"University of Turku Technical Reports, No.5 — August 2015"

HARDWARE ACCELARATED VISUAL TRACKING ALGORITHMS

A Systematic Literature Review

Sirpa Korhonen |Leo Sakari |Tero Säntti |Olli Lahdenoja |Teijo Lehtonen

Sirpa Korhonen,
University of Turku, TRC, 20014 Turun yliopisto, Finland,
sirpa.k.korhonen@utu.fi

Leo Sakari,
University of Turku, TRC, 20014 Turun yliopisto, Finland,
leo.sakari@utu.fi

Tero Säntti,
University of Turku, TRC, 20014 Turun yliopisto, Finland,
tero.santti@utu.fi

Olli Lahdenoja,
University of Turku, TRC, 20014 Turun yliopisto, Finland,
olli.lahdenoja@utu.fi

Teijo Lehtonen,
University of Turku, TRC, 20014 Turun yliopisto, Finland,
teijo.lehtonen@utu.fi

www.trc.utu.fi

ISSN 2341-8028 | ISBN 978-951-29-6189-4

Technology Research Center

Turun yliopisto
University of Turku

Abstract

Many industrial applications need object recognition and tracking capabilities. The algorithms developed for those purposes are computationally expensive. Yet, real time performance, high accuracy and small power consumption are essential measures of the system. When all these requirements are combined, hardware acceleration of these algorithms becomes a feasible solution. The purpose of this study is to analyze the current state of these hardware acceleration solutions, which algorithms have been implemented in hardware and what modifications have been done in order to adapt these algorithms to hardware.

Keywords

Hardware acceleration, visual object tracking, optical flow, augmented reality, computer vision

Contents

1 Introduction

Computer vision and augmented reality have become a growing area of research. There are interesting applications in automotive industry, robotics, building and maritime industry. These applications require a capability to recognize objects and track the motion of these objects. Scale Invariant Feature Transform (SIFT) [3], Speeded-Up Robust Features (SURF) [1] and Binary Robust Independent Elementary Features (BRIEF) [2] are examples of feature extraction algorithms developed for this purpose. Lucas-Kanade is an example of optical flow estimation algorithm, which can be used in tracking.

The algorithms used in feature extraction, optical flow estimation and pose calculation are complex and thus require a lot of computing capacity, especially when real time performance is needed. Hardware acceleration is one solution to overcome the computational bottleneck. Graphics Processing Unit (GPU) is a rivaling solution. In the research of hardware acceleration the first step is to try identify the parts of the algorithms which are most suitable to be implemented in hardware. Then the second step is to find the most suitable techniques to adapt the algorithm to hardware.

Feature regocnition and tracking algorithms are rather new research areas and maybe due to this the naming conventions in the publications are not consistent. Therefore, we clarify that in this paper we name the part of algorithms, where the features are recognized from image, as feature detection. Then the phase, where the vectors characterizing the feature are created, is called feature description. The whole procedure including both feature detection and feature description we call feature extraction.

This paper is organized as follows: In Chapter 2 we describe the research process. In Section 2.1 the research objectives and review questions are presented. In Chapter 3 we present analyzes of the selected papers. In Chapter 4 we present the classification of the papers in terms of key characteristics. Finally, in Chapter 5 we discuss the results and in Chapter 6 we present our conclusions and visions for the future.

2 Research process

A systematic literature review is a systematic and repeatable approach to identify and study all relevant research publications on a specific research question or phenomenon. The method consists of literature search, study selection, data extraction and synthesis. In this review seven major publication databases and search engines were used, seen in Figure 1 which illustrates the review process. In total 9682 papers were found. In the first selection round only titles were read and based on this 352 relevant articles were chosen to second round, in which the abstracts were read. 139 articles from this second round were selected to full text reading. All these third round papers were read by at least two persons to select the papers to the final review. In case these two persons had different opinions, a third person read the paper and gave the deciding vote. 32 papers were finally chosen for this review. In all search engines the following search string was used:

[(fpga OR asic OR ic OR chip OR co-processor) AND ("machine vision" OR "computer vision" OR "robotic vision" OR "optical flow" OR "motion flow" OR "motion estimation" OR "augmented reality" OR "feature extraction" OR slam OR surf OR sift OR klt OR "Lucas-kanade")].

Search was performed using header and abstract option. Searches were done in December 2014.

This survey was focused mainly on articles published 2010 or after that. Only 4 of the selected articles were published before 2010, oldest being from 2006. These old papers were included in the review because they presented interesting architectural structures. Otherwise rather new articles were chosen, because they represent implementations done using current-state technologies and the requirements are set according to current applications. Field Programmable Gate Arrays (FPGA), CPUs and GPUs have developed a lot in terms of performance recently. FPGAs' capacities have increased a lot. Similarly, the requirements of applications have increased, as new augmented reality and computer vision applications need better accuracy and real-time functionality with increasing image resolution.

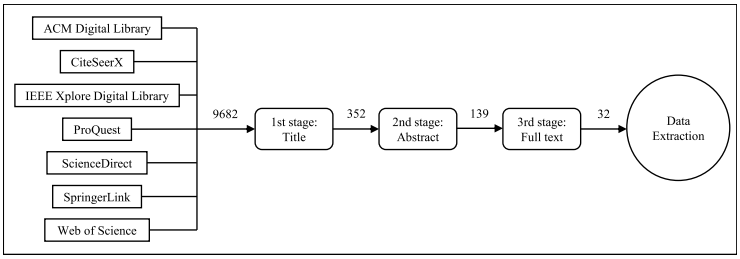

Figure 1: Process to select papers.

2.1 Review questions

In this review the main objective was to find out the current state of hardware acceleration in feature extraction and object tracking algorithms, which can be applied to mixed reality systems. The research questions were

1) What algorithms and which parts of them have been implemented in hardware? How well did the solutions perform?

2) Are there any parts in the investigated algorithms that have not been implemented in hardware, why?

3) Are there any competing solutions to a given problem? What are the benefits of each?

3 Analysis of the material

The selected papers can be divided into two main categories, namely feature extraction algorithms and object tracking algorithms. The feature detector algorithms included are scale space detector, Canny edge detector, Harris corner detection and Features from Accelerated Segment Test (FAST). SIFT, SURF, O-DAISY, BAsis Sparse-coding Inspired Similarity (BASIS) and Massively Parallel Keypoint Detection and Description (MP-KDD) are feature extraction algorithms included in the selected papers. The optical flow estimation algorithm included was Lucas-Kanade and there were articles about tensor based algorithms. Both full and partial hardware (HW) implementations of these algorithms were presented in the papers. Performance comparisons between hardware solution and corresponding GPU and CPU implementations were also presented in many papers as well as comparisons to other HW implementations. In most of the papers the parameters analysed were execution time, image resolution, accuracy and hardware utilization. Power consumption and design time were also discussed in some papers. Design time is interesting to system engineers designing real products.

3.1 Feature extraction algorithms

SIFT is an algorithm for feature extraction. It is a robust algorithm but computationally complex. The most computation intensive part is the Gaussian filtering in the feature detection part of SIFT. In [P7, P19, P29, P31, P32], the detection part has been implemented in FPGA in order to decrease the needed computing capacity. In [P19, P29, P31, P32] the feature description part is implemented on Digital Signal Processor (DSP). When converted to hardware implementation the architecture is often designed to increase parallelism in operations. The mathematical operations are simplified in some papers, e.g. the Gaussian filtering length has been fixed smaller, but then the number of errors is increased. One drawback in HW implementation is that floating point operations are often converted to fixed point, which means some loss in accuracy. It is also stated in many papers that if the operations are copied as such to hardware the amount of hardware resources would not be feasible. In some papers part of the SIFT algorithm has been replaced by some other algorithm. In [P29] Wang et al. have replaced the feature extraction part of SIFT by BRIEF. The authors claim that this combines the stability and repeatability of SIFT with the computational efficiency of BRIEF and thus the real time requirements from real-life

computer vision system are met. In [P27] Suzuki et al. utilize corner detection to reduce the computational complexity of the feature detection part of SIFT. In [P4] Bonato et al. implement the whole SIFT on an embedded FPGA module. In [P14] Jie et al. and in [P22] Qasaimeh et al. implement the whole SIFT in FPGA.

SURF is another widely used algorithm for feature extraction. In [P1, P5, P26] the authors present implementations of full SURF algorithm on FPGA. Krajnik et al. [P16] implemented SURF detection part on FPGA and description is run on an embedded processor module. The implementation is compared to GPU based system. In [P8] Yongsig et al. present an implementation of SURF, where SURF is partitioned to several sub-IPs which are implemented on Xilinx zynq-7020 processing platform. It contains a complete ARM based system with memory interfaces.

3.2 Optical flow algorithms

Optical flow estimation algorithms can be used for tracking the motion of features and objects from image to image. Wei et al. [P30] developed a tensor-based algorithm to be implemented on an FPGA. They report real-time processing performance and good accuracy. Bodily et al. [P3] have also studied tensor based optical flow implementation. The parameters they discussed are performance, cost, power, embedability, memory architecture, flexibility and design productivity. Both algorithms are run in parallel. In [P18] Mahlingan et al. and in [P15] Kalyan et al. present FPGA implementations of Lucas-Kanade. In both of these papers the target has been to develop a VLSI architecture of the algorithm. However, prototyping has been done on an FPGA. Pauwels et al. [P20] have also compared GPU and FPGA implementations of an optical flow estimation algorithm. The parameters they have used are arithmetic complexity, external and on-chip memory access, data dependency, accuracy, speed, power consumption, cost and design time. Their conclusion is that it depends on the application whether GPU or FPGA should be chosen. In [P28] Lucas-Kanade has been implemented on DSP. The system as total consists of Harris corner detector implemented on FPGA and DSP as co-processor running Lucas-Kanade. The main driver for this combination has been to achieve a balance between throughput and development time. DSP suits well for iterative processing, and development time is relatively short. The emphasis in this study was to optimize the whole system, not optimizing Lucas-Kanade performance only.

3.3 Feature detection algorithms

Possa et al. [P21] have implemented feature detection algorithms, namely Canny edge detector and Harris corner detector, on FPGA. For comparison respective algorithms have been implemented on GPU. A new architecture has been developed to reduce latency and memory requirements. The results show that FPGA implementation is competitive with GPU. It is faster in smaller image sizes but becomes slower when image resolution is increased. In [P9] Dohi et al. present FAST corner detector implementation on an FPGA. The advantage of FAST compared to other corner detector methods is that it does not use difference of Gaussians, which makes detector

computationally effective. However, in FPGA implementation the problem is the size of the corner detection look-up table. In this paper the problem is solved by table compression techniques. Harris corner detection HW implementation is presented in [P2] and [P28]. In [P28] Harris corner detector is part of feature tracking system, where Lucas-Kanade optical flow is implemented on DSP.

3.4 Others

Idris et al. [P13] have presented Extended Kalman filter (EKF) HW acceleration. EKF is used in Simultaneous Localization and Mapping (SLAM) for position and motion estimation, where EKF is the most time consuming part and thus the obvious candidate for HW acceleration.

4 Techniques used to adapt the algorithms to hardware implementation

Usually the algorithms are not optimal for hardware implementation as such, especially if the target is to maximize speed of execution and at the same time keep the amount of logic in minimum. Minimizing the amount of logic also implies smaller power consumption, which is important e.g. in mobile and robotic applications. A consequence of adapting algorithm to hardware implementation may be decreased accuracy. Various optimization techniques for keeping accuracy sufficient and at the same time achieve efficient hardware implementation are studied in many of the reviewed articles. On the other hand, decreased accuracy is not always a problem because there maybe applications where the loss is acceptable. Many of the techniques introduced in this chapter are useful for adapting the algorithm to FPGA but they may increase the performance in software implementations also. In general, combining different techniques was quite common in the reviewed papers.

4.1 Data flow architecture modifications

Data flow architecture is an area, in which there are many possibilities for modifications in order to optimize the data flow for hardware implementation. In hardware, e.g. FPGA, it is easy to make massively parallel operations. Thus algorithms which contain extensive data parallelism are good candidates for FPGA implementation. The algorithms included in this study are containing data parallelism and in addition the incoming pixels are processed in such way that it is possible to pipeline the operations to each pixel. In some cases the original algorithm is reformulated in order to take full advantage of the parallelism in hardware. SURF implementations and the applied data flow modifications are presented in [P1, P5, P10, P26]. Examples of SIFT data flow architectures are presented in [P4, P7, P14, P19, P22]. In [P7] the most computationally expensive part of SIFT i.e. the detection of interest points is reformulated by introducing parallel algorithm for scale-space extrema detection. In this study the target has also been to keep the amount of used hardware resources reasonable, and therefore interleaving is applied to processing of

octaves for pyramid processing. Due to interleaving a result is received every two clock cycles, which slightly decreases the performance for this process, but amount of hardware resources as total is decreased because the same unit can be used for other convolotion operations.

For feature description algorithms algorithmic scaling and modifications are used when targeting to hardware implementation. Xiao et al. [P31] use these techniques. Parallelization in data flow architecture is presented by Bonato et al. [P4].

Examples of using parallelism and pipelining in optical flow algorithms are presented in [P15, P18, P30]. In [P30] Wei et al. present a fully pipelined data flow for tensor based optical flow estimation. All steps happen sequentially and there is no iterative processing. Besides the dataflow architecture in this article the trade-off between accuracy and efficiency is studied. The kernel sizes in the three convolution operations (Gradient Calculation, Gradient weighting and Tensor calculation) impact to accuracy and amount of needed hardware resources. Optimum solution can be identified by analysis using e.g. MATLAB simulations.

Kalyan et al. [P15] and Mahalingam et al. [P18] present pipelined dataflow architectures of optical flow algorithms. In [P18] the Lucas-Kanade algorithm implementation the computing of the optical flow happens simultaneously with memory loading of subsequent frames. In this article a lot of emphasis is on methods used for decreasing the loss in accuracy which happens inevitably when fixed point numbers are used instead of floating point representation. Those methods are discussed later in this paper in Section 4.3.

Pipelining reduces the need for intermediate data storing and run time memory accesses. An example of this can be found in [P28], where Tomasi et al. present a hardware implementation of Harris corner detector. In [P23] Schaeferling et al. present a solution in which the main idea is to minimize the repeated memory accesses to same pixels in SURF. In SURF basically a large amount of memory is needed for storing the integral image and for the Hessian matrix calculation in detector phase. In [P1, P8, P17], the authors are presenting architectural modifications which are targeted to reduce the amount of these two memories. In [P10] Fan et al. modify the integral image buffering so that multiple data can be accessed in one clock cycle.

4.2 Algorithm modifications and reformulations

Different kinds of algorithmic modifications and reformulations are also possible. Scaling or even replacing part of the algorithm with totally new algorithm have been proposed. These kind of changes are not always meant only for adapting the system to hardware, but may improve the system performance in any platform. An example is in [P31], where the SIFT feature detection part was implemented in hardware. To adapt SIFT feature detector to hardware parallel Gaussian filtering is is used instead of cascaded filtering. In addition, the number of layers in Gaussian Pyramid and DOG pyramid can be adjusted according to image size and desired FPGA capacity. Similar modifications are common in the reviewed papers. SIFT is not as such highly parallel in data and pipelined in processing, and adjustments are

needed to make feasible hardware implementation. In [P29] Wang et al. replace SIFT feature description and matching with BRIEF respectively. In [P10] Fan et al. present an interesting sliding window method to the detector phase of SURF. Additionally parallelization in data flow architecture and a new method in integral image buffering is used.

4.3 Use of fixed point numbers instead of floating point representation

Computer vision algorithms are basically using floating point representation of numbers, but especially in hardware implementations the fixed point representation is preferred because it helps in keeping the amount of logic feasible. However, the consequence is decreased accuracy. This trade-off between accuracy and hardware efficiency has been discussed in many of the reviewed papers. In [P18] is studied the possibilities to decrease the loss in accuracy when using fixed-point numbers for Lucas-Kanade optical flow algorithm. Every step of L-K algorithm impacts the accuracy, but special attention is put to deviation value in smoothing step and the threshold value parameter during thresholding operation. These parameters were selected after careful analysis based on Yosemite test sequence. Accuracy was also improved by using powers of two coefficients in kernel matrices. Thirdly, the bit widths in each stage were scaled uniquely. This was done in order to be minimize the hardware overhead and accuracy error. Similarly bit width trimming is used in [P30].

In [P19] Mishra et al. propose the use of Look Up Tables (LUT) for SIFT implementation. According to the authors this removes the need of floating point representation in all stages of the algorithm.

4.4 Other techniques

In [P30] the authors state that the highly pipelined dataflow architecture causes the system performance bottleneck to be in the memory accesses. Therefore a multiport memory controller that provides four separate memory ports is used. The external memory access speed problem in FPGA implementations is notified in many of the reviewed papers.

There are several hardware related tricks that can or even should be used when optimizing the hardware implementation. Instead of normal dividers LUT based dividers can be used, especially if target is to save hardware resources. In [P19] LUT's are used in every step of SIFT feature detection part. Bit width trimming in different stages of algorithm is another possibility. This is useful again when it is essential to save hardware resources. Use of state-of-art FPGA technologies which incorporate embedded processors and internal memory enables implementing whole systems on FPGA. However, the final implementation is not pure hardware accelerator, but merely a SW/HW co-design system. However, having the whole system on same chip reduces delays significantly.

5 Performance statistics

This chapter contains the technical details of the reviewed papers. The tables are divided into four different categories according to the implemented algorithm area: optical flow, feature detection, feature description and feature extraction. The tables include the name of the implemented algorithm, the used platform (FPGA, DSP and/or ASIC), the used image resolution and the reported performance of the system. The feature extraction table also includes a column for the operating frequency (or clock rate) of the system.

Because of the amount of papers and the highly varying performance statistics given in the papers, the column *Throughput* was added to all four tables. It contains a generalized performance value for the systems and makes it possible to directly compare the performance of the implementations. Although some papers have provided the throughput value in the desired form (Mpix/s), most of the values in the column have been derived from the reported performance values that can be found in the individual papers.

For a system with horizontal resolution M, vertical resolution N and a frame rate f, the throughput has been calculated as follows:

$$Throughput = N * M * f \tag{5.1}$$

If the frame rate has not been given as FPS, it can be calculated from the delay values. For example, 10 ms processing time equals 100 FPS. If the frame size or the timing details have not been expressed, the throughput can not be calculated.

Out of the 32 selected papers, two focused on a custom ASIC chip design while 30 implemented their design on an FPGA. Four of the FPGA papers also used a DSP as a part of their design. The most common algorithms in the review were SURF and SIFT, both of which were implemented in nine different papers.

Table 1 contains the papers that have implemented optical flow algorithms. Two of the papers [P3, P20] focus on comparing the performance of FPGAs to GPUs. An algorithm that was first introduced in [P30] is also used in [P3]. This can be seen in the identical throughput (19,7 MPix/s) of the two systems.

Paper	Algorithm	Platform	Resolution	Performance	Throughput (Mpix/s)
[P15]	Lucas-Kanade	FPGA	1200x680	500-700 FPS	571,2
[P20]	Gautama	FPGA	Various	Various	53,0[1]
[P28]	Lucas-Kanade	DSP	640x480	160 FPS	49,2
[P18]	Lucas-Kanade	FPGA	640x480	8 ms/frame	38,4
[P30]	Custom*	FPGA	640x480	64 FPS	19,7
[P3]	Custom*	FPGA	640x480	64 FPS	19,7
[P24]	KLT	FPGA	N/A	18 FPS	N/A

Table 1: Optical flow
[1] as stated by the authors

Table 2 groups together the papers that have implemented feature detection algorithms. [P21] appears twice in the table because both Canny and Harris detectors were implemented in it. Other algorithms presented in the table are FAST and SIFT. SIFT uses scale-space filtering for obtaining the locations of the keypoints. It is worth mentioning that only the scale-space filtering part of SIFT is implemented in the papers in this table.

Paper	Algorithm	Platform	Resolution	Performance	Throughput (Mpix/s)
[P21]	Canny	FPGA	Various	Various	242,0[1]
[P21]	Harris	FPGA	Various	Various	232,0[1]
[P19]	SIFT	FPGA+DSP	Various	0,932 ms[2]	77,3[2]
[P7]	SIFT	FPGA	320x240	900 FPS	69,1
[P2]	Harris	FPGA	1024x800	76 FPS	62,3
[P28]	Harris	FPGA	Various	Various	60,1[1]
[P4]	SIFT	FPGA	320x240	33 ms	2,3
[P9]	FAST	FPGA	N/A	62,5 FPS	N/A

Table 2: Feature detection
[1] as stated by the authors
[2] fastest resolution

Table 3 contains the two papers that implemented the feature description part of feature extraction. Both of the papers implemented a custom descriptor. Although the implementation in [P12] is faster, the throughput in [P11] is much higher as it uses a higher image resolution.

Paper	Algorithm	Platform	Resolution	Performance	Throughput (Mpix/s)
[P11]	O-DAISY	FPGA	Various	30 FPS[1]	125,0[1]
[P12]	BASIS	FPGA	640x480	60 FPS	18,4

Table 3: Feature description
[1] fastest resolution

Table 4 contains all papers in the survey that implement the whole process of

feature extraction from the acquisition of a raw image to the extracted feature data. While a clear majority of the papers implemented a version of SIFT or SURF, two other algorithms were also chosen for the implementation. The BRIEF descriptor is used together with the SIFT detector in [P29], while a fully custom algorithm, MP-KDD, is used for the full process in [P25].Krajnik et al. [P16] implemented SURF detection part on FPGA and description is run on an embedded processor module. The implementation is compared to a GPU based system. A similar speed but smaller power consumption is reported. The target application was small mobile robot, where the FPGA's smaller power consumption is an advantage.

5.1 Optimization criterias

It is difficult to compare the different designs by performance and throughput. In different studies the optimization criterias are different. In some studies the maximum resolution has been the first priority and very often frame rate has been the most important parameter. On the other hand, in some cases an average 30 or 64 fps frame rate has been used, and optimization is done for other reasons. In those cases it has been assumed that 30 or 64 fps is sufficient for most of the applications and there was seen no reason to look for maximum fps. In many papers target has been to keep the amount of used FPGA resources reasonable. The publication years and used FPGA technologies are presented in Table 5.5. It looks that there has been no major effort in trying to optimize the design to some specific FPGA technology. On the contrary, the technology and FPGA family has been chosen according to common availability and so that the platform does not limit implementation e.g due to too few I/Os.

Paper	Algorithm	Platform	Clock rate (Mhz)	Resolution	Performance	Throughput (Mpix/s)
[P27]	SIFT	FPGA	168	1920x1080	60 FPS	124,42
[P10]	SURF	FPGA	156	640x480	356 FPS	109,36
[P1]	SURF	FPGA	N/A	640x480	3,029 ms total	101,42
[P29]	SIFT/BRIEF	FPGA	159	1280x720	60 FPS	55,30
[P14]	SIFT	FPGA	50	512x512	6,55 ms total	40,02
[P22]	SIFT	FPGA	29	640x480	95,28 FPS	29,27
[P17]	SURF	ASIC	200	640x480	60 FPS	18,43
[P26]	SURF	FPGA	25	640x480	60 FPS	18,43
[P5]	SURF	FPGA	200	640x480	56 FPS	17,20
[P25]	MPKDD	FPGA	50	160x120	600-760 FPS	14,60[2]
[P16]	SURF	FPGA	N/A	1024x768	80 ms total[1]	9,83[1]
[P8]	SURF	FPGA	200	640x480	64,6 ms total	4,76
[P32]	SIFT	FPGA+DSP	107	320x256	18 ms total[1]	4,55[1]
[P8]	SURF	FPGA	200	300x300	23,4 ms total	3,85
[P6]	SURF	ASIC	100	640x480	12,5 FPS	3,84
[P31]	SIFT	FPGA+DSP	27	360x288	0,763 ms/descriptor[1]	1,21[1]
[P23]	SURF	FPGA	50	Various	1020 ms total[3]	0,30[3]

Table 4: Feature extraction
[1] Assuming 100 detected features
[2] at 760 FPS
[3] fastest resolution

Paper	Publ.	Platform	Technology
[P11]	2011	FPGA	
[P12]	2011	FPGA	
[P27]	2012	FPGA	Xilinx Virtex-5
[P10]	2013	FPGA	XILINX XC6CSX475T
[P1]	2012	FPGA	XILINX XC6VSX204T
[P29]	2014	FPGA	XILINX XUPV5-LX110T
[P14]	2014	FPGA	XILINX Virtex-5
[P22]	2014	FPGA	XILINX Virtex-5
[P7]	2013	FPGA	XILINX Virtex-2 Pro
[P4]	2008	FPGA	ALtera STRATIX II
[P17]	2014	ASIC	
[P26]	2012	FPGA	XILINX XC4 s
[P5]	2010	FPGA	XILINX Virtex-5
[P25]	2014	ASIC	
[P16]	2014	FPGA	XILINX Virtex-5
[P8]	2013	FPGA	
[P32]	2013	FPGA+DSP	Xilinx XC4
[P8]	2013	FPGA	XILINX Zynq-7020
[P6]	2014	ASIC	TSMC 65nm process
[P31]	2013	FPGA+DSP	Altera Cyclone III
[P23]	2012	FPGA	XILINX Virtex-5
[P19]	2014	FPGA+DSP	XILINX Virtex-6
[P21]	2014	FPGA	Altera Cyclone IV
[P2]	2014	FPGA	Altera Cyclone III
[P12]	2011	FPGA	XILINX Virtex-6
[P13]	2012	FPGA	
[P9]	2011	FPGA	XILINX Virtex-5
[P30]	2007	FPGA	XILINX Virtex-2 pro
[P18]	2010	FPGA	XILINX Virtex-2 pro
[P15]	2011	FPGA	Altera Cyclon II
[P20]	2012	FPGA	XILINX Virtex-4
[P28]	2014	FPGA	XILINX Virtex-4
[P24]	2006	FPGA	XILINX Virtex-2 pro
[P3]	2010	FPGA and GPU	

Table 5: Implementation platforms

6 Results of the review

The first review question was to find out which algorithms or parts of them have been implemented in hardware. SIFT and SURF have been very popular. These algorithms are robust and accurate and therefore useful in many applications. On the other hand, they are computationally complex and require a lot of computing capacity for real time system performance. This is an obvious reason why they have raised interest to research hardware acceleration for them. Implementations of SIFT are in most cases partial, whereas SURF implementations are usually full hardware implementations. For SIFT the feature/keypoint detection has been the most popular part selected for hardware implementation. The simple reason is that this is the computationally most expensive part of SIFT. Among optical flow estimation algorithms the Lucas-Kanade has been very popular as hardware implementation. It is suitable for hardware implementation because data can be operated highly parallel, whereas as the processing can be pipelined without major algorithmic modifications and compromises in accuracy. Tensor based optical flow algorithms have also been implemented in hardware. In addition, Harris and FAST corner detectors have been implemented in hardware.

In general it can be said that for hardware implementation the most suitable algorithms or parts of them are those which have extensive data parallelism and pipelined processing. Fixed point format is preferred, because it helps in keeping the amount of hw resources feasible. Various optimizations are though possible to reduce it's the impacts to e.g. accuracy. Usually the original software implementations are using floating point presentation.

The second review question was to find out if there are algorithms or parts of them that have not been implemented in hardware. The result was simply, that among those algorithms which were included in this survey, there are no parts that had not been implemented in hardware.

The third question was to find out other techniques to implement these algorithms. The most popular techniques were GPU based and FPGA implementation. Another popular techniques was combining DSP and hardware implementations i.e. part of the algorithms was in FPGA and other part running on DSP. Current most advanced FPGA plalforms offer processor cores and in few cases those were also used. FPGAs were also used as test platforms, when final solution was targeted to ASIC. There were pure CPU solutions presented but only for comparison purposes, so that the performance improvements achieved by hardware acceleration or GPU could be shown.

In Section 3.1 the various techniques used to adapt these algorithms to hardware implementation were discussed. In many papers several techniques were combined. Firstly, dataflow architecture was modified in order to increase parallelism. Secondly, execution was pipelined. Memory architectures and accesses were optimized. Algorithmic scaling and modifications were also researched.

The performance figures collected in Chapter 5 show that hardware acceleration is feasible solution to the computing capacity problems in systems where these algorithms are used. Pauwels et al. [P20] compare FPGA and GPU implementations of optical flow algorithm and conclude that GPU overcomes FPGA in most cases. However, it depends on the application, design time, power consumption requirements and many other factors whether GPU or HW acceleration should be chosen.

7 Conclusions

In this survey it was found that HW acceleration of visual tracking algorithms is a widely studied field. In many of the reviewed papers an implementation of a one algorithm was studied and the optimization criteria where clearly determined. But there were also articles which were studying full systems including feature detection, description and tracking. Optimization criteria were partly different in these two cases. In system level optimizations throughput and design time were of interest. In articles concentrating on specific algorithm, the criteria were accuracy, hardware resource utilization and speed or performance. Power consumption was described only qualitatively, though in a few papers measured results were presented. It is difficult to present feasible comparisons between different hardware implementations and between GPU and FPGA implementations because the used technologies, FPGA family, GPU platforms impact signifigantly to the power consumption. Many different techniques have been used to adapt the algorithms to hardware. However, new demanding application areas are coming all the time. Image resolutions are growing and at the same time the requirements for accuracy, small power consumption and real time processing remain unchanged. Due to this HW acceleration of visual tracking algorithms is an important research area in future and new solutions are needed. Possible research topics can be SMART cameras i.e systems where are markable part of the data processing is happening in the camera chip. Another research topic could be looking for possibilities to utilize various sensors like gyroscopes and accelerometers in pose estimation and tracking phase.

Acknowledgements

The research has been carried out during the MARIN2 project (Mobile Mixed Reality Applications for Professional Use) funded by Tekes (The Finnish Funding Agency for Innovation) in collaboration with partners; Defour, Destia, Granlund, Infrakit, Integration House, Lloyd's Register, Nextfour Group, Meyer Turku, BuildingSMART Finland, Machine Technology Center Turku and Turku Science Park. The authors are from Technology Research Center, University of Turku, Finland.

Bibliography

Selected papers

[P1] N. Battezzati et al. "SURF algorithm in FPGA: A novel architecture for high demanding industrial applications". In: *Design, Automation Test in Europe Conference Exhibition (DATE), 2012*. 2012, pp. 161–162.

[P2] Merwan Birem and François Berry. "DreamCam: A modular FPGA-based smart camera architecture". In: *Journal of Systems Architecture* 60.6 (2014), pp. 519 –527. issn: 1383-7621.

[P3] John Bodily et al. "A Comparison Study on Implementing Optical Flow and Digital Communications on FPGAs and GPUs". In: *ACM Trans. Reconfigurable Technol. Syst.* 3.2 (2010), 6:1–6:22. issn: 1936-7406.

[P4] V. Bonato, E. Marques, and G.A. Constantinides. "A Parallel Hardware Architecture for Scale and Rotation Invariant Feature Detection". In: *Circuits and Systems for Video Technology, IEEE Transactions on* 18.12 (2008), pp. 1703–1712. issn: 1051-8215.

[P5] D. Bouris, A. Nikitakis, and I. Papaefstathiou. "Fast and Efficient FPGA-Based Feature Detection Employing the SURF Algorithm". In: *Field-Programmable Custom Computing Machines (FCCM), 2010 18th IEEE Annual International Symposium on*. 2010, pp. 3–10.

[P6] ShanShan Cai et al. "Optimization of speeded-up robust feature algorithm for hardware implementation". English. In: *Science China Information Sciences* 57.4 (2014), pp. 1–15. issn: 1674-733X.

[P7] Leonardo Chang et al. "FPGA-based detection of SIFT interest keypoints". English. In: *Machine Vision and Applications* 24.2 (2013), pp. 371–392. issn: 0932-8092.

[P8] Yong-Sig Do and Yong-Jin Jeong. "A new area efficient SURF hardware structure and its application to Object tracking". In: *TENCON 2013 - 2013 IEEE Region 10 Conference (31194)*. 2013, pp. 1–4.

[P9] K. Dohi et al. "Pattern Compression of FAST Corner Detection for Efficient Hardware Implementation". In: *Field Programmable Logic and Applications (FPL), 2011 International Conference on*. 2011, pp. 478–481.

[P10] Xitian Fan et al. "Implementation of high performance hardware architecture of OpenSURF algorithm on FPGA". In: *Field-Programmable Technology (FPT), 2013 International Conference on*. 2013, pp. 152–159.

[P11] J. Fischer et al. "A rotation invariant feature descriptor O-DAISY and its FPGA implementation". In: *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. 2011, pp. 2365–2370.

[P12] Spencer G Fowers et al. *Nature-Inspired BASIS Feature Descriptor and Its Hardware Implementation*. 2011.

[P13] Mohd. Yamani Idna Idris et al. "A co-processor design to accelerate sequential monocular SLAM EKF process". In: *Measurement* 45.8 (2012), pp. 2141 –2152. issn: 0263-2241.

[P14] Jie Jiang, Xiaoyang Li, and Guangjun Zhang. "SIFT Hardware Implementation for Real-Time Image Feature Extraction". In: *Circuits and Systems for Video Technology, IEEE Transactions on* 24.7 (2014), pp. 1209–1220. issn: 1051-8215.

[P15] T.R.S. Kalyan and M. Malathi. "Architectural implementation of high speed optical flow computation based on Lucas-Kanade algorithm". In: *Electronics Computer Technology (ICECT), 2011 3rd International Conference on*. Vol. 4. 2011, pp. 192–195.

[P16] Tomáš Krajník et al. "FPGA-based module for SURF extraction". English. In: *Machine Vision and Applications* 25.3 (2014), pp. 787–800. issn: 0932-8092.

[P17] Sang-Seol Lee et al. "Memory-efficient SURF architecture for ASIC implementation". In: *Electronics Letters* 50.15 (2014), pp. 1058–1059. issn: 0013-5194.

[P18] V. Mahalingam et al. "A VLSI Architecture and Algorithm for Lucas Kanade-Based Optical Flow Computation". In: *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on* 18.1 (2010), pp. 29–38. issn: 1063-8210.

[P19] P. Mishra et al. "Embedded hardware architectures for scale and rotation invariant feature detection". In: *Electronics, Computing and Communication Technologies (IEEE CONECCT), 2014 IEEE International Conference on*. 2014, pp. 1–6.

[P20] K. Pauwels et al. "A Comparison of FPGA and GPU for Real-Time Phase-Based Optical Flow, Stereo, and Local Image Features". In: *Computers, IEEE Transactions on* 61.7 (2012), pp. 999–1012. issn: 0018-9340.

[P21] P.R. Possa et al. "A Multi-Resolution FPGA-Based Architecture for Real-Time Edge and Corner Detection". In: *Computers, IEEE Transactions on* 63.10 (2014), pp. 2376–2388. issn: 0018-9340.

[P22] M. Qasaimeh, A. Sagahyroon, and T. Shanableh. "A parallel hardware architecture for Scale Invariant Feature Transform (SIFT)". In: *Multimedia Computing and Systems (ICMCS), 2014 International Conference on*. 2014, pp. 295–300.

[P23] Michael Schaeferling, Ulrich Hornung, and Gundolf Kiefer. "Object Recognition and Pose Estimation on Embedded Hardware: SURF-based System Designs Accelerated by FPGA Logic". In: *Int. J. Reconfig. Comput.* 2012 (2012), 6:6–6:6. issn: 1687-7195.

[P24] J. Schlessman et al. "Hardware/Software Co-Design of an FPGA-based Embedded Tracking System". In: *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW '06. Conference on.* 2006, pp. 123–123.

[P25] Cong Shi et al. "A massively parallel keypoint detection and description (MP-KDD) algorithm for high-speed vision chip". English. In: *Science China Information Sciences* 57.10 (2014), pp. 1–12. issn: 1674-733X.

[P26] T. Sledevic and A. Serackis. "SURF algorithm implementation on FPGA". In: *Electronics Conference (BEC), 2012 13th Biennial Baltic.* 2012, pp. 291–294.

[P27] T. Suzuki and T. Ikenaga. "SIFT-based low complexity keypoint extraction and its real-time hardware implementation for full-HD video". In: *Signal Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012 Asia-Pacific.* 2012, pp. 1–6.

[P28] Matteo Tomasi, Shrinivas Pundlik, and Gang Luo. "FPGA–DSP co-processing for feature tracking in smart video sensors". English. In: *Journal of Real-Time Image Processing* (2014), pp. 1–17. issn: 1861-8200.

[P29] Jianhui Wang et al. "An Embedded System-on-Chip Architecture for Real-time Visual Detection and Matching". In: *Circuits and Systems for Video Technology, IEEE Transactions on* 24.3 (2014), pp. 525–538. issn: 1051-8215.

[P30] Zhaoyi Wei, Dah jye Lee, and Brent E. Nelson. *FPGA-based Real-time Optical Flow Algorithm Design and Implementation.*

[P31] Han Xiao et al. "Real-time scene recognition on embedded system with SIFT keypoints and a new descriptor". In: *Mechatronics and Automation (ICMA), 2013 IEEE International Conference on.* 2013, pp. 1317–1324.

[P32] Sheng Zhong et al. "A real-time embedded architecture for SIFT". In: *Journal of Systems Architecture* 59.1 (2013), pp. 16 –29. issn: 1383-7621.

Other references

[1] Herbert Bay et al. "Speeded-Up Robust Features (SURF)". In: *COMPUTER VISION AND IMAGE UNDERSTANDING* 110.3 (2008), 346–359. issn: 1077-3142.

[2] ZM. Calonder et al. "BRIEF: Computing a Local Binary Descriptor Very Fast". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 34.7 (2012), pp. 1281–1298. issn: 0162-8828.

[3] DG Lowe. "Distinctive image features from scale-invariant keypoints". In: *INTERNATIONAL JOURNAL OF COMPUTER VISION* 60.2 (2004), pp. 91–110. issn: 0920-5691.