1000 fps Visual Servoing on the Reconfigurable Wide SIMD Processor

Yifan He, Zhenyu Ye, Dongrui She, Roel Pieters, Bart Mesman, Henk Corporaal
Eindhoven University of Technology, the Netherlands
{y.he, z.ye, d.she, r.s.pieters, b.mesman, h.corporaal}@tue.nl

Keywords: Visual Servoing, FFOS, Reconfiguration, Wide SIMD

Abstract

Visual servoing has been proven to obtain better performance than encoders at comparable cost. However, the often computationally intensive vision algorithms and the ever growing demands for higher frame rate make its realization very challenging. This paper demonstrated the feasibility of achieving high frame-rate visual servoing applications on the wide Single-Instruction-Multiple-Data (SIMD) processors. An industrial application, Organic-Light-Emitting-Diode (OLED) substrate localization, is chosen as a typical example. Firstly, we proposed an improved vision pipeline for repetitive structure localization, which is more robust, and also more friendly to embedded processors and FPGA/ASIC realization. Then, a highly-efficient SIMD processor for vision servoing is also proposed. The number of processing elements (PEs) in this proposed SIMD processor is dynamically reconfigurable, which (i) enables efficient processing for different vector lengths; (ii) can easily adjust to the various performance requirements; and (iii) saves energy consumption. For input frames of 120 × 45 resolution, the proposed vision pipeline can process a frame within 275 µs, sufficient to meet the throughput requirement of 1000 fps with a latency of less than 1 ms for the whole vision servoing system. Compared to the reference realization on MicroBlaze, the proposed SIMD processor achieves a 18× performance improvement.

1 Introduction

Visual servoing applies image sensors instead of encoders to acquire the position of components being controlled. On one hand, it reduces the number and accuracy requirement of encoders, and has been proven to obtain better performance than encoders at comparable cost in several applications, e.g., inkjet printing [2, 9]. On the other hand, it also dramatically increases the computing workload due to the often computationally intensive vision algorithms. The ever growing demand for higher frame rate makes the realization of visual servoing systems even more challenging [3, 4].

Among the stream processors for vision/image processing, wide Single-Instruction-Multiple-Data (SIMD) processors are very popular [1, 6, 7] because (i) the massive number of processing elements (PEs) inside an SIMD processor potentially renders very high throughput; (ii) massive parallelism in streaming applications typically shows up as data-level parallelism (DLP) which is naturally supported by SIMD architectures; and (iii) SIMD is a low-power architecture as it applies the same instructions to all PEs. All these merits make SIMD architecture a very interesting candidate for visual servoing.

In order to demonstrate the feasibility of achieving high frame-rate visual servoing on SIMD processor, an industrial application, Organic-Light-Emitting-Diode (OLED) substrate localization, is analyzed in detail. In visual servoing systems, encoders are typically sampled at 1 kHz [10]. To ensure the stability, we set the same sample rate (1000 fps) as the basic requirement for our visual servoing system.

Firstly, we improved the existing Fast-Focus-On-Structures (FFOS) algorithm for OLED center detection [9]. The proposed vision pipeline is not only more robust, but also more friendly to embedded processors and FPGA/ASIC realization. The proposed algorithm only uses 32-bit fixed-point operations while rendering sub-pixel accuracy. Moreover, the processing time is deterministic, which is crucial for the latency oriented applications. After developing the vi-
sual pipeline, a highly-efficient SIMD architecture for vision servoing applications is also proposed, which is based on our previous design [6]. The number of processing elements (PEs) in this proposed SIMD processor can be dynamically reconfigured to match the resolution of the input frame and/or the performance requirement of the application. For input frames of $120 \times 45$ resolution, the proposed vision pipeline can process a frame within $275 \mu s$, sufficient to meet the throughput requirement of 1000 fps with a latency of less than $1 ms$ for the whole vision servoing system. Compared to the reference realization on MicroBlaZe [12], the proposed SIMD processor achieves a $18 \times$ performance improvement.

The remainder of this paper is organized as follows. Section 2 shows the visual servoing setup and our proposed vision pipeline. In Section 3, the proposed SIMD processor for visual servoing applications is presented in details. Section 4 elaborates the realization of the proposed OLED center detection algorithm on our SIMD processor. Finally, Section 5 draws the conclusions of this work.

2 Visual Servoing & Algorithm Development

The visual servoing setup, which is described in [2], is shown in Fig. 1. The camera and lights are fixed at the top of the setup. The OLED structures are mounted on the moving X-Y table, which moves in 1.5 dimension [2]. The system architecture is shown in Fig. 2. At each frame interval, the camera takes an image of the moving OLED structures. The image is then transferred to the vision processing platform through Ethernet or Camera Link interface. In the vision processing step, which is the main focus of this paper, the vision processing platform processes the input image and localizes the centers of the OLED structures. The data-acquisition is realized by using EtherCAT, where DAC, I/O, and ADC modules are installed to drive the current amplifiers of the motors. Based on the relative positions of the detected OLED centers, X-Y table is then driven to a proper position.

Latency is a key parameter in control systems. The delay in our visual servoing system mainly consists of four parts, which are $(i)$ exposure time; $(ii)$ image readout; $(iii)$ vision pipeline; and $(iv)$ control. To reduce the delay of the system, only the Region Of Interest (ROI) of the image taken by the camera is read out and processed. A typical size of ROI for our OLED substrate localization application is $120 \times 45$ pixels or $160 \times 55$ pixels. The exposure time and image readout time is fixed for a specific ROI size, lighting source, and camera interface. The timing of the control part is also fixed given a specific mechanical setup. Therefore, the major source of reduction in the delay can only come from the vision processing part. In order to achieve 1000 fps throughput as well as $1 ms$ latency, the timing budget remaining for vision processing is only about $350 \mu s$ for a $120 \times 45$ resolution input (Fig. 3).

In order to meet this tight budget, we firstly proposed a robust and efficient vision pipeline (Fig. 4). This vision pipeline is based on a previous PC-based implementation [9]. However, the previous PC-based implementation is suboptimal for embedded processors and FPGA/ASIC realization. On one hand, the using of contour tracing algorithm in the previous vision pipeline has several drawbacks: $(i)$ the processing time is not deterministic; $(ii)$ less robust due to higher false detection rate; and $(iii)$ less efficient on parallel architectures. Though there exists algorithms to perform contour tracing on SIMD ar-
architectures [11], the computational load of this algorithm is still too heavy to reach 1000 fps. By exploiting the properties of repetitive structures, this paper proposed an erosion-projection method to replace the contour tracing, which nicely solved the aforementioned issues. On the other hand, the floating point operations are usually too costly for embedded processor and dedicated hardware realization. Therefore, only 32-bit fixed-point operations are used in our proposed vision pipeline, yet still rendering sub-pixel accuracy.

In the proposed vision pipeline, when an input frame is received, the best threshold for binarization is calculated with the OTSU algorithm[8]. After binarization, noise and unrelated patterns are removed through the erosion step, remaining only the dominant structures (i.e., OLEDs). The number of erosion iterations is determined by the feature to be detected, the size of the unrelated patterns, and the quality of the picture. In our case, two iterations are applied to a 120 × 45 input frame. After erosion, horizontal and vertical projections are performed to generate the row and column vectors. Rough center and bonding box of each OLED are then obtained by a post-processing on these two vectors. The accurate OLED center is finally calculated by the weighted center-of-gravity inside each bonding box. At the end of the vision pipeline, the centers of the OLED structures are localized, as shown in Fig. 5.

This new approach reduces the complexity of the algorithm, provides improved robustness, and is also more friendly to embedded processors and dedicated hardware realization. The details of the vision pipeline implementation will be discussed in Section 4.

The proposed reconfigurable wide SIMD processor for visual servoing is based on XetalPro, our ultra low-energy and high-throughput SIMD processor [6]. The block diagram is indicated in Fig. 6. The control processor (CP) is a 32-bit MIPS-like processor, equipped with a 32-bit 1-cycle multiplier and a 32-bit 16-cycle pipelined divider. The main task of the CP is to control the program flow, to handle interrupts, to configure other blocks, and to communicate with the outside world. The processing elements (PEs) and their corresponding scratchpad memory (SM) and frame memory (FM) banks are partitioned into tiles. Each tile consists of 8 PEs. This is based on the reconfiguration granularity requirement as well as the layout constraints. In the current implementation, there are 320 PEs in total (40 tiles). The 128bit×1024 pseudo-dual port SRAM per PE constitutes the frame memory (FM). The relatively large capacity of the FM allows on-chip storage of multiple VGA frames or images with higher resolution. The SM is a 32-entry scratchpad memory to exploit the often available data locality and to reduce the energy consumption of accessing the large FM. The communication network between PEs and SMs enables each PE to directly access data from its left and right neighbors. The whole system runs at 125 MHz with 1.2 V supply voltage under TSMC 65 nm technology, and offers a peak throughput of 80 GOPS (counting multiply and add operations only).

With 320 PEs, the proposed SIMD processor is able to provide a tremendous amount of processing power. However, in practice, applications may not require or cannot fully utilize its entire capability. For example, in our OLED substrate localization application, two typical image resolutions are 120 × 45 pixels and 160 × 55 pixels, which lead to natural vector sizes of 120 and 160 respectively. Thus, only 120 PEs (15 tiles) or 160 PEs (20 tiles) is required for each case. The processor must be configured in such a way that the num-

![Figure 4: Proposed Vision Pipeline of Fast Focus On Structures (FFOS)](image1)

![Figure 5: OLED Structures and Detected Centers](image2)

![Figure 6: Block Diagram of the Reconfigurable Wide SIMD Processor](image3)
number of active PEs and the corresponding communication network meet this requirement. Another important motivation for a reconfigurable SIMD processor is the power consumption. When not all PEs are required, the unused PEs can be fully shut down to save power consumption.

The number of active tiles of the proposed SIMD processor is dynamically configurable to meet various vector lengths or performance requirements. In order to enable this feature, two types of tiles are designed, which are only of slight difference. The Basic Tile (Fig. 7(a)) composes the minimal system (8-PE SIMD) when $MUX_0$ is configured to choose only among immediate number ($imm$), constant '0', and data read from $PE_I$'s own scratchpad memory. The Augmented Tiles (Fig. 7(b)) can be enabled/disabled according to the application. To configure an SIMD processor with $M+1$ tiles, $MUX_0 \sim MUX_{M-1}$ are fixed to their right neighbor (i.e., data from next $PE_0$'s SM) and $MUX_M$ has the freedom to choose among the other three inputs except its right neighbor. The configuration is done through setting the control registers ($CTRL_0 \sim CTRL_M$) by CP at run time.

Each PE has a two-stage pipeline and shares the instruction fetch and decode stage of the CP. Fig. 8 shows the structure of the 16-bit PE, which consists of a local register (ACCU) for immediate result feedback, a shadow register for selectively copying the data in ACCU, and a flag register (FLAG) for conditional instruction execution. Each PE supports 16-bit ADD/SUB, MUL, MAC, logical operations, which can be further compounded with other operations (e.g., absolute, negative). All instructions are executed in a single cycle. The global sum of the ACCU registers (one per PE) is calculated by an adder tree. The latency of the adder tree is three cycles. Another main differences between this proposed SIMD and our previous Xcel-Pro is that local indirect addressing is supported with the local address generator. It has been shown that local indirect addressing could significantly improve the performance of some applications (e.g., Histogram, Hough Transform) in a massively-parallel SIMD processor [5].

4 Visual Servoing on the Proposed Wide SIMD Processor

Before implementing the vision pipeline on the proposed wide SIMD processor, we first analyzed the sequential reference implementation on MicroBlaze. Based on the analysis of the kernels in the vision pipeline, the parts that can benefit from vectorization are identified. With this information, the kernels of the vision pipeline are partitioned into (i) the vector part, which is executed on the PE array of the proposed SIMD processor; and (ii) the scalar part, which is processed on the CP of the proposed SIMD processor. The result shows that it is feasible to achieve > 1000 fps and < 1 ms latency visual servoing on the proposed wide SIMD processor.

4.1 MicroBlaze Implementation

MicroBlaze is a simple soft-core with a RISC instruction set [12]. It is chosen because it represents the typical general purpose processors (GPP) in embedded systems. Also MicroBlaze is configurable and easy to verify on FPGA. The MicroBlaze used here has the following configuration:

- 5-stage pipeline.
• 32-bit multiplier and 32-bit divider.
• All instructions and data are in local memory with one cycle latency.

For a 120 × 45 resolution input frame, the execution time of the MicroBlaze implementation is almost 5 ms at 125 MHz, which is over 14 times of the 350 µs budget available for vision processing. Fig. 9 shows the cycle count breakdown of the vision pipeline. Over 95% of the total execution time is spent on five kernels: histogram + CH/CIA (Cumulative Histogram and Cumulative Intensive Area), binarization, erosion, find-rough-center, and weighted center-of-gravity. The computation part of these kernels are mostly pixel-wise operations with few dependency, which makes them very good candidates for vectorization.

4.2 Wide SIMD Implementation

There are two main sources of inefficiency in the sequential implementation. Firstly, the fundamental limitation in operation throughput makes it impossible to achieve high performance. Secondly, the overheads such as loop control and address calculation greatly reduce the effective computational throughput. On the proposed architecture, the PE array provides a peak throughput of 2 × num_of_PEs operations per cycle if MAC instruction is utilized. In addition, the concurrent execution of the CP and PE array exploits the instruction-level parallelism (ILP) and reduces the time for overhead operations.

The distribution of kernels between PE array and CP is shown in Fig. 10. With this mapping, the data communication between CP and PE array is minimized. To process an image of size $w \times h$, where $w$ is the image width and $h$ is the image height, $w$ PEs are required (thus $w/8$ tiles are enabled in our proposed SIMD processor), which indicates that the PE array can provide a peak throughput of $2w$ operations per cycle. When a frame is captured and ready for processing, the processor uses the input shift register to get the frame to the frame memory (FM) line by line, as illustrated by Fig. 11. Each PE processes one column of the frame. The unused tiles are shut down to save power. Each PE needs $2h + 512$ FM entries to process a column ($h$ entries for the space of the grey-level input image, $h$ entries for the binary image, 256 entries for the shared space of the distributed histogram and CH, and 256 entries for the space of distributed CIA). The typical frame size in our case is 120 × 45 or 160 × 55, so the capacity of the FM is sufficient to process the whole frame. The memory mapping of the frame memory for a 120 × 45 resolution input is shown in Fig. 12.
4.2.1 Histogram and CH/CIA

Algorithm 1: Histogram and CH/CIA

Input: \( w \times h \) image at \( i_{\text{base}} \) in FM
Output: \( CH/\text{CIA} \) in CP

\begin{algorithm}
\begin{algorithmic}[1]
\State \( h_{\text{base}} \leftarrow \) DCH base address in FM
\State \( r_{\text{base}} \leftarrow \text{DCIA base address in FM on PE} \)
\State \text{mem on PE 0 to } w-1 \text{ do}
\For {i \leftarrow 0 \text{ to } 255}
\State // Initialization
\EndFor
\For {i \leftarrow 1 \text{ to } h-1}
\For {j \leftarrow 0 \text{ to } 255}
\State \( w_{\text{addr}}, \text{accu} \leftarrow \text{mem}[i_{\text{base}} + i] + h_{\text{base}} \)
\EndFor
\EndFor
\For {i \leftarrow 0 \text{ to } 255}
\State \( \text{mem}[i_{\text{base}}] \leftarrow 0 \)
\EndFor
\For {i \leftarrow 1 \text{ to } 255}
\State \( \text{accu} \leftarrow \text{mem}[\text{ch}_{\text{base}} + i] \)
\EndFor
\For {i \leftarrow 0 \text{ to } 252}
\State \( \text{accu} \leftarrow \text{accu} + 3 \)
\EndFor
\State \( \text{CP: CH}[255] \leftarrow \text{get accu sum}() \)
\State \( \text{CP: CH}[254] \leftarrow \text{get accu sum}() \)
\State \( \text{CP: CH}[253] \leftarrow \text{get accu sum}() \)
\State \( \text{accu} \leftarrow \text{mem}[\text{cia}_{\text{base}} + 0] \)
\State \( \text{accu} \leftarrow \text{mem}[\text{cia}_{\text{base}} + 1] \)
\State \( \text{accu} \leftarrow \text{mem}[\text{cia}_{\text{base}} + 2] \)
\EndFor
\State \( \text{accu} \leftarrow \text{accu} + 3 \)
\State \( \text{CP: CIA}[255] \leftarrow \text{get accu sum}() \)
\State \( \text{CP: CIA}[254] \leftarrow \text{get accu sum}() \)
\State \( \text{CP: CIA}[253] \leftarrow \text{get accu sum}() \)
\end{algorithmic}
\end{algorithm}

4.2.2 Binarization

Binarization is straight forward as shown in Algorithm 2. Each pixel can be processed independently, so the kernel benefits a lot from the DLP provided by the PE array. In addition, the loop control overhead is reduced due to the concurrent execution of CP and PE array, resulting in a speed-up of 312× for a 120 × 45 resolution input.

4.2.3 Erosion

The erosion in Algorithm 3 uses a cross kernel. To calculate one output pixel, a PE needs to get the four neighboring pixels, two of which are located in the neighborhood PEs’ FM. These two pixels can be accessed using the neighborhood communication in the proposed processor. For OLED center detection, erosion is called twice.

As indicated by Fig. 9, the erosion kernel is the most time consuming part in the MicroBlaze implementation. Table 1 shows that after vectorization, speed-up of 274× is achieved for a 120 × 45 resolution input. We can see from the cycle breakdown on the proposed SIMD processor (Fig. 13) that it is no longer the bottleneck.

4.2.4 Row/Column Projection

Row/column projection is part of the finding the rough centers of the OLEDs. The row projection on the proposed architecture is in Algorithm 4. It is done by using the adder tree in the PE array.

Algorithm 5 is the column projection on wide SIMD. Different from row projection, here the projection is performed by the PEs and then the resulting vector is shifted to the CP.
The speed-up of this kernel is limited by the reduction operation in row projection, the sequential shifting in column projection and the serial processing in finding the rough centers.

4.2.5 Weighted Center-of-Gravity

Algorithm 6 shows the calculation of the weighted center of gravity for one bounding box. Each PE uses its PE ID (pe_id) to determine whether it is inside the bounding box and to calculate the moment of the box. Typically, there are nine OLEDs in one frame. The kernel runs for each bounding box separately.

The implementation uses the bounding-box width (40 pixels) as the vector size and achieves a speed-up of 42.5×. This speed-up comes from three factors: DLP, ILP, and the support of MAC instruction.

The performance of the complete implementation for a 120×45 resolution input is shown in Table 1. The wide SIMD implementation is able to achieve a speed-up of 18× (comparing to the reference MicroBlaze implementation), resulting in an execution time of 275 μs, which is well below the 350 μs budget for vision processing. Fig. 13 shows the cycle count breakdown on the proposed architecture. In contrast to Fig. 9, finding max σyB is now most time consuming. Because it is sequential and can only be done on the CP. The only improvement on finding max σyB is the result of reduced divider latency.

Table 2 shows the results of input images with different sizes. We can see that the wide SIMD implementation has very good scalability.

5 Conclusions & Future Work

This paper shows the feasibility of achieving high frame-rate visual servoing application on the massively-parallel SIMD processor through the industrial OLED substrate localization application. With the proposed wide SIMD processor
for vision servoing, we demonstrated that more than 1000 fps is achievable with a latency of less than 1 ms. Compared to the reference realization on MicroBlaze, a 18x reduction on the processing time is gained. The speed-up is mainly achieved by utilizing the often available data-level parallelism and the concurrent execution of scalar processor (CP) and vector processor (PE array). The massive number of dynamically reconfigurable PEs, local indirect addressing, global adder tree, and combined operation execution of the proposed SIMD processor greatly enable the performance improvement for visual servoing applications.

For the future work, we would like to measure and compare the energy consumption in detail. We would also like to enable the fault-tolerance features of our SIMD processor to deal with the increasingly severe manufacturing variability issue.

References


Table 2: Performance at Different Resolutions in Cycles

<table>
<thead>
<tr>
<th>Kernel</th>
<th>120 × 45</th>
<th>160 × 55</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input+Initialize</td>
<td>6680</td>
<td>10080</td>
</tr>
<tr>
<td>OTSU: Hist. &amp; CH/CIA</td>
<td>4047</td>
<td>4097</td>
</tr>
<tr>
<td>Binarization</td>
<td>15840</td>
<td>15840</td>
</tr>
<tr>
<td>Erosion</td>
<td>1038</td>
<td>1278</td>
</tr>
<tr>
<td>Find-Rough-Center</td>
<td>4601</td>
<td>5770</td>
</tr>
<tr>
<td>Weighted Center of Gravity</td>
<td>1971</td>
<td>2295</td>
</tr>
<tr>
<td>Total</td>
<td>34402</td>
<td>39635</td>
</tr>
<tr>
<td>FM lines used</td>
<td>602</td>
<td>622</td>
</tr>
<tr>
<td>Time (125MHz)</td>
<td>275 µs</td>
<td>317 µs</td>
</tr>
</tbody>
</table>
