off-chip memory.

Table 1. Comparison of off-chip memory access time (cycles) of architecture 1 and architecture 2. The "N" in Architecture 1 represents the number of blocks that the flow data is split into.

	Architecture 1	Architecture 2
Worst case (46,080 pts/frame)	10,971 + 18,432 × N	46,080 + 18,432 = 64512
Normal case (8600 pts/frame)	10,971 + 1720 × 2 × N =17581 (N=2)	8600 + 1720 × 2 =12,040



(a) Architecture 1. (b) Architecture 2.

Fig. 3. Two architectures for point tracking in TG.

The result shows that, in worst case, the first architecture will be faster only if N<3. However, it requires a 76.8 kB on-chip memory to store a block's flow data when N=3. The memory required for flow data is too much for us since our target maximum memory size is 128 kB. In normal case, even if we store the whole frame's flow on chip (153.6 kB), the first architecture is still slower than the second one. Thus, for point tracking in current frame, we choose the second architecture, which requires less data fetching in most of the video sequences and requires no extra on-chip memory.

Note that the number of points in normal case, 8600, is derived from our observation of the 12 videos we randomly choose. On the contrary, that there are 46,080 points in worst case means that all points with a step size of 5 pixels will be sampled in each frame, which seems unlikely to happen.

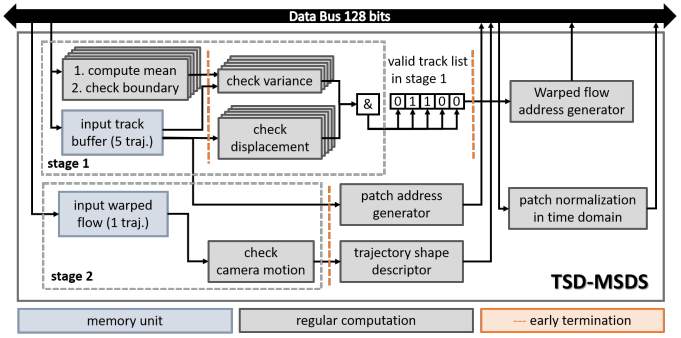


Fig. 4. Architecture of TSD-MSDS.

4.2. Two-Stage Trajectory Pruning

To prune the trajectories, we need to check three requirements. The first requirement is that points should not be beyond the boundary of a frame. The second requirement is that trajectories should not be random, static, or irregular. The third requirement is that trajectories should not be generated from camera motion. The first two requirements can be checked by points in trajectories, but the validation of the third requirement needs extra 15 cycles to load warped flow from off-chip memory. According to our analysis of the distribution of the cause of pruned trajectories, 10% of the pruned trajectories can be detected by the first requirement, and 66% of the pruned trajectories can be detected by the second requirement. This means that loading warped flow for checking the third requirement is unnecessary most of the time. Hence, we decide to split the pruning process into two stages (Fig. 4). In the first stage, we check the first (stage 1-1) and second (stage 1-2) requirements, and valid trajectories will be marked in a list. Once the first stage is finished, only marked trajectories in the list will proceed to stage 2. If a trajectory passes the second stage, it will go on to request its patch feature for final feature normalization (MSDS).

Early terminations in our two-stage trajectory pruning can save 30% of the time spent on loading warped flow for checking the third requirement in parallel pruning, where all three requirements are checked simultaneously.

5. HARDWARE IMPLEMENTATION RESULT

We implement our design using TSMC 40nm technology. Table 2 summarizes the final specification of our implementation. The chip area is 3.1 mm^2, and the throughput is 127 fps in worst case and 203 fps in normal case when operating under the frequency of 215 MHz. The total gate count is 476.42k for logic. The size of on-chip memory is 40.8 kB. The bandwidth is 2.4 GB/s. Such a high bandwidth is mainly introduced by loading and writing patch features, especially when collecting patch features in the past 15 frames. The performance of our chip on the HMDB51 dataset is **51.34%** mAP, where a 3% mAP drop in our reconstructed experiment is introduced by our hardware-oriented algorithm. The final chip layout is shown in Figures 5.

With our final specification, we can also deal with videos in higher resolution. For videos with a single target, we resize each frame to a width of 320 pixels since tasks related to classification do not require high-resolution frame for better performance, which is proved by [20]. For videos with multiple targets, we can propose N windows containing humans, and then resize them to a width of 320 pixels. A possible robust method for window proposal is YOLO [21]. In this way, we can achieve 31.75 fps for N=4 in worst case, and 33.8 fps for N=6 in normal case.

Since there is no existing work implementing trajectory-based features, where features are extracted from more than 10 frames in general, we can only compare our final specification with software implementation. Our software platform is a PC with a 3.40 GHz quad-core CPU and 20 GB RAM. However, we only use a single core of the CPU. The result shows that the throughput of our design is 81.2 times as large, and the power efficiency is 13,340 times as large.

Table 2. Final chip specification. "*" represents normal case.

	iDT
Technology	TSMC 40nm
Chip Area	1.76 mm $\times$ 1.76 mm = 3.1 mm^2
Frequency	200 MHz
Gate Count	476.42k
Memory	40.8 kB
Bus Width	128 bits
Throughput (fps)	127 / 203 *
Throughput (track/s)	390k / 624k *
Bandwidth	2.4 GB/s
Power Consumption	58.44 mW

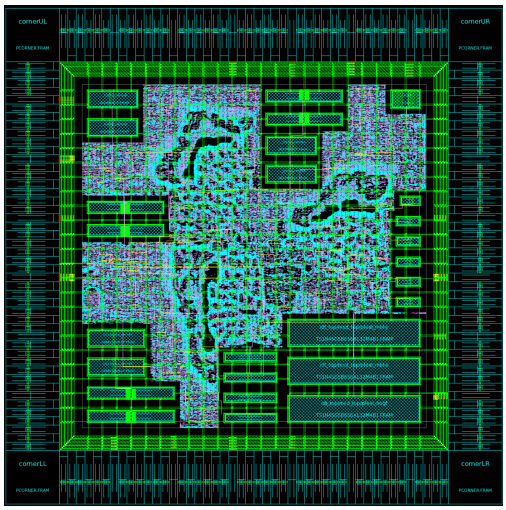


Fig. 5. Chip layout of Improved Dense Trajectories.

6. CONCLUSION

We are the first to propose a hardware-friendly algorithm and an ASIC implementation of Improved Dense Trajectories for real-time action recognition.

With our high throughput, low on-chip memory and computation flexibility, our chip can be applied to many real-time applications on mobile devices or be combined with deep learning engines to implement state-of-the-art action recognition systems.

7. REFERENCES

[1] H. Wang, A. Klser, C. Schmid, and C. L. Liu, "Action recognition by dense trajectories," in *CVPR 2011*, 2011, pp. 3169–3176.

[2] Heng Wang and Cordelia Schmid, "Action recognition with improved trajectories," in *The IEEE International Conference on Computer Vision (ICCV)*, December 2013.

[3] Xiaojiang Peng, Changqing Zou, Yu Qiao, and Qiang Peng, *Action Recognition with Stacked Fisher Vectors*, pp. 581–595, Springer International Publishing, Cham, 2014.

[4] B. Fernando, E. Gavves, J. Oramas M., A. Ghodrati, and T. Tuytelaars, "Rank pooling for action recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 773–787, 2017.

[5] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman, "Convolutional two-stream network fusion for video action recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[6] César Roberto de Souza, Adrien Gaidon, Eleonora Vig, and Antonio Manuel López, *Sympathy for the Details: Dense Trajectories and Hybrid Classification Architectures for Action Recognition*, pp. 697–716, Springer International Publishing, 2016.

[7] Hakan Bilen, Basura Fernando, Efstratios Gavves, Andrea Vedaldi, and Stephen Gould, "Dynamic image networks for action recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[8] Amlan Kar, Nishant Rai, Karan Sikka, and Gaurav Sharma, "Adascan: Adaptive scan pooling in deep convolutional neural networks for human action recognition in videos," *CoRR*, vol. abs/1611.08240, 2016.

[9] Limin Wang, Yu Qiao, and Xiaoou Tang, "Action recognition with trajectory-pooled deep-convolutional descriptors," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[10] Junjie Cai, Michele Merler, Sharath Pankanti, and Qi Tian, "Heterogeneous semantic level features fusion for action recognition," in *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*, 2015, ICMR '15, pp. 307–314.

[11] J. Wang, A. Cherian, and F. Porikli, "Ordered pooling of optical flow sequences for action recognition," in *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017, pp. 168–176.

[12] Y. Shi, Y. Tian, Y. Wang, and T. Huang, "Sequential deep trajectory descriptor for action recognition with three-stream cnn," *IEEE Transactions on Multimedia*, vol. 19, no. 7, pp. 1510–1520, 2017.

[13] Amr Suleiman, Yu-Hsin Chen, Joel S. Emer, and Vivienne Sze, "Towards closing the energy gap between HOG and CNN features for embedded vision," *CoRR*, vol. abs/1703.05853, 2017.

[14] Hongying Meng, Michael Freeman, Nick Pears, and Chris Bailey, "Real-time human action recognition on an embedded, reconfigurable video processing architecture," *Journal of Real-Time Image Processing*, vol. 3, no. 3, pp. 163–176, 2008.

[15] Zuoxun Hou, Hongbo Zhu, Nanning Zheng, and Tadashi Shibata, "A single-chip 600-fps real-time action recognition system employing a hardware friendly algorithm," in *Circuits and Systems (ISCAS), 2014 IEEE International Symposium on*. IEEE, 2014, pp. 762–765.

[16] Xiaoyin Ma, Jose Rodriguez Borbon, Walid Najjar, and Amit K Roy-Chowdhury, "Optimizing hardware design for human action recognition," in *Field Programmable Logic and Applications (FPL), 2016 26th International Conference on*. IEEE, 2016, pp. 1–11.

[17] H. S. Seong, C. E. Rhee, and H. J. Lee, "A novel hardware architecture of the lucas–kanade optical flow for reduced frame memory access," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 6, pp. 1187–1199, 2016.

[18] K. Seyid, A. Richaud, R. Capoccia, and Y. Leblebici, "Fpga based hardware implementation of real-time optical flow calculation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. PP, no. 99, pp. 1–1, 2016.

[19] Gokhan Koray Gultekin and Afsar Saranli, "An fpga based high performance optical flow hardware design for computer vision applications," *Microprocessors and Microsystems*, vol. 37, no. 3, pp. 270 – 286, 2013.

[20] Heng Wang, Dan Oneata, Jakob Verbeek, and Cordelia Schmid, "A robust and efficient video representation for action recognition," *International Journal of Computer Vision*, vol. 119, no. 3, pp. 219–238, 2016.

[21] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi, "You only look once: Unified, real-time object detection," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.