ALMARVI

"Algorithms, Design Methods, and Many-Core Execution Platform for Low-Power Massive Data-Rate Video and Image Processing"

Project co-funded by the ARTEMIS Joint Undertaking under the
ASP 5: Computing Platforms for Embedded Systems
ARTEMIS JU Grant Agreement n. 621439

Parallel and Power-Aware Image Segmentation Algorithms (Architecture and Design)

D2.4

Due date of deliverable: September 30, 2015

Start date of project: 1-4-2014		Duration: 36 months
Organisation name of lead contractor for this deliverable:		UEF
Author(s):	UEF (Maarit Tamminen)	
Validated by:	UTIA (Barbara Zitova)	
Version number:	V1.3	
Submission Date:	<23-10-2015>	
Doc reference:	-	
Work Pack./ Task:	WP2 Task 2.3	
Description: *(max 5 lines)*	This document provides architecture and design details including architecture and design figures for various image segmentation algorithms; it will also provide architecture and design details on scalability and relaxation of segmentation methods, power-aware scalability, etc.	

Nature:	R		
Dissemination Level:	PU	Public	X
	PP	Restricted to other programme participants (including the JU)	
	RE	Restricted to a group specified by the consortium (including the JU)	
	CO	Confidential, only for members of the consortium (including the JU)	

DOCUMENT HISTORY

Release	Date	Reason of change	Status	Distribution
V1.3	23/10/2015	UEF (Maarit Tamminen)	Final	Artemis

Table of Contents

Glossary

Abbreviation / acronym	Description
ARM	Advanced RISC Machines
CNN	Convolutional Neural Network
CPU	Central Processing Unit
cuDNN	CUDA Deep Neural Network
FCN	Fully Convolutional Network
GDT	Geodesic Distance Transformations
GPU	Graphic Processing Unit
FPGA	Field Programmable Gate Array
HOG	Histogram of Oriented Gradients
HW	Hard Ware
LRN	Local Response Normalization
MANR	Motion Adaptive Noise Reduction
MM	Mathematical Morphology
Mpx	Megapixel
msec	Millisecond
MTF	Motion Transfer Function
OpenCL	Open Computing Language
OpenCV	Open Computer Vision
OpenMP	Open Multi-Processing
OS	Operating System
ReLU	Rectified Linear Unit
RGB	Red, Green, Blue
rVEX	University of Delft's parallel hardware platform
SLIC	Simple Linear Iterative Clustering
SVM	Support Vector Machine
SW	Soft Ware
TTA	Nokia's Transport Triggered Architecture

1. Executive Summary

This document represents deliverable D2.4, which is part of Task 2.3 in WP2. Therefore, this document describes research and development of the methods and advanced algorithms for image segmentation, feature extraction, and clustering. The key focus is on developing algorithms for massive data-rate image/video processing and analysis, tailored towards the specific ALMARVI requirements and system specifications to support for performance and power scalability. The high-performance and low-power algorithms are amenable to parallelization using ALMARVI multi-/many-core execution platform instances (including CPU + GPU/FPGA + TTA + rVEX). Based on this, parallelized low-power algorithms and implementations for different image/video processing functions will be consolidated in this deliverable.

Furthermore, the work performed in this document is aimed at presenting challenges in the low quality images/videos in surveillance and healthcare applications. Therefore, solutions to the enhancement and evaluation of the image/video quality are proposed. Different advanced image segmentation, feature extraction, and clustering methods will be utilized.

The following four objectives will be taken into account in the ALMARVI project. However, Task 2.3 will mainly focus on objectives 1, 2, and 3. The algorithms/methods in Task 2.3, which achieve all or some of these objectives are listed below:

1. *Enabling Massive Data Rate Processing:* Background modelling and subtraction algorithm based methods (see Section 3.1), Mean-shift filter based clustering (see Section 3.2), Simple linear iterative clustering method (see Section 3.3), Level-sets, snakes, graph-cuts, and distance transform based methods (see Section 3.4), Mathematical morphology based methods (see Section 3.5), Hardware-limited segmentation methods (see Section 3.6), Convolutional neural networks (see Section 3.7), and Machine learning based scene analysis using modern methods (see Section 3.8). Thus, all algorithms/methods in Task 2.3 achieve this objective.
2. *Achieving Low Power Consumption:* Background modelling and subtraction algorithm based methods (see Section 3.1), Mean-shift filter based clustering (see Section 3.2), Simple linear iterative clustering method (see Section 3.3), Level-sets, snakes, graph-cuts, and distance transform based methods (see Section 3.4), Mathematical morphology based methods (see Section 3.5), Hardware-limited segmentation methods (see Section 3.6), Convolutional neural networks (see Section 3.7), and Machine learning based scene analysis using modern methods (see Section 3.8). Thus, all algorithms/methods in Task 2.3 achieve this objective.
3. *Composability, Flexibility, and Cross-Domain Applicability:* Level-sets, snakes, graph-cuts, and distance transform based methods (see Section 3.4), Mathematical morphology based methods (see Section 3.5), Hardware-limited segmentation methods (see Section 3.6), Convolutional neural networks (see Section 3.7), and Machine learning based scene analysis using modern methods (see Section 3.8). Thus, almost all algorithms/methods in Task 2.3 achieve this objective.
4. *Robustness to Variability:* Hardware-limited segmentation methods (see Section 3.6), Convolutional neural networks (see Section 3.7), and Machine learning based scene analysis using modern methods (see Section 3.8). Thus, only three algorithms/methods in Task 2.3 achieve this objective, but it will be more comprehensively achieved in the next steps of the research and development work.

The partners involved in this deliverable were UEF, UTIA, ASEL, BUT, and HURJA. The results of this task will be employed in healthcare and security/surveillance demonstrators (Tasks 5.1 and 5.2) in which all four ALMARVI objectives will be achieved more comprehensively.

The deliverable is organized as follows. Section 2 provides introduction to the parallelizable and power-aware image/video segmentation, feature extraction, and clustering algorithms. In Section 3, advanced algorithms for image/video segmentation, feature extraction, and clustering are explored in eight Sections.

2. Introduction

One of the ALMARVI project's main goals is to develop adaptive, scalable, and parallelized algorithms for massive data-rate image/video processing and analysis to enable high-end services in key industrial domains, such as healthcare, security/surveillance, and mobile devices. This development enables to combine the power and performance challenges at all system layers, i.e. hardware, system software, and algorithms, while adapting each system layer to react to the run-time changing scenarios of available energy budgets, resources, user-defined constraints, etc.

The key focus is on developing algorithms for massive data-rate image/video processing and analysis, tailored towards the specific ALMARVI requirements and system specifications to support for performance and power scalability. To increase the potential of parallelization, the architectural features of the heterogeneous acceleration fabrics will be exploited in designing these algorithms using ALMARVI multi-/many-core execution platform instances (including CPU + GPU/FPGA + TTA + rVEX). Based on this, parallelized low-power algorithms and implementations for different image/video processing functions will be consolidated in this deliverable. Furthermore, scalability will be considered as an important design consideration in these image/video segmentation, feature extraction, and clustering algorithms.

Currently, several methods have been developed for image/video segmentation. Therefore, it is necessary to be able to evaluate the performance of image segmentation algorithms objectively. *Image segmentation* is the process of partitioning a digital image/video into multiple segments (set of pixels, superpixels). The goal of segmentation is to simplify and/or change the representation of an image/video into something that is more meaningful and easier to analyze.

Correspondingly, the goal of *feature extraction* is to extract useful information from an input image/video. The image/video is transformed into a reduced set of features (also named a "features vector"). The extracted features are expected to contain the relevant information from the input data, so that the desired task can be performed by using this reduced representation instead of the complete initial data.

The goal of *clustering* is to group a set of objects from an image/video in such a way that objects in the same group (called a "cluster") are more similar (in some sense or another) to each other than to those in other groups (clusters). It is a main task of exploratory data mining and a common technique for statistical data analysis, used in many fields, such as image/video analysis, machine learning, pattern recognition, information retrieval, and bioinformatics.

In this respect, the research and development of algorithms for image/video segmentation, feature extraction, and clustering will bring significant additional value to the goals of ALMARVI project. The following partners contributed to this deliverable: UEF, UTIA, ASEL, BUT, and HURJA. The results of this task will be employed in healthcare and security/surveillance demonstrators (Tasks 5.1 and 5.2) in which all four ALMARVI objectives will be achieved more comprehensively than in Task 2.3.

The rest of the deliverable is organized as follows. Section 3 explains advanced algorithms for image/video segmentation, feature extraction, and clustering as well as how each of these algorithms/methods will achieve all or some of the four objectives of the ALMARVI project. Section 4 concludes the document, and references related to the research and development of this deliverable are presented in Section 5.

3. Image/Video Segmentation, Feature Extraction, and Clustering Algorithms

In the following Chapters high-performance and low-power algorithms and advanced image segmentation, feature extraction, and clustering methods are proposed. The key objective is to utilize different methods for analysis and processing of massive image/video data through efficient algorithms and their parallel implementations.

3.1 Background Modelling and Subtraction Algorithm based Methods

In any computer vision surveillance system the most fundamental step is the segmentation of the background and foreground in real-time. The most typical method is background subtraction. [1] Background subtraction calculates the input image while subtracting each new frame from the input image and outputting the results with given threshold. The resulting output image will contains highlights of moving objects. One of these background subtractions algorithms uses Gaussian mixtures to model the background segmentation.

The OpenCV library, which implements the Gaussian mixture-based background/foreground segmentation algorithm, was chosen to be utilized in this research work. [1] The algorithm was chosen for robustness under changing lighting conditions and for the algorithms ability to detect shadows for moving objects. The shadow detection is used to reduce noise and to get clear output image, where the shadows are removed. The background pixels were modeled by a mixture of K Gaussian distributions where K value can range from 3 to 5. It learns the background over time from the video feed using weight from the color and time proportion of the pixel, meaning how long the pixel has stayed on the same color range – allowing some variation for shadows and general fluctuation of the image quality. Pixels of same colors that stay longer in the same area are most likely the pixels that belong to the background.

As shown in Figure 1, the learning rate for the model was set to 0.1 which means it'll use 90 % of the old frames before replacing the pixels in the model. The values range from -1 to 1, where -1 is automatic value chosen by the algorithm, 0 doesn't update the model at all and 1 means it would replace the model every new frame. A faster learning rate detects variations in the lighting better but the overall accuracy of the model is compromised. A slow rate keeps the model more consistent but a sudden change in lighting forces the algorithm to relearn the whole model again, causing a spike in the performance and makes it possible to lose some frames. [2]

The images in Figure 2 show, how the algorithm takes a full color input image (a) and outputs a black and white image where the foreground layer is marked with white pixels (b). If the shadows were kept in the image (b), they'd appear in the image as gray pixels.

a) Sudden lighting with learning rate of 0.1

b) Sudden lighting change with default learning rate

c) Lighting change after 80 frames with default learning rate. The light is still part of the foreground.

d) Lighting change after 80 frames with learning rate of 0.1. Light is now part of the background

Figure 1. Images with varying learning rate.

a) *b)*

Figure 2: a) Original image from video, b) foreground layer obtained from background subtraction.

3.2 Mean-Shift Filter Based Clustering

Clustering is a common method for image segmentation. Mean-shift is a clustering approach. It is a non-parametric iterative mode-seeking algorithm that is useful for detecting the maxima of a density function using discrete samples. The algorithm was originally introduced in Fukunaga and Hostetler [3]. Mean shift considers feature space as an empirical probability density function and the input data point as discrete samples from that density function. For example, in the context of image/video processing, feature space consists of pixel values (one for grayscale images and three for colour images) and the pixel coordinates (two), resulting in three (for grayscale images) or five (for colour images) dimensional feature vectors. Dense regions (or clusters) in the feature space, then correspond to the modes (or local maxima) of the probability density function. Hence a mode seeking algorithm, such as mean shift, can also be used for clustering, and, with proper post-processing, segmentation.

Each input data point is associated with the nearby peak of the density function. To achieve this, for each data point a local window is defined and the local mean within this window is calculated. Then the center of the window is shifted to the computed mean and these steps are repeated until convergence:

1. Set a local window around each data point.
2. Compute the mean of data within this local window (mean of all features, including pixel coordinates).
3. Shift the window to the computed mean and repeat until convergence.

Before presenting the formulation of the algorithm, we need to define what is meant by a *kernel.* A kernel is function that satisfies the following conditions:

(1) $\int_{R^d} \phi(x) = 1$

(2) $\phi(x) \geq 0$

A well-known example is the Gaussian:

(3) $\phi(x) = e^{\frac{x^2}{2\sigma^2}}.$

As explained above, the stationary points obtained by mean shift iterations represent the modes of the density function. All points associated with the same stationary point belong to the same cluster. Let us define:

$$g(x) = -K'(x), \quad (4)$$

where $K(x)$ is a kernel function, and mean shift $m(x)$ as:

$$m(x) = \frac{\sum_{i=1}^{n} g\left(\frac{x-x_i}{h}\right) x_i}{\sum_{i=1}^{n} g\left(\frac{x-x_i}{h}\right)} - x. \quad (5)$$

Given these definitions mean shift algorithm can be stated as follows. For each pixel:

1. Compute mean shift vector $m(x)$
2. Move the density estimation window by $m(x)$
3. Repeat till convergence

A detailed derivation of the algorithm and its relation to density function estimation can be found in [4].

For some application scenarios the output of the mean shift filter is a close enough approximation to segmentation. However, if the output is to be processed by upcoming blocks in a pipeline then explicit pixel labelling is required. There are multiple ways to achieve this labelling and current approach is to keep converged means for all pixels and assign pixel labels based on slightly post-processed pixel modes (such as applying simple k-means clustering on the converged modes and using the resulting cluster centres).

The mean shift algorithm explained above is embarrassingly parallel. Each pixel can be processed completely independently of the others and the thread to data mapping (excluding hardware platform specific resource optimization related issues) is trivial. The algorithm was parallelized using C/OpenCL. An initial implementation, in Figure 3 that is not thoroughly optimized was tested with an input size of 640x480, in Figure 4, and a hard coded iteration number of 4 (typically convergence is achieved in 2-3 iterations). Obtained timings were as follows in Table 1:

Table 1. Obtained timings (msec) of the images.

Graphic card	**Buffer memory implementation time** (msec)	**Texture memory implementation time** (msec)
AMD E6760	274.8	172.9
NVIDIA GTX 780	24.4	21.3

Figure 3. Input image.

Figure 4. Mean shift filtered image.

3.3 Simple Linear Iterative Clustering Method

In this work Simple Linear Iterative Clustering (SLIC) method was utilized. SLIC is a superpixel algorithm that can be leveraged for effective and efficient image segmentation. Superpixel algorithms group pixels into perceptually meaningful atomic regions that can be used to replace the rigid structure of the pixel grid is presented in Figure 6.

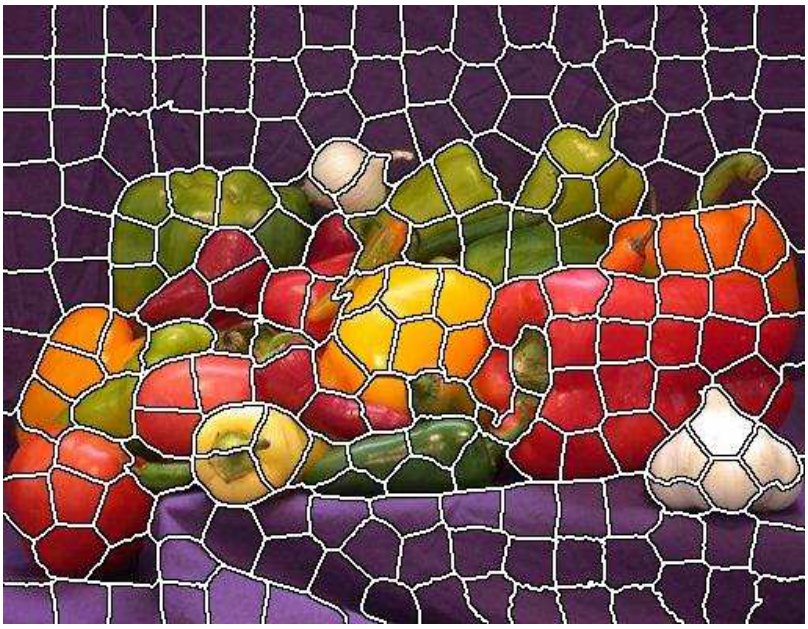
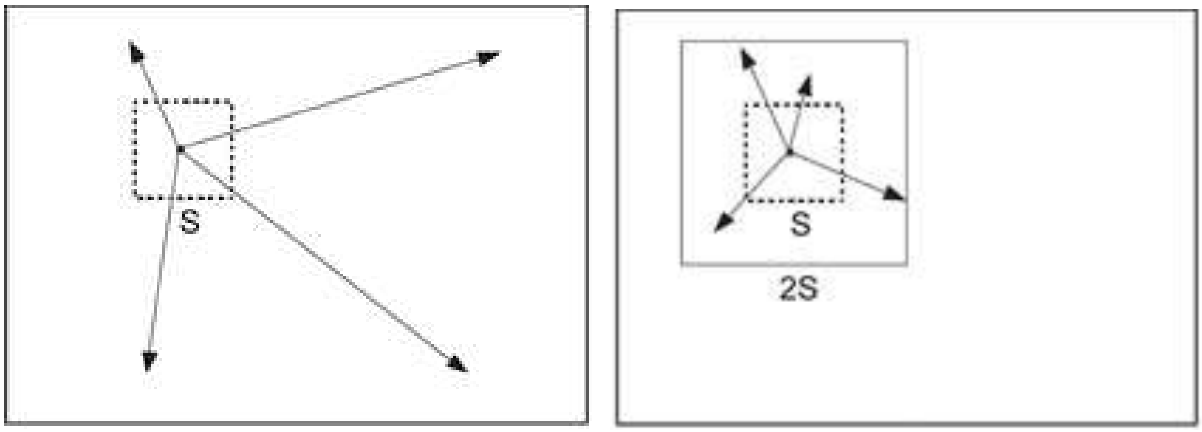


Figure 5. Superpixels constructed by SLIC.

SLIC is computationally less dense than other well performing segmentation algorithms. Basically, the algorithm can be summarized as spatially constrained k-means. The only parameter of the algorithm is k, the desired number of approximately equally sized superpixels.

For colour images, the first step is to convert to one of the uncorrelated colour spaces such as CIELAB. The clustering of the pixels begins with an initialization step where k initial cluster centres are sampled. To avoid putting a superpixel on an edge and to reduce the chance of seeding with a noisy pixel the centres are moved to seed locations corresponding to the lowest gradient position in a 3x3 neighbourhood. Next is the assignment step. Each pixel is associated with the nearest cluster centre whose search region overlaps its location. Due to this limited search region the number of distance calculations is significantly reduced compared to the conventional k-means clustering where each pixel must be compared with all cluster centres. Since the expected spatial extent of a superpixel is a region of approximate size SxS, the search for similar pixels is done in a region 2Sx2S around the superpixel centre as shown in the Figure 5 below. Once each pixel has been associated to the nearest cluster centre, an update step adjusts the cluster centres to be the mean of all the pixels belonging to the cluster. The L2 norm is used to compute a residual error E between the new cluster centre locations and previous cluster centre locations. The assignment and update steps can be repeated iteratively until the error converges. Finally, a post processing step enforces connectivity by reassigning disjoint pixels to nearby superpixels.



a) b)

Figure 6. Search regions for conventional k-means and SLIC. a) Standard k-means searches the entire image and b) SLIC searches a limited region

Based on this, the SLIC algorithm can be stated as follows:

Initialization

Initialize cluster centres $C_k = [l_k, a_k, b_k, x_k, y_k]$ by sampling pixels at regular grid steps S.

Move cluster centres to the lowest gradient position in a 3x3 neighbourhood.

Set label l(i)=-1 for each pixel i.

Set distance d(i)=∞ for each pixel i.

Repeat

//assignment

for each cluster centre C_k **do**

for each pixel i in a 2Sx2S region around C_k **do**

Compute the distance D between C_k and i.

if D < d(i) **then**

set d(i) = D

set l(i) = k

end if

end for

end for

//update

Compute new cluster centres.

Compute residual error E.

until E ≤ threshold

Enforce Connectivity.

A detailed explanation of the algorithm as well as computational complexity comparisons to other segmentation algorithms can be found in [5]. In its original form, SLIC algorithm cannot be parallelized efficiently due to the atomic comparison that is in the innermost loop. Some possible improvement is achieved by traversing the pixels in the local neighbourhood in raster scan order in terms of segmentation purposes. In this work it was chosen to rearrange the for-loops and modify the search order to obtain a parallelizable version which is presented below:

Initialization

Initialize cluster centres $C_k = [l_k, a_k, b_k, x_k, y_k]$ by sampling pixels at regular grid steps S.

Move cluster centres to the lowest gradient position in a 3x3 neighbourhood.

Assign the pixel to the nearest cluster centre based on initial grid interval S.

Repeat

for each pixel i **do**

Locally search the nearby N cluster centres and label this pixel with the nearest cluster's index.

end for

Update each cluster centre based on pixel assignment and compute residual error E.

until E ≤ threshold

Enforce Connectivity.

The number of neighbouring cluster centres to be searched, N, is a control knob that can be used to trade-off computational complexity with segmentation quality. Current experiments concentrate on N=4 and N=9. The modified SLIC algorithm can be parallelized but the thread to data mapping is nontrivial.

The forward and backward colour space transformations were embarrassingly parallel with a trivial thread to pixel mapping. It was the same for cluster centre initialization. Since the step size S, which determined the approximate super-pixel size, had a direct effect on the thread mapping and since typical GPU optimizations were dependent on the block size, we planned to have multiple implementations for varying granularities of the k-means stage. They were two implementations, one based on a 16x16 blocks and one based on 32x32 blocks. Number of these optimized kernels may need to be increased as we further optimized our implementation. To avoid atomic instructions, cluster centre updates were done with a single thread. After k-means converges, the labelled image was transferred back to the host CPU where a *sequential* post-processing algorithm was applied to enforce connectivity. So far no OpenMP work has been conducted on the CPU side.

C/C++ model can be implemented in OpenCL for accelerating on GPUs and Field Programmable Gate Arrays (FPGAs). This initial implementation ran at 24 msec/frame processing 640x480 input images into 384 super-pixels, and provided up to 6x acceleration with a NVIDIA GTX 780 GPU (compared to Intel Core-i7 4770).

3.4 Level-Sets, Snakes, Graph-Cuts and Distance Transform based Methods

This research work concentrated on image segmentation and feature extraction methods with special focus on level-sets, snakes, and distance transforms -based methods [6, 7, 8]. The experiments were performed with two types of image processing algorithms: energy minimization and Mathematical Morphology (MM) based segmentation algorithms. Experimenting their scalability in a parallel computation setting was performed as well.

In energy minimization, based image segmentation algorithms, there is an attempt to minimize an objective function, which determines parts in a partition. A popular and widely used objective function for energy minimization is the Mumford-Shah functional [*7*]. Popular paradigms for this type of functional minimization are the Level-Set method, Active Contours or Snakes and Graph-cuts. All three paradigms have in common that they are computationally intensive and difficult to parallelize. For parallelization purposes, experiments on parallelizing these paradigms is being conducted in a multilevel setting, which is derived from the multigrid method. Multilevel methods requires extensive optimization for achieving good performance. So the optimization of the multilevel method was the main focus in this work, which in turn augmented the performance of the three aforementioned paradigms. [9, 10]

In MM, based on image segmentation algorithms as well, geodesic distance transformations (GDTs) can be utilized to compute the shortest path on a given surface. The computation can either be done using a raster scan approach (popularized by Rosenfeld and Pfaltz) or wave propagation (as pointed out by Piper and Granum). For the future work, for the purpose of automation, we will consider the raster scan approach, since choosing an initial propagation seed will not be required. GDTs in a raster scanning setting are cheap to compute (O(n) in computation and also in space complexity), and also parallelization is easier. However GDTs are more sensitive to noise and may yield poor segmentation results if feature objects are in close proximity.

Detailed performance analysis will later be performed using scalable ordinary and low-power implementations of these algorithms for the ALMARVI execution platform instances (including CPU + GPU/FPGA + TTA + rVEX). In order to achieve real time or near real time processing times, we will utilize parallelization using multi-core CPU, GPU, and TUT/Nokia's Transport Triggered Architecture (TTA) as well as University of Delft's rVEX parallel hardware platform.

3.5 Mathematical Morphology based Methods

Mathematical Morphology (MM) was initiated in late sixties by Georges Matheron and Jean Serra for the analysis of spatial structures, aiming at the analysis of object shape and form. [11, 12] MM can simplify image data by eliminating irrelevancies while still preserving shape characteristics such as edges, holes, corners, fillets, wedges and cracks with various sized and shaped structuring elements. [13] The algorithms have stayed relatively unchanged to this day.

In this research work the recursive and sequential combination of dilation and erosion resulted in the elimination of specific image details whose size were smaller than what was defined in the structuring element. [13] Using the basic opening and closing operations of MM with different shape and sized structuring elements, several different resulting images were obtained and they can be seen in the following Figures 7, 8, 9, and 10.

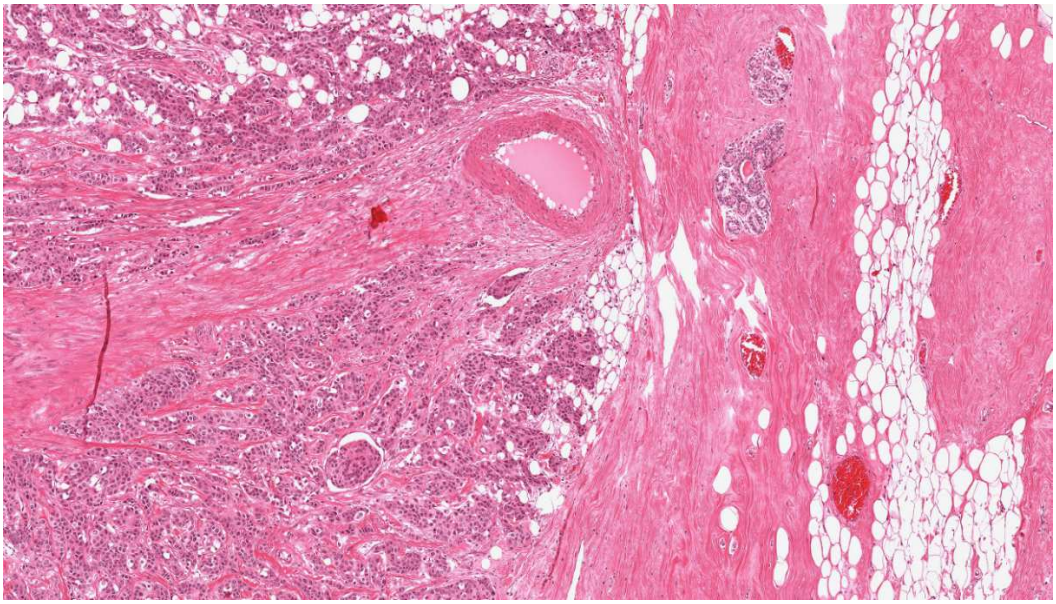


Figure 7. Original RGB image.

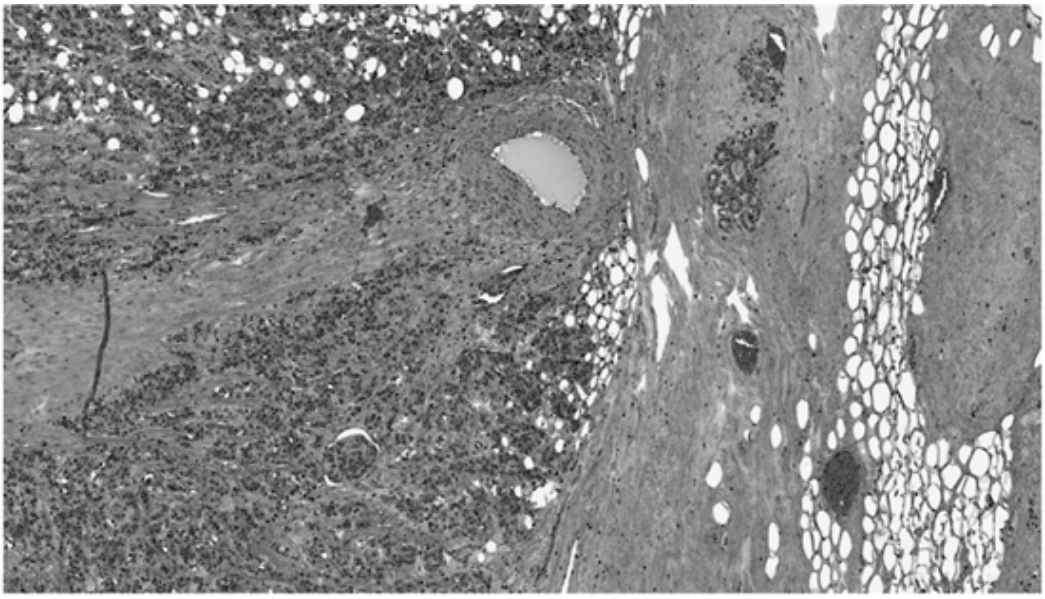


Figure 8. Opening with square 3x3.

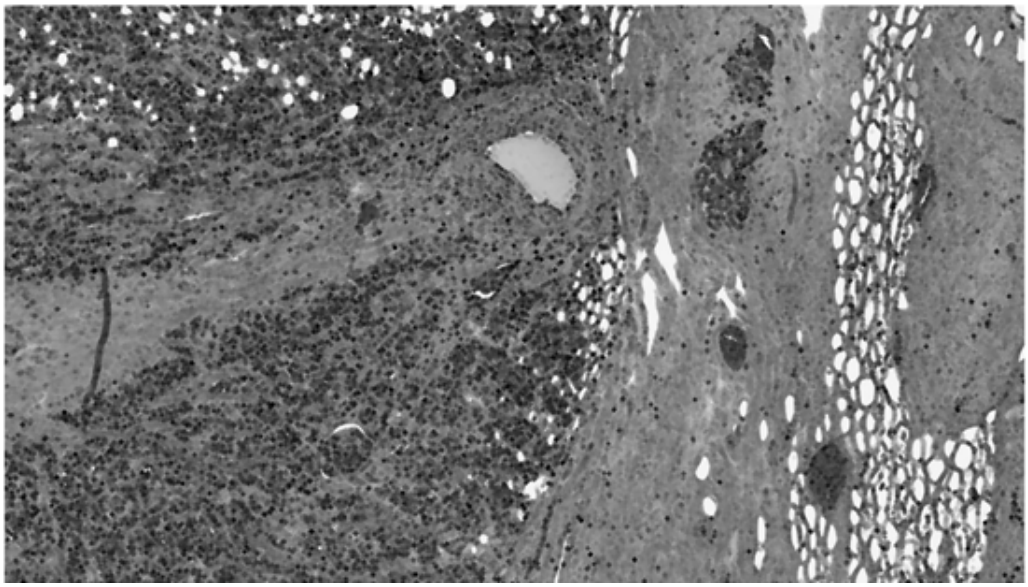


Figure 9. Opening with a diamond.

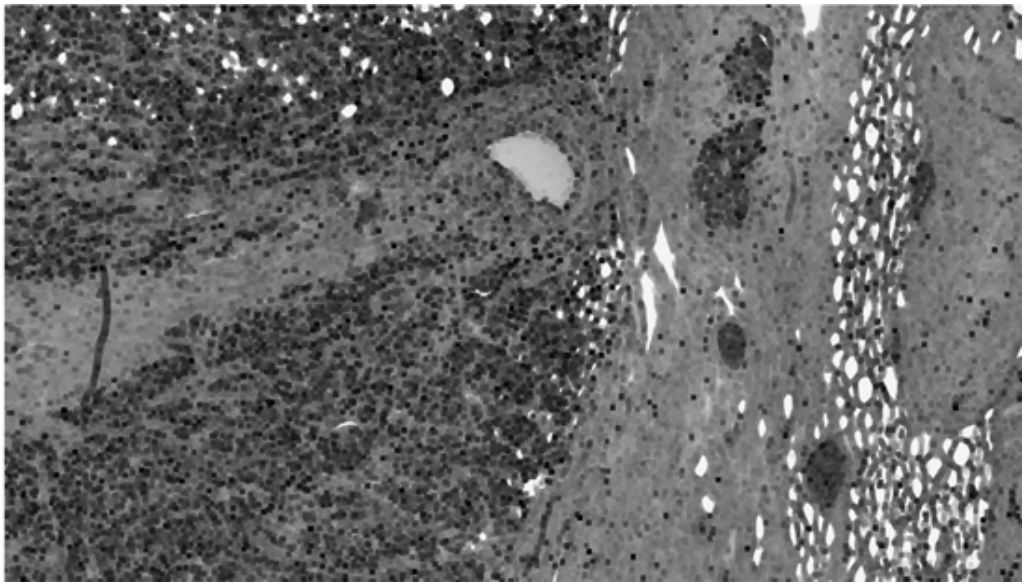


Figure 10. Opening with a ball.

Top-Hat Transform is a common technique for contrast enhancement and small target detection. Top-Hat computes a morphological opening of an image and subtracts the result from the original image (Figure 11). A resulting transform image is seen in Figure 12. The image shows a clear contrast difference between the nuclear area and other tissue sections. Bottom-Hat Transform is defined as the difference between the original image and its closing which is the collection of background parts of and image that fit a particular structuring element, resulting Bottom-Hat image is seen in Figure 13. Top-Hat and Bottom-Hat can be used together recursively to achieve a better result.

White Top-Hat Transform (WTH) is defined as the difference between the function and its morphological opening and it extracts bright regions of an image according to the used structuring element (see Equation 6): [14]

(6) $WTH(x,y) = f(x,y) - f \circ B(x,y)$

Black Top-Hat Transform (BTH) uses closing to enhance the dark regions (see Equation 7): [14]

(7) $BTH(x,y) = f \bullet B(x,y) - f(x,y)$

The nuclei in an image appears small and dark spots inside a cell. Therefore, the use of Black Top-Hat Transform is justified. The performance and time requirements of Top-Hat Transform based methods are investigated i.e. in the article of Yardimci et al. [15]

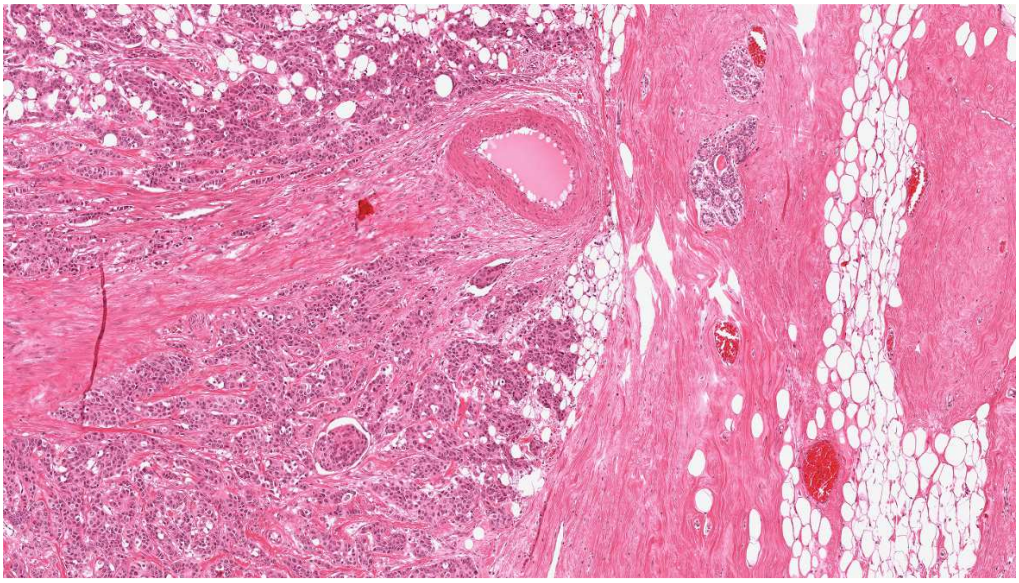


Figure 11. Original image

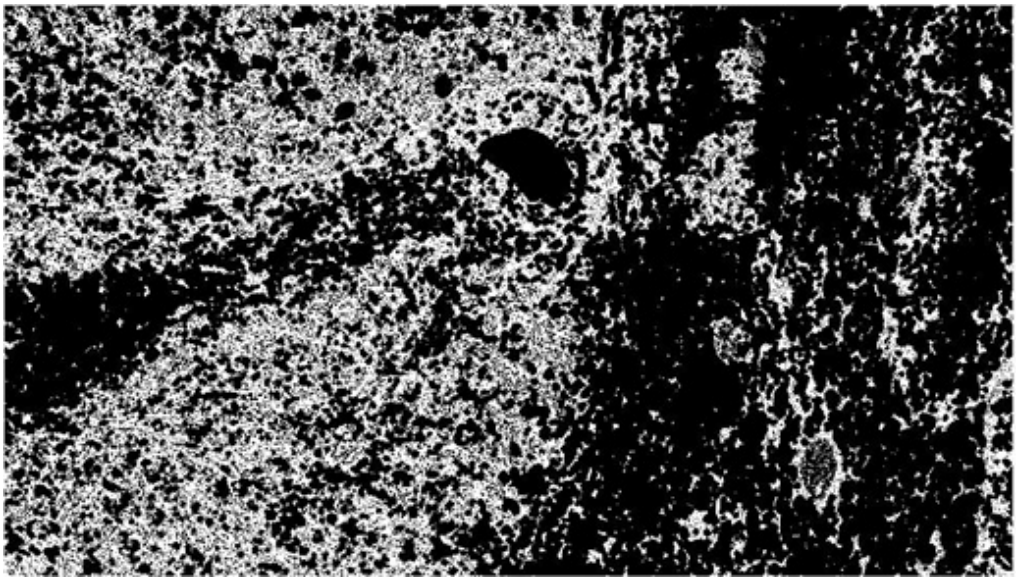


Figure 12. Top-Hat Transform on binary image with a disk-shaped structuring element with radius of 4.

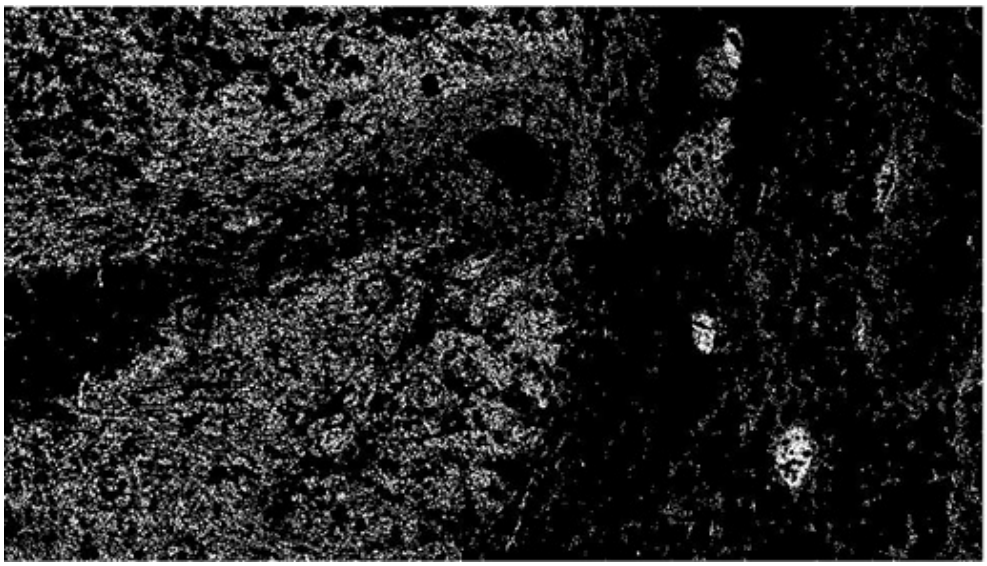


Figure 13. Top-Hat and Bottom-Hat Transform on binary image with a disk-shaped structuring element with radius of 4.

In another work, multiple morphology algorithms were chosen to be utilized to reduce the noise from the video feed and to merge contours that are close to each other, most likely belonging to the same object but divided by things like shadows that weren't removed in the background subtraction phase. The image (left) in Figure 14 is obtained from the foreground layer from the background subtraction. It's then cleaned (right) from noise by first running the erosion algorithm and then filling back the holes with dilate, both algorithms are from OpenCV.

The contours obtained from the cleaned foreground layer are merged using an empty mask which the contours are drawn on as seen on image (right side). The morphology open method from OpenCV is used to merge the contours. The purpose is to acquire a region of interest that has as much of the foreground as possible. See Figure 25.

Figure 14: Left: Image after applying erosion again, then dilation. Right: Image after morph open.

3.6 Hardware-Limited Segmentation Methods

In this work hardware implementation of an edge detection algorithm implementation for monochrome video sequences with resolution 1920x1080 8-bit per pixel was researched. The edge detection algorithm was based on a sequence of two image processing operations. A motion adaptive noise reduction algorithm (with selectable strength) MANR was followed by a Sobel filter, SOBEL performing the edge detection on the filtered video pixels.

The MANR noise reduction algorithm was implemented with a recursive temporal filter that uses a programmable motion transfer function (MTF) to control both the shape of the noise reduction curve, as well as the "strength" of the noise reduction.

The difference between the current and previous frame was calculated as a first step. The difference image is filtered by a 2D averaging FIR filter, using a 3x3 kernel. The absolute value of the low-pass filter output, the motion value, was affected to a greater extent by objects moving in or out of the kernel (motion) than random noise. This motion value was used as an index to the MTF look-up table. The Motion Transfer Function reflects the probability density function of the noise superimposed on the stream. Assuming Gaussian noise, the S-curve shape of the MTF intends to minimize the error that motion is falsely characterized as noise.

In the next phase of the MANR algorithm, the value corresponding to the calculated motion value from the MTF was used as a multiplier to scale the pixel-by-pixel difference value. The resulting value was summed with the current frame pixel value, resulting in an output pixel that contains a percentage of the previous frame and the current frame. This same output was then written to memory and becomes the previous frame for the next cycle, thus forming a recursive filter. Consequently, the entire input frame was filtered in a recursive fashion.

The implemented MANR and SOBEL algorithms were processing only the 8 bit per pixel intensity channel and work with the 1920x1080 frames. The MANR and SOBEL algorithm were implemented in C and executed on embedded ARM Cortex A9 processor of the Xilinx ZYNQ 7000 all programmable platform. ARM processor worked with 666 MHz clock. To address the HW implementation, the algorithm was implemented partially in SW (on ARM Cortex A9) and partially of the ZYNQ 7000 platform on HW with the clock 100 MHz.

Three specific cases of partial HW implementation of the edge detection algorithm were compared:

- SOBEL filter in HW, remaining part of algorithm on ARM.
- MANR filter in HW, remaining part of algorithm on ARM.
- MANR and SOBEL in HW and there is direct local data pipe from MANR to SOBEL in HW.

The computing time and the relative acceleration in comparison to the ARM implementation alternatives are summarized in Table 2.

Table 2. Performance comparison.

Performance comparison:	**Frame size**	**Time to process one frame**	**Acceleration**
SOBEL filter in HW:	Frame 1920x1080	867 ms	1,33x
MANR filter in HW:	Frame 1920x1080	771 ms	1,54x
MANR and SOBEL in HW:	Frame 1920x1080	393 ms	3,04x
ARM SW no HW	Frame 1920x1080	1195 ms	1,00x

The input HD video sequence is shown in Figure 15.

Figure 15. Input video sequence of 1920x1080 frames processed in HW with the motion adaptive noise reduction and the Sobel filter.

Figure 16 is presenting the output HD video sequence of 1920x1080 frames processed in HW with the motion adaptive noise reduction and the Sobel filter. The horizontal micro-line is dividing the Top part of Figure 16 with the MANR filter ON and the bottom with the MANR filter OFF. The left part of Figure 16 displays the frame without SOBEL and the right part with SOBEL filter.

Summary of the conditions of the test:

- Board Xilinx ZC702, ARM A9 (release optimized C code, no use of NEON unit, 666 MHz system clock, DDR3 1GB).
- The ARM C program was running in the “bare metal” configuration without an OS.
- The programmable logic HW implementation was working with 100 MHz clock.
- The sequence of 10 frames monochrome 1920x1080 was read from SD card by ARM processor.
- The sequence of 10 frames monochrome 1920x1080 was stored to SD card by ARM processor.
- The time needed for SD card file RD and file WR and time needed for the file format conversion to/from the monochrome (8 bit per pixel) is not included in the measured execution time and related acceleration figures.
- The HW implementation has been compiled by the Win7 64bit version of the SDSoC Environment version 2015.2. This environment has been developed by Xilinx and targets the ZYNQ 7000 family of all programmable devices.
- The MANR and SOBEL HW implementation presented in this section is building on the demo material developed by Xilinx and provided to the users in the SDSoC Environment as part of the version 2015.2 release.

The implementation with chained MANR and SOBEL blocks needed some resources for algorithm and some resources for the data communication described as follows. The HW implementation has these data and clock cycle parameters.

- Each video frame has 1920x1080 = 2.073.600 pixels.
- SOBEL in HW needs 2076610 = 2.073.600 + 3.010 clock to process one frame.
  MANR in HW needs 18.662.453 = (9 x 2.073.600) + 53 clock to process one frame.

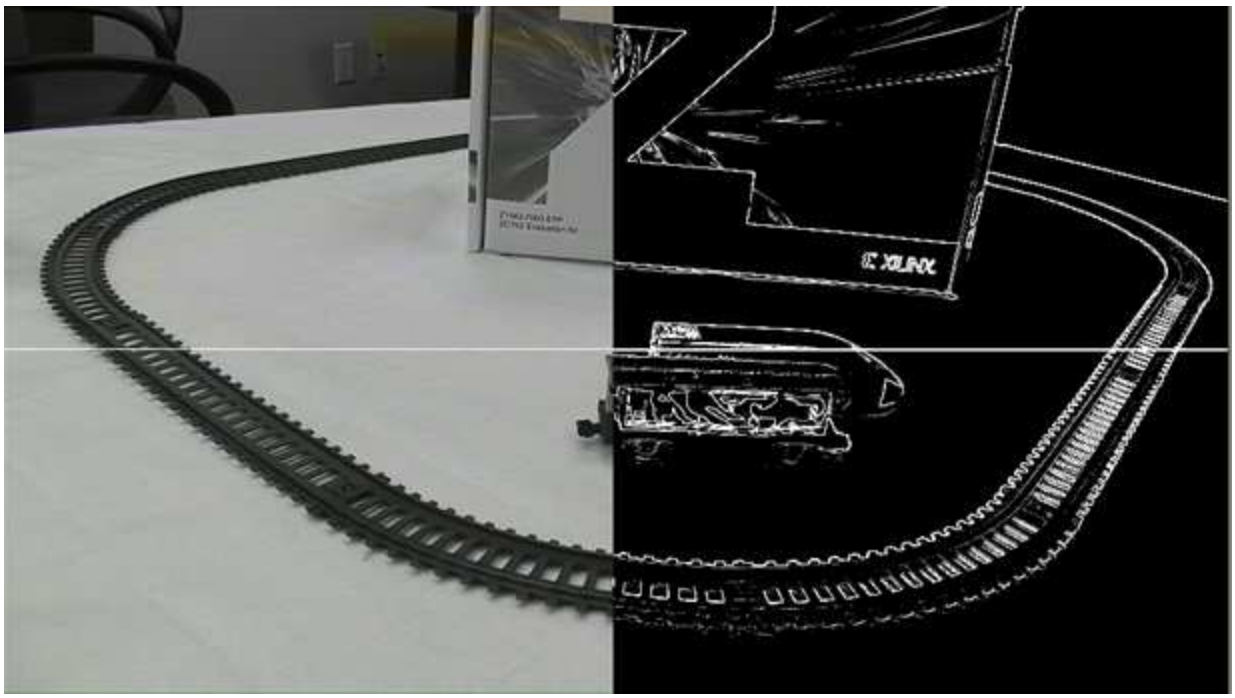


Figure 16. Output video sequence of 1920x1080 frames processed in HW with the motion adaptive noise reduction and the Sobel filter. Top part with MANR, bottom without MANR.

Table 3 is indicating the HW resources used by different parts of the chained implementation.

Table 3. HW resources used by implementation with chained MANR and SOBEL blocks.

MANR in HW + SOBEL in HW	Only MANR + SOBEL	MANR + SOBEL + data movers	Resources of xc7z020
BRAM blocks	8 + 3 = 11	8+3+16=27	280
DSP blocs	3 + 0 = 3	3+0+0=3	220
FF (flip-flops)	759 + 440 = 1.199	759 + 440 + 16.993 = 18.192	106.400
LUT (look-up-tables)	1.265 + 476 = 1.741	1.265 + 476 + 11.244 = 12.685	53.200

Table 2 indicated that:

- Relatively large proportion of HW resources is taken by the data movers, DMA and the AXI4 infrastructure.
- The MANR and SOBEL HW blocks take relatively small proportion of resources.
- The SOBEL HW implementation can compute single pixel in single clock cycle plus fixed overhead of 3010 clock cycles per frame.
- MANR HW implementation can compute single pixel in 9 clock cycles plus fixed overhead of 53 clock cycles per frame.

The PL resources and the data-parallelism due to the processing of the frame line-by-line should allow to realize these improvements of the design:

- Replicate 9x the internal MANR kernel in HW.
- Execute in parallel all 9 MANR HW kernel instances.
- Synchronise locally results from 9 MANR HW kernel instances and send them to SOBEL HW.
- Increase the HW clock from 100 MHz to 142.5 MHz (HDMI 1920x1080p60 clock).

Implementation of these steps will lead to HW capable to process the HD 1920x1080p60 video sequence in HW with the rate 1 pixel in 1 clock cycle. These steps have not been implemented in time of writing of this report. We intend to modify the MANR in C to direct the SDSoC compiler and the related HLS compiler in this outlined direction.

In this work the example of the MANR and SOBEL algorithm was used to indicate, how it was intended to address in HW these four challenges of the ALMARVI project:

- *Enabling Massive Data Rate Processing:* Full HDMI data rate in HW algorithms, which need for the operation more than 1 video-frame.
- *Achieving Low Power Consumption:* Direct streaming reducing DMA channel overhead and power.
- *Composability, Flexibility, and Cross-Domain Applicability:* HW design from single reference C source of SOBEL as MANR filter. Portable to multiple different HW platforms with programmable logic section.
- *Robustness to Variability:* In design of HW data movers and connectivity of HW blocks. HW block remains identical, but it can be connected (chained) directly with an AXI-stream bus or in another context it can be connected to a data mover.

3.7 Convolutional Neural Networks

Neural network based approach for image segmentation was proposed in this work. Some neural-based approaches perform segmentation directly on the pixel data, obtained either from a convolution window or the information is provided to a neural classifier in the form of local features. In recent years, Convolutional Neural Networks (CNNs) have transformed the state-of-the-art in many computer vision tasks. [16; 17; 18; 19; 20]. The trend of successful applications of convolutional networks was initiated [21] on large-scale image classification for the ILSVRC1 2012 challenge. Although neural networks, and more relevantly convolutional networks, had been successfully applied to important vision problems before, their benefit was not as clear and they were not accepted by mainstream computer vision community.

As shown in Figure 17, the architecture of the image classification network [21] is a large feedforward multi-layer network which consist of convolutional layers, Rectified Linear Units (ReLU), Local Response Normalization (LRN) layers and max-pooling layers. The architecture is notable as it is archetypal for a many of today's state-of-the-art networks in diverse vision areas.

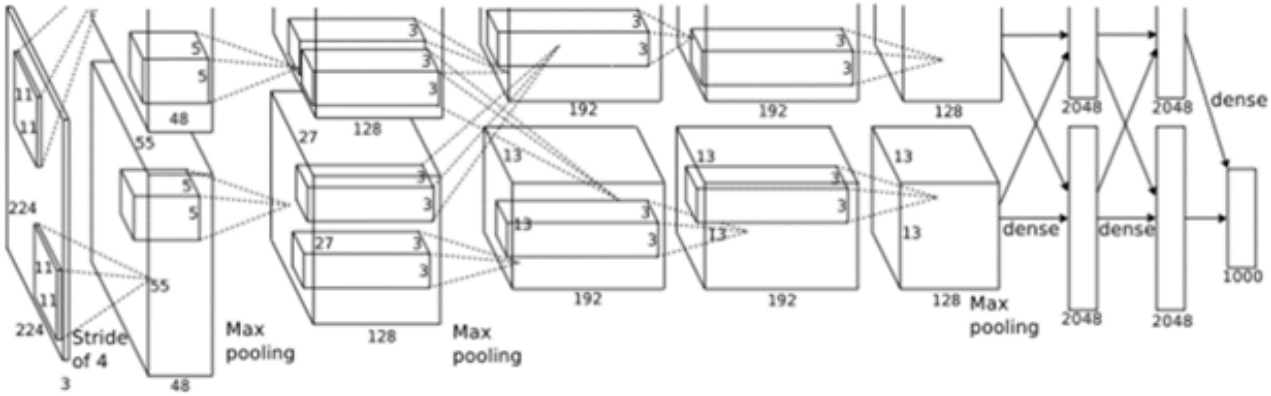


Figure 17. Architecture of image classification network [21]. The larger blocks represent the outputs and inputs of individual layers and smaller boxes represent convolutional kernels.

In *classical feedforward networks*, the layers containing trainable parameters are fully connected, that is all neurons from a previous layer are connected to all neurons of the next layer. In the type of networks considered in this text, the operation the neurons perform is a weighted sum of inputs. The values of the weights are determined during

[1] ImageNet Large Scale Visual Recognition Challenge 2012

training on a set of exemplar inputs and desired outputs. The weighted sums of neurons in a single layer can be expressed as a single matrix-vector multiplication.

In *convolutional networks*, neurons are connected only to a small number of neurons of the previous layer. These connections are local (2D neighbourhood for images, 3D neighbourhood for volumetric data). Such local connections naturally follow the statistical nature of real signals in which spatially close data points (e.g. pixels) are often strongly correlated and should be processed together. Moreover in a convolutional layer, the one neuron type (neuron with specific weights) is replicated across the whole extent of the input signal - effectively computing the same operation across the whole image. The activations of neurons of the same type are called channels. The idea of processing the whole input the same way makes sense in many tasks where positions of objects in an image can be arbitrary (e.g. image classification and object detection). Even if the images exhibit strong structure (e.g. facial recognition), the convolutional processing is adequate at the lower levels of a network as such layers mostly learn to efficiently represent image data independently of the target task (e.g. the early layers detect edges and generic local shapes).

Computational structure and requirements of other layers of convolutional networks is similar to the convolutional layer. The operations are local and uniform over whole input. ReLU nonlinearity is a point-wise operation, LRN normalizes some norm-like quantity of activations in a small neighbourhood, and max-pooling applies MAX operation in a small neighbourhood separately on individual channels. Only the linear convolutional layers contain trainable parameters - other layer are hard-wired. Dropout, which is used to regularize convolutional networks, is not a layer in the strict sense of the word, it randomly sets activations to 0 with predefined probability during training and does not affect the network in “production” mode.

A convolutional layer can be exactly expressed as linear convolution and existing highly-optimized libraries could be, in theory, used to compute this operation. However, the convolutions in convolutional networks tend to be spatially smaller compared to convolutional kernels used in image processing. The typical kernel sizes start from 3x3 pixels and rarely exceed 7x7 pixels. Moreover, convolutions in image processing typically process one or three channels (e.g. grayscale or Red, Green, Blue (RGB) channels) and the number of channels in neural networks easily exceeds 512. For this reason, specially optimized routines have been developed for both CPUs and GPUs (e.g. NVIDIA cuDNN2).

Large networks, such as those used in image classification, may contain hundreds of millions weights [20] and tens of layers [17]. Despite the large size of such networks, they are able to classify over hundred images per second on a high-end GPU. During training the speed is reduced roughly to one half as error gradients have to be backward propagated through the network. Typical speed during training is 100 images per second on a single GPU. In case of large networks, hundreds of millions images have to propagated through the network to achieve state-of-the-art classification performance - this translates to training times ranging from a week up to several months on a single GPU. In order to reduce the training time, several parallel systems [23; 24; 25] have been described using both GPU and CPU. Recently, even systems accelerating on FPGAs are emerging [26].

Existing research shows that large convolutional networks trained on one task perform well on other related tasks [27]. This allows to cut-down training time by reusing pre-trained networks and to apply large networks even on very small datasets. The common practice is to start from one of the publically available image classification networks trained on ImageNet database, replace one or two top layers with random weights and to fine-tune the resulting network for a new task. Such networks can be trained in several hours and they tend to provide state-of-the-art results on many vision tasks.

Image Restoration Using CNN

As shown in Figure 18, image restoration belongs to the preprosessing section in the image processing chain. [28] Therefore, image restoration is one of the tasks performed by the algorithm before the segmentation step.

[2] NVIDIA cuDNN https://developer.nvidia.com/cudnn

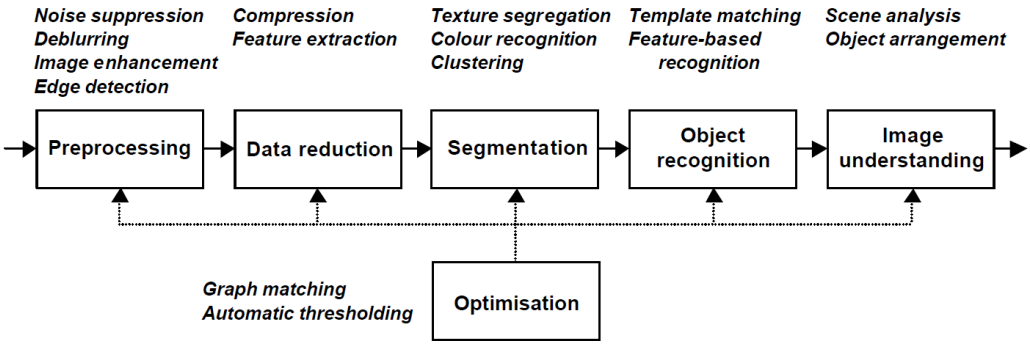
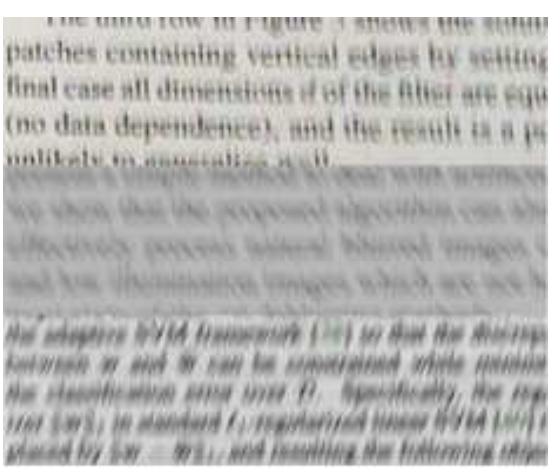


Figure 18. Image Processing Chain.

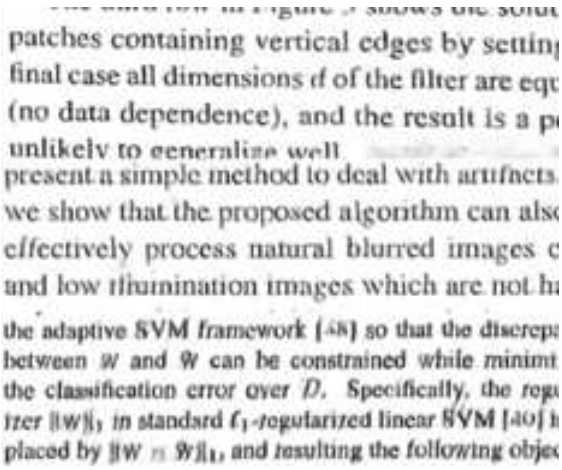
Neural networks have been traditionally applied to many low-level image processing tasks including noise removal [29] and edge detection [30]. Recently, convolutional networks achieved state-of-the-art results in several sub-tasks of image restoration including denoising [31], structured noise removal [31], non-blind deconvolution [33; 34], sub-tasks of blind deconvolution [36], and full end-to-end blind deconvolution[3]. CNNs has been applied to space-invariant non-blind deconvolution [34]. They initialize first two layers with a separable decomposition of an inverse filter and then they optimize the full network on artificial data. This network can handle complex blurs and saturation, but it has to be completely retrained for each blur kernel. In this work a small CNN (15,697 parameters) was utilized to remove additive white Gaussian noise with unknown energy [31].

Rain drops and dirt were removed by CNN learned on generated data [32]. They report significant quality increase with CNN over a patch-based network. A small 3-layer CNN with ReLUs was utilized for state-of-the-art single image super resolution [35]. A novel blind deconvolution method was proposed [36]. It incorporates a "sharpening" CNN which is trained directly to provide good blur kernel estimation. In the approach of Schuler et al., the final image reconstruction is provided by a standard non-blind deconvolution method based on the kernel estimated using the CNN.

A significant result was achieved by Hradiš et al.[3], who proposed a method for segmenting images consisting of texture regions based on a blind deconvolution setting. Based on this a segmented blind image deconvolution was proposed. It was shown that CNNs can learn to directly map blurry images to their high-quality sharp versions in a blind deconvolution setting. On the task of text document deblurring, the authors showed that CNNs significantly outperform existing blind deconvolution methods, including those optimized for text, in terms of image quality and OCR accuracy as shown in Figures 19 and 20. In fact, CNNs outperformed even state-of-the-art non-blind methods for anything but the lowest noise levels.



a)



b)

Figure 19. Result of CNN text deblurring from Hradiš et al.[3]. Top - examples of performance on real images.

[3] [Hradiš et al., 2015] is a result of ALMARVI project.

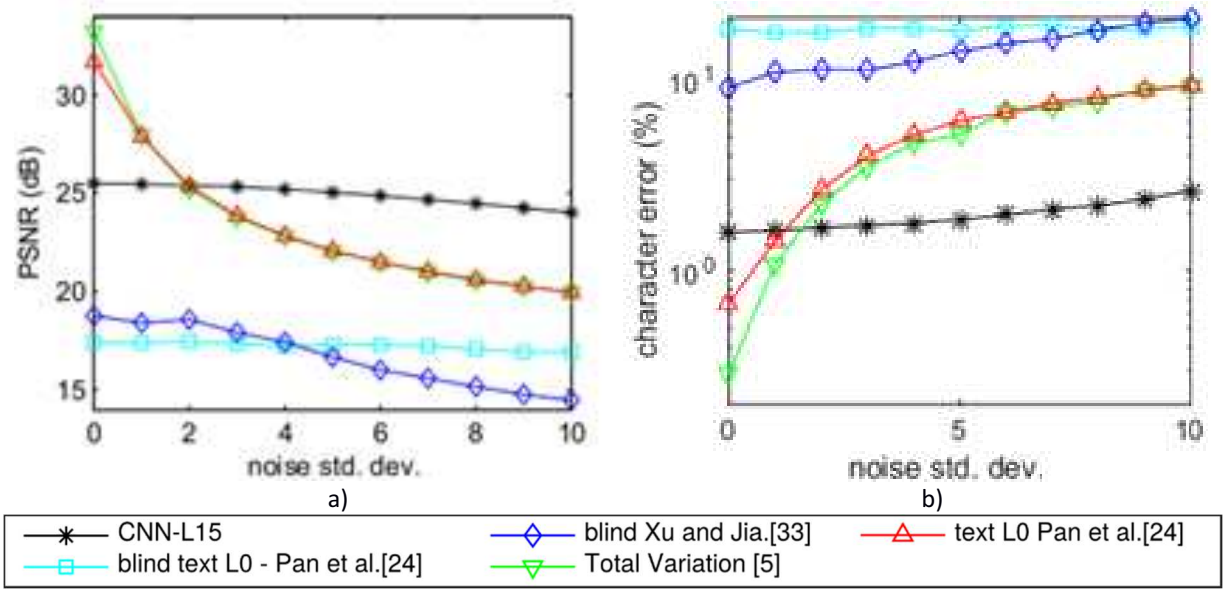


Figure 20. a) Image quality of restored images for different noise levels. b) OCR character error rate when the OCR was applied to the restored images.

In most cases, the texture segmentation using CNNs utilize only convolutional layers and element-wise non-linearities. They omit fully connected feedforward layers and max-pooling layers. Their fully convolutional structure allows them to be applied to images of any size and they produce outputs of the same size as their input due to the lack of pooling operations.

This work focused on improving the approach of Hradiš et al.[3] and extending it to domains directly relevant to the ALMARVI project. In this case, the approach to reconstruct blurry and otherwise corrupted images of license plates taken by cameras in highway surveillance and traffic monitoring systems was modified. It was noticed that the main source of image corruption in traffic monitoring systems is motion blur, weather conditions, and adverse lighting conditions.

CNNs were trained for license plate reconstruction on images artificially corrupted by motion blur, noise, and JPEG compression artefacts. As in Hradiš et al.[3], the networks were composed of multiple layers, which combine convolutions with element-wise Rectified Linear Units (ReLU):

$$F_0(y) = y \tag{8}$$

$$F_l(y) = \max(0, W_l * F_{l-1}(y) + b_l), l = 1, \dots, L-1 \tag{9}$$

$$F(y) = W_L * F_{L-1}(y) + b_L \tag{10}$$

The input and output were both 3-channel RGB images with their values mapped to interval [−0.5, 0.5]. Each layer applied c_l convolutions with filters spanning all channels $c_l - 1$ of the previous layer. The last layer was linear (without ReLU). The networks were trained by minimizing mean squared error on a dataset $D = (x_i, y_i)$ of corresponding clean and corrupted image patches:

$$\underset{W,b}{\operatorname{argmin}} \frac{1}{2|D|} \sum_{(x_i,y_i)\in D} \|F(y_i) - x_i\|_2^2 + 0.0005\|W\|_2^2 \tag{11}$$

The optimization method used was Stochastic Gradient Descent with momentum. The sixe of clear patches x_i was set to 16×16 pixels, which was believed to provide good trade-off between computational efficiency and diversity of mini-batches (48 patches in each mini-batch). The the size of clean patches x_i was set to 16×16 pixels. It was believed to provide good trade-off between computational efficiency and diversity of mini-batches (96 patches in each mini-batch). Size of the blurred training patches y_i depends on the spatial support of a particular network.

Weights from uniform distribution were initialized with variance equal to $1/n_{in}$, where n_{in} is the size of the respective convolutional filter (fan-in). This recipe was derived based on Caffe framework [36, 37].

The largest and deepest network that was trained had 15 convolutional layers and 2.3 M parameters. Compared to CNNs, used in other computer vision tasks, this network was still small and computationally efficient – it required 2.3M multiply-accumulate operations per pixel. Assuming 50% efficiency, contemporary GPUs, which provided peak single-precision speeds exceeding 4 Tflops, it should have been able to process a 1Mpx image in 2s. In case the processing was to be limited only to the regions of car license plates, this network could be easily computed in real time even on mainstream CPUs. The network was typically trained for more than a week from random weight initialization on a single GPU.

The pairs of blurred license plates and their respective reconstructions computed by CNN is shown is Figure 21.

a) *b)*

Figure 21. a) Pairs of blurred license plates and b) their respective reconstructions computed by our CNN.

Semantic Segmentation Using CNN

Semantic segmentation is a per-pixel classification where each pixel is associated with one of predefined semantic classes (object or stuff). Convolutional networks have become the state-of-the-art semantic segmentation approach. Published results of CNNs evaluated on the PASCAL Visual Object Classes [39] show a 20% relative improvement over the previous state-of-the-art method based on the combination of region proposal algorithm, CNN, and Support Vector Machine (SVM) classifier [21].

The image classification CNNs which label a whole image with a semantic class [23] are typically constrained to fixed-sized input, and they produce single set of class probabilities characterizing the whole input. Such networks can be adapted to process arbitrarily-sized images by making all their layers convolutional - that is, the fully-connected layers are converted to convolutions [38]. Such networks are similar to the image restoration networks of Hradiš et al.[3] with the addition of max-pooling layers. Such fully convolutional networks produce spatial probability maps with resolution depending on the size of an input image and the down-sampling factor due max-pooling layers in the network. Typical image classification networks down-sample inputs by factor of 32.

The small spatial resolution of outputs, provided by image classification networks adapted to fully convolutional operation, has not been sufficient for many semantic segmentation tasks, where pixel-wise segmentation would be preferred. To produce the same sized output as the input image, the CNN has to incorporate some kind of up-sampling. Simple interpolation of the low-resolution outputs is not an option as small objects and thin parts of objects have to be segmented as well. A superior approach to simple interpolation is to compute the output of a fully convolutional network (FCN) on slightly shifted inputs [40]; however, computing the full-resolution output using this technique is still prohibitively expensive and inefficient. The comparison between CNN and FCN is shown in Figure 22.

In this work it was proposed to incorporate up-sampling directly into the semantic segmentation network - they introduced backwards strided convolution (deconvolution). [39] The backwards strided convolution has to provide fractional convolution stride in the forward pass and has to reduce resolution in the backward pass. This is exactly the same as in the existing implementations of convolution for CNNs except the forward pass and backward pass are swapped. To provide better spatial localization of object boundaries, Long et al. add links that combine the final prediction layer with lower layers with finer strides. Combining fine layers and coarse layers lets the model make local predictions that respect global structure. With the network architecture [39], processing time of images from PASCAL VOC with average resolution of 0.18 Mpx is in the range of tens up to hundreds of millisecond on a high-end GPU. Memory requirements depends mainly on the exact network architecture and ranges from megabytes to hundreds of megabytes.

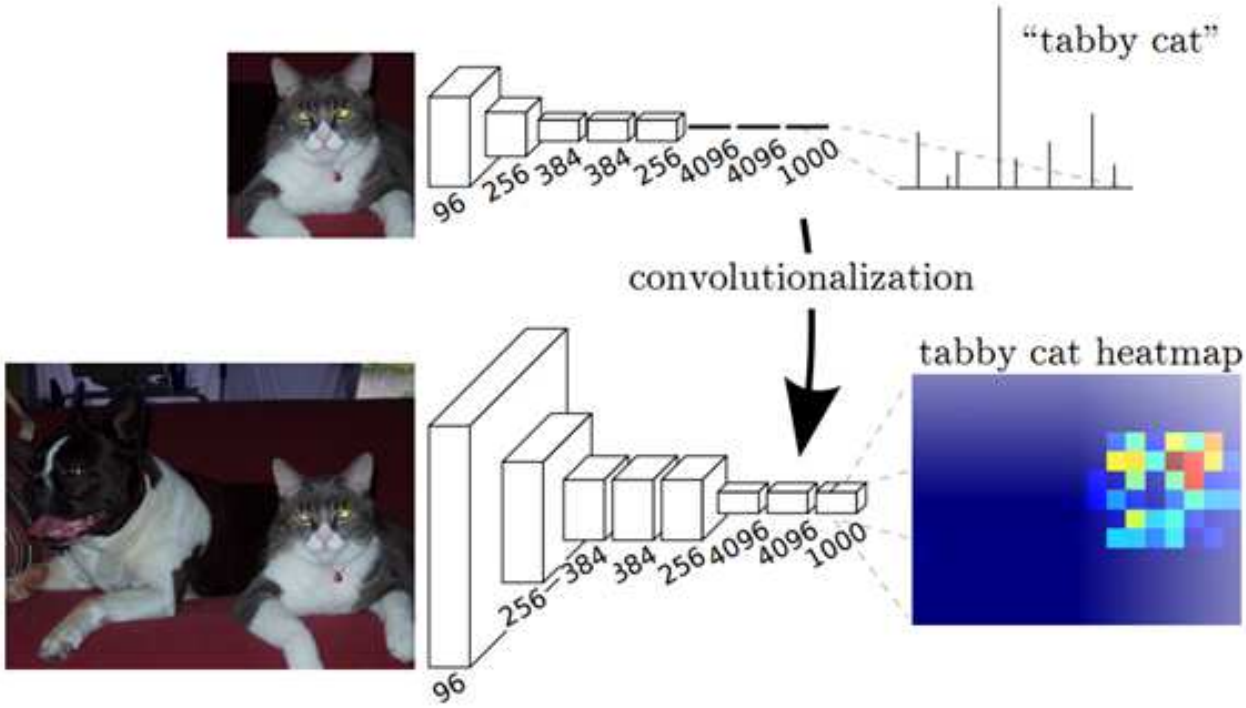


Figure 22. Convolutional Neural Network (CNN) vs. Fully Convolutional Network (FCN). [39]

The pixel-wise semantic segmentation requires per-pixel labelled images for training. Such annotation is highly labour intensive to create and should ideally be avoided in large-scale training. Therefore an approach to train pixel-wise segmentation without the need of pixel-wise annotations was proposed. [40]

It was proposed to annotate only small number of objects and to learn pixel-wise segmentation from such data by incorporating objectness prior into the CNN cost function. The objectness prior captures probability that a pixel belongs to any object class. As the difference is in the training stage and not the feed-forward stage, the run time characteristics of a CNN trained this way does not change and depends only on the particular network architecture.

Pseudolabel Approach for CNN

Traditionally, visual categories could be learned by Support Vector Machines on histograms of local features [42]. Current approaches have shifted towards CNNs [23, 40, 29], which require vast amounts of data and computational power to learn millions of parameters. The CNN approach is shown in Figure 23. Such approaches have achieved near-human performance in face recognition [43], and have beaten previous approaches in image classification of both very broad and very specific categories [41].

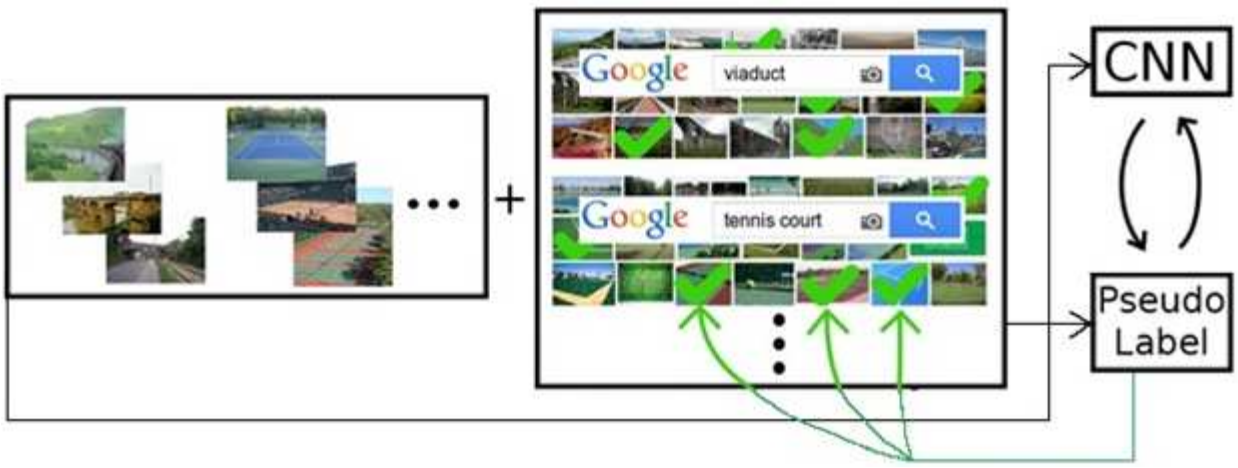


Figure 23. Convolutional Neural Network approach has achieved near-human performance in face recognition, and have beaten previous approaches in image classification of both very broad and very specific categories.

Deep Learning relies on large labelled datasets, with several hundred images for each category, but the creation of such datasets is demanding. Imperfect datasets can be created cheaply in an automated fashion, but near-perfect labelling, required by current approaches, relies on manual selection. One way to do Deep Learning on small datasets is to initialize network parameters from an existing network. Networks have been shown to produce excellent embeddings, which generalize well to new categories [29, 44]. However, this approach is limited, and a larger dataset would further improve results.

Therefore, there is a natural need for an approach which couples cheap automatically retrieved weakly labelled images with pre-initialized CNNs. Pseudolabel [45] is such an approach. However, it remains to be adopted to a large, challenging classification dataset. Pseudolabel exploits the iterative nature of Neural Network training, and expands a small set of correctly labelled training data with some of the imperfectly labelled training data. This approach selects samples from the imperfect dataset to best supplement the correct data.

We adapted and extended the Pseudolabel approach, allowing for use on web-scale datasets of millions of images. The accuracy distribution across classes with and without pseudolabels is shown in Figure 24. The results are demonstrated on a toy problem devised from the SUN 397 dataset, and on the full SUN 397 dataset expanded with images gathered from Google's image search without human intervention. The toy problem allows us to analyse the properties of the data selection progress during training, The query set were retrieved from Google's image search separately for each category, by searching the full name of the SUN 397 category (ex.: ``swimming pool indoor''), and retrieving all full scale original images. Only images which produced an erroneous http query were ignored, and the number of images found was between 230 and 1359, with mean 796. Using these findings, state-of-art accuracy is achieved on the full dataset. Train and test images are retrieved from the SUN 397 dataset. These are divided into a train set and test set randomly, by using n for training, and the rest for testing. We performed experiments with n = [5, 20, 50]. The query set were retrieved from Google's image search separately for each category, by searching the full name of the SUN 397 category (ex.: ``swimming pool indoor''), and retrieving all full scale original images.

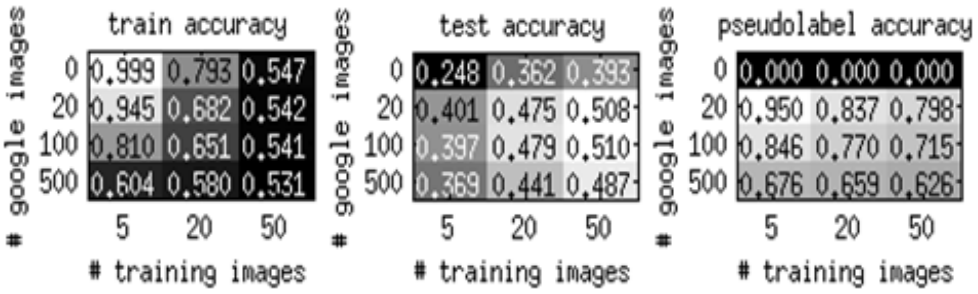


Figure 24. The accuracy distribution across classes with and without pseudolabels. Note that the quality of retrieved images decreases with additional images, offsetting the benefit from Pseudo-Labels on larger queries.

By adapting pseudo-labels to real-world datasets, ground breaking results have been accomplished, facilitating progress in classification and localization where image data is sparse. The method was justified, experimentally analysed, and validated. Finally, state-of-art results were presented on the SUN 397 dataset with few images in each category.

3.8 Machine Learning Based Scene Analysis Using Modern Methods

Linear Support Vector Machine

Support Vector Machines (SVM) are popular algorithms for machine learning for tasks such as classification. [46] The first SVM was introduced in 1979 by Vapnik [47]. Since then it has been it has received numerous updates. The SVM that's close to the modern SVM's was introduced by Boser, Guyon and Vapnik in 1992. [48] Currently several image segmentation approaches based on SVM have been developed.

In this research work a linear SVM for the classification task for scalability on large datasets was chosen to be utilized. The chosen algorithm was implemented by scikit-learn library. [49] It was trained with Histogram of Oriented Gradients (HOG) features of 12 624 positive samples and 50 000 negative samples.

It was observed that the SVM had about 95 % reliability against test dataset with 3 705 positive and 50 000 negative samples. In live situations the reliability was lower due to more noise and the variation between the camera angles in the training data versus the real angle of the camera that was capturing the video feed. The different angle affected the false positives more than the false negatives meaning that the SVM saw non-human objects as humans more often than classifying humans as non-human objects. HOG was used for feature extraction The HOG features were calculated from grayscale images that were obtained from cropping the image from the extracted foreground layer. The HOG feature presentation of the image, cropped from the original video feed using the foreground layer obtained from background subtraction, is shown in Figure 25.

To reduce the amount of false positives, a second self-learning SVM was introduced that kept track of the found humans and gave them classification based on the order they were first started to being tracked. If the first SVM claimed to have found a human but the second SVM could not verify it, the result was discarded. The second SVM wasn’t yet very reliable, it verified the false positive about half the time, but the overall false positive count was lowered.

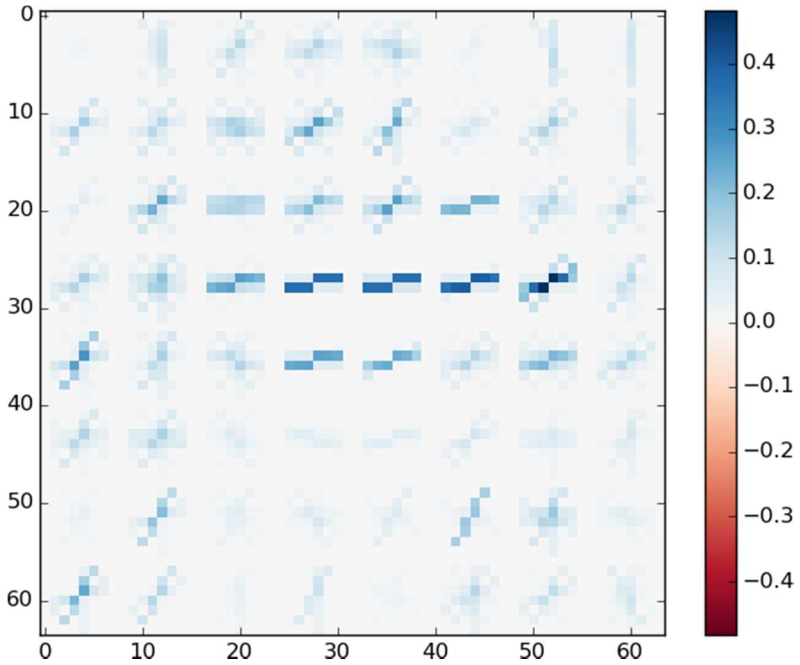


Figure 25: The HOG feature presentation of the image (see Figure 14), cropped from the original video feed using the foreground layer obtained from background subtraction. Values ranging from 0.0 to 1.0. The image was resized: 64x64 pixels.

The SVM is run on a server machine which combines the acquired data from the Raspberry Pis. The software as whole can be used in any OS or hardware that has Python support. For optimal power consumption Raspberry Pi2 is used for the frontend. For the backend a desktop PC is used but it can be replaced by a cloud service. The frontend consist of the background subtractor and the morphology methods. The backend only houses the SVM and as such can be run on external server or in a cloud. These fall within the plans to use Raspberry Pi and a server machine which were set at the beginning.

4. Conclusions and Future Work

This document provided architecture and design details of advanced algorithms/methods for image/video segmentation, feature extraction, and clustering. The key focus was on developing algorithms for massive data-rate image/video processing and analysis, tailored towards the specific ALMARVI requirements and system specifications to support for performance and power scalability.

The high-performance and low-power algorithms described in this document were amenable to parallelization using ALMARVI multi-/many-core execution platform instances (including CPU + GPU/FPGA + TTA + rVEX). Based on this, parallelized low-power algorithms and implementations for different image/video processing functions were explained in this deliverable. Furthermore, the work described in this document was aimed at presenting challenges in the low quality images/videos in surveillance and healthcare applications. Thus, solutions to the enhancement and evaluation of the image/video quality were proposed in which different advanced image segmentation, feature extraction, and clustering methods were utilized.

The four objectives of the ALMARVI project were taken into account in Task 2.3, although the main focus in Task 2.3 is related to the objectives 1, 2, and 3. The algorithms/methods in Task 2.3, which achieve all or some of these objectives are listed below:

1. *Enabling Massive Data Rate Processing:* All algorithms/methods in Task 2.3 achieve this objective (see Sections 3.1-3.8).
2. *Achieving Low Power Consumption:* All algorithms/methods in Task 2.3 achieve this objective (see Sections 3.1-3.8).
3. *Composability, Flexibility, and Cross-Domain Applicability:* Level-sets, snakes, graph-cuts, and distance transform based methods (see Section 3.4), Mathematical morphology based methods (see Section 3.5), Hardware-limited segmentation methods (see Section 3.6), Convolutional neural networks (see Section 3.7), and Machine learning based scene analysis using modern methods (see Section 3.8). Thus, almost all algorithms/methods in Task 2.3 achieve this objective.
4. *Robustness to Variability:* Hardware-limited segmentation methods (see Section 3.6), Convolutional neural networks (see Section 3.7), and Machine learning based scene analysis using modern methods (see Section 3.8). Thus, only three algorithms/methods in Task 2.3 achieve this objective, but it will be more comprehensively achieved in the next steps of the research and development work.

The partners involved in this deliverable were UEF, UTIA, ASEL, BUT, and HURJA. The results of this task will be employed in healthcare and security/surveillance demonstrators (Tasks 5.1 and 5.2) in which all four ALMARVI objectives will be achieved more comprehensively.

Our future research and development work will enable cross-domain applicability, composability, flexibility, and interoperability in order to expand the business horizon over different product categories and different application domains without completely re-designing the whole system. However, implementations of different image/video processing and analysis algorithms/methods, described in this deliverable, need to be further optimized for the final ALMARVI execution platform. Furthermore, the objective 4, robustness to variability, will be more comprehensively achieved in the next steps of the research and development work.

5. References

[1] P. KaewTraKuPong and R. Bowden. An Improved Adaptive Background Mixture Model for Real-Time Tracking with Shadow Detection. Springer Link, pp 135-144, 2001

[2] http://opencv.org/

[3] K. Fukunaga and L. Hostetler, The Estimation of the Gradient of a Density Function with Applications in Pattern Recognition, IEEE Transactions on Information Theory. Vol 21, pp 32-40, 1975

[4] D. Comaniciu and P. Meer, Mean Shift: A Robust Approach Towards Feature Space Analysis, IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol 24 No 5 May 2002

[5] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk, SLIC Superpixels Compared to State-of-the-Art Superpixel Methods, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 34, no. 11, Nov. 2012

[6] L. Ikonen: Distance Transforms on Gray-Level Surfaces. Doctoral Dissertation, Lappeenranta University of Technology, June 15, 2006.

[7] P. Andrey and T. Boudier: Adaptive Active Contours (Snakes) for the Segmentation of Complex Structures in Biological Images. First Image J User and Developer Conference, Luxemburg, Germany, May 18-19, 2006.

[8] J.A. Sethian: Level Set Methods and Fast Marching Methods. Cambridge University Press, Cambridge, 2nd Edition, 1999.

[9] T.F. Chan and L.A. Vese. A Level Set Algorithm for Minimizing the Mumford-Shah Functional in Image Processing. IEEE Workshop on 2001 in Vancouver.

[10] D. Mumford and J.S. Optimal Approximations by Piecewise Smooth Functions and Associated Variational Problems. Communication on Pure and Applied Mathematics. Vol XLII 577-685, 1989, John Wiley & Sons

[11] J. Serra, P. Soille, Mathematical Morphology and Its Applications to Image Processing, Springer Science & Business Media, 2012

[12] V. Vizilter, Y.P. Pytev, A.I. Chulichkov and L.M. Mestetskiy Morphological Image Analysis for Computer Vision Applications. 2014

[13] F. Y. Shih, Image processing and Mathematical Morphology: Fundamentals and Applications, CRC Press, 2009

[14] B. Xiangzhi, Z. Fugen, X. Bindang, Multiple Linear Feature Detection Through Top-Hat Transform by Using Multi Linear Structuring Elements, Optik – International Journal for Light and Electron Optics, Vol. 123, pp. 2043-2049, Novermber 2012

[15] Ozan Yardimci, Seyit Tunc, Ilkay Ulusoy Pamas: Performance and Time Requirement Analysis of Top-Hat Transform Based Small Target Detection Algorithms, SPIE Proceedings Vol. 9476, Automatic target Recognition XXV, 28 May 2015. Doi: 10.1117/12.2176325

[16] Q. Shan, J. Jia, and A. Agarwala. High-Quality Motion Deblurring from a Single Image. In ACM Transactions on Graphics (TOG), Vol. 27, page 73. ACM, 2008

[17] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. CoRR, abs/1409.1, 2014

[18] L. Zhong, S. Cho, D. Metaxas, S. Paris, and J. Wang. Handling Noise in Single Image Deblurring Using Directional Filters. In CVPR, 2013

[19] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In CVPR, 2014

[20] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deep-Face: Closing the Gap to Human-Level Performance in Face Verification. In CVPR, 2014

[21] A. Krizhevsky, I. Sutskever, and G.E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks NIPS 2012: Neural Information Processing Systems, Lake Tahoe, Nevada, 2012

[22] Q. Shan, J. Jia, and A. Agarwala. High-quality Motion Deblurring from a Single Image. In ACM Transactions on Graphics (TOG), Vol. 27, page 73. ACM, 2008

[23] D. Strigl, K. Kofler, and S. Podlipnig. Performance and Scalability of GPU-Based Convolutional Neural Networks. In Proceedings of the 18th Euromicro Conference on Parallel, Distributed and Network-based Processing, 2010

[24] S. Zhang, C. Zhang, Z. You, R. Zheng, and B. Xu, Asynchronous Stochastic Gradient Descent for DNN Training, 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 26-31 May 2013

[25] W. Chan and I. Lane. Distributed Asynchronous Optimization of Convolutional Neural Networks. INTERSPEECH, 2014
[26] K. Ovtcharov, O. Ruwase, J.-Y. Kim, J. Fowers, K. Strauss, and E.S. Chung. Accelerating Deep Convolutional Neural Networks Using Specialized Hardware. Microsoft Research, 2015
[27] J. Donahue and Y. Jia and O. Vinyals and J. Hoffman and N. Zhang and E. Tzeng and T. Darrell: DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. ICML, 2014
[28] M. Egmont-Petersen, D. de Ridder, and H. Handels. Image Processing with Neural Networks – A review. Pattern Recognition 35(2002) 2279–2301
[29] H. C. Burger, C.I. Schuler, and S. Harmeling. Image denoising: Can Plain Neural Networks Compete with BM3D? In CVPR, 2012.
[30] V. Srinivasan, P. Bhatia, S.H. Ong. Edge Detection Using a Neural Network. 1994. Elsevier Science. Vol. 27, Issue 12, December 1994, Pages 1653–1662
[31] V. Jain and S. Seung. Natural Image Denoising with Convolutional Networks. In NIPS, 2009
[32] D. Eigen, D. Krishnan, and R. Fergus. Restoring an Image Taken through a Window Covered with Dirt or Rain. In: ICCV, 2013
[33] C.J. Schuler, H. Chrisopher Burger, S. Harmeling, and B. Scholkopf. A Machine Learning Approach for Non-blind Image Deconvolution. In CVPR, 2013
[34] L. Xu, J.SJ. Ren, C. Liu, and J. Jia. Deep Convolutional Neural Network for Image Deconvolution. In Advances in Neural Information Processing Systems (NIPS), 2014
[35] C. Dong, C.C. Loy, K. He, and X. Tang. Learning a Deep Convolutional Network for Image Super-Resolution. In ECCV, pages 184–199. Springer International Publishing, 2014
[36] C.J. Schuler, M. Hirsch, S. Harmeling, and B. Schölkopf. Learning to Deblur. CoRR, abs/1406.7, 2014
[37] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional Architecture for Fast Feature Embedding. arXiv preprint arXiv:1408.5093, 2014
[38] X. Glorot and Y. Bengio. Understanding the Difficulty of Training Deep Feedforward Neural Networks. In Proceedings of the International Conference on Artificial Intelligence and Statistics, 2010
[39] J. Long, E. Shelhamer, E. Darrel. Fully Convolutional Networks for Semantic Segmentation. In CVPR, 2015
[40] P. Sermanet, D. Eigen, X. Zhang, M. Mathiew, R. Fergus, L. Cun. OverFeat: Integrated Recognition, Localization and Detection Using Convolutional Networks. CoRR, abs/1312.6229, 2013
[41] O. Russakovsky, A. Beramon, V. Ferrari, L. Fei-Fei. What's the Point: Semantic Segmentation with Point Supervision. ArXiv e-prints, 2015
[42] Van De Sande, Koen EA, T. Gevers, and C. GM Snoek. Evaluating Color Descriptors for Object and Scene Recognition. Pattern Analysis and Machine Intelligence, IEEE Transactions on 32.9 (2010): 1582-1596
[43] Y. Taigman, M. Yang, M. Aurelio Ranzato, and L. Wolf. Deepface: Closing the Gap to Human-Level Performance in Face Verification. IEEE. 2014
[44] A.S. Razavian, et al. CNN Features Off-the-Shelf: An Astounding Baseline for Recognition. Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on. IEEE, 2014
[45] L. Dong-Hyun. Pseudo-label: The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks. Workshop on Challenges in Representation Learning, ICML. Vol. 3. 2013
[46] C-C. Chang and C-Jen Lin. LIBSVM: A Library for Support Vector Machines. Department of Computer Science National Taiwan University, Taipei, Taiwan
[47] http://www.svms.org/history.html
[48] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A Training Algorithm for Optimal Margin Classifiers. In D. Aussler, editor, 5th Annual ACM Workshop on COLT, pages 144{152, Pittsburgh, PA, 1992. ACM Press
[49] http://scikit-learn.org/