Izmir Institute of Technology
Department of Computer Engineering
CENG415 Senior Design Project Seminar 1

re$\sqrt{ision}$

Renderscript for Computer Vision

Anıl Can Aydın, Onur Temizkan, Ulaş Akdeniz
180201060, 180201004, 180201063

supervised by
Asst. Prof. Dr. Mustafa Özuysal

May 25, 2016

Contents

1 Description of the Work

Computer vision applications need parallel computation to perform in reasonable response time. In order to satisfy that performance, developers use libraries like OpenCL and CUDA which move the computation load to GPU. However high performance GPU intensive computation on Android platform is a tough issue because of hardware dependencies and incompatibilities of libraries. For instance OpenCL GPU computation library is not supported on all Android devices. So, creating hardware independent applications using OpenCL is simply impossible. Because of those restrictions mentioned above, Renderscript computation module is presented by Google in 2011[1]. It is a hardware-independent computation engine that operates at the native level[2]. But there is no vision library written in Renderscript.

reVision is an experimental Computer Vision project that strives to prove performance benefits of using Renderscript on Android applications. This project includes the performance comparison between Renderscript, OpenCV, and Java implementations on a Computer Vision algorithm. For this purpose, Harris Corner and Edge Detector Algorithm[3] has chosen.

Although, Renderscript is presented in 2011, it is not a commonly used option to make computationally intensive tasks. It is a considerable approach to use Renderscript for Computer Vision, but there are not enough data to support this idea. To provide a reliable analysis, performance of Renderscript on Computer Vision is compared with OpenCV[4] which is a de facto standard in its domain. Also it is compared with Java, because of both running on CPU and being the main language of Android platform.

2 Project Management

2.1 Overall Strategy

reVision's iterative development process consists of project management and planning, requirement analysis, implementation and testing. Each iteration results in a more optimized version of each implementation.

Summary of an iteration is described below.

Planning: The scope of the implementations to be written, desired specs and details are discussed.

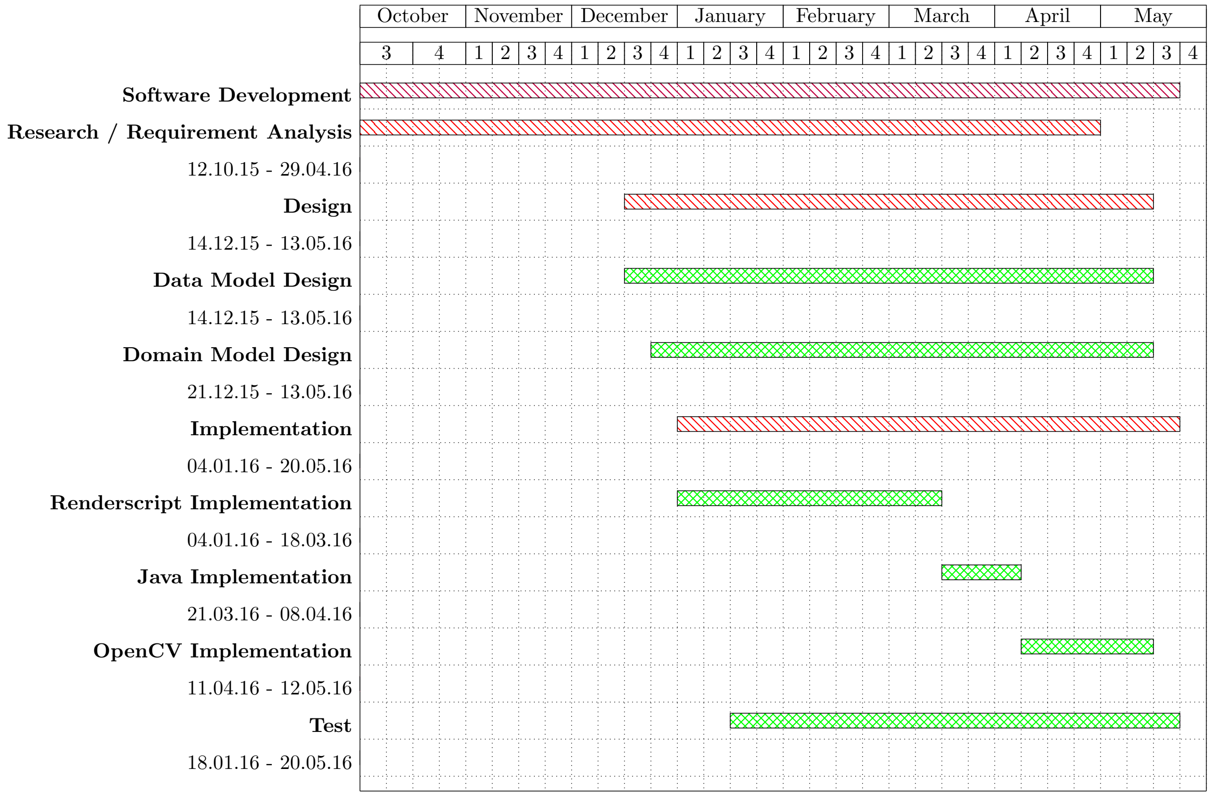
Requirement Analysis: The requirements of the implementations, needed arguments and optional specs are discussed.

Implementation: The sample application for each variation is implemented.

- Obtaining consistent results for each implementation is main objective of every iteration.
- Proceed every iteration with phases such as: Design, Implement, Test and Evaluate.

Testing: Sample applications are tested on different devices in different resolutions.

2.2 Gantt Chart



2.3 Detailed Work Description

2.3.1 Work Package List

Work Package No	Work Package Title	Type of Activity[1]	Lead Participant No	Lead Participant Short Name	Person weeks[2]	Start Month	End Month
WP-1	Research and Requirement Analysis	MGT	1-2-3	AA-OT-UA	26	1	8
WP-2.1	Data Model Design	SUPP	1-2-3	AA-OT-UA	20	3	8
WP-2.2	Domain Model Design	SUPP	1-2-3	AA-OT-UA	19	3	8
WP-3.1	Renderscript Implementation	SUPP	1-2-3	AA-OT-UA	10	4	6
WP-3.2	Java Implementation	SUPP	1-2-3	AA-OT-UA	3	6	7
WP-3.3	OpenCV Implementation	SUPP	1-2-3	AA-OT-UA	5	7	8
WP-4	Testing	MGT	1-2-3	AA-OT-UA	18	4	8
TOTAL					**101**		

[1] **SUPP** stands for Support activities; **MGT** stands for Management of the consortium.
[2] The total number of person-weeks allocated to each work package.

2.3.2 Deliverable List

Deliverable No	Deliverable Name	WP No	Nature[1]	Dissemination Level[2]	Delivery Date
D-1	Requirement Design Documents	WP-1	R	CO	24.12.2016
D-2.1	Data Model Design	WP-2.1	R	CO	13.05.2016
D-2.2	Domain Model Design	WP-2.2	R	CO	13.05.2016
D-3.1	Renderscript Implementation	WP-3.1	P	PU	18.03.2016
D-3.2	Java Implementation	WP-3.2	P	PU	08.04.2016
D-3.3	OpenCV Implementation	WP-3.3	P	PU	12.05.2016

2.3.3 Milestones List

Milestone Number	Milestone Name	Work Package(s) Involved	Expected Date	Means of Verification
M1	Renderscript Application	WP-3.1	18.03.2015	Validated by supervisor
M2	Java Application	WP-3.2	08.04.2016	Validated by supervisor
M3	OpenCV Application	WP-3.3	12.05.2016	Validated by supervisor

[1] **R** = Report, **P** = Prototype, **D** = Demonstrator
[2] **PU** = Public, **CO** = Confidential, only for members of the consortium(including the Commission Services)

2.3.4 Work Package Descriptions

Work Package Number	WP-1	**Start Date:**		12.10.2015
Work Package Title	Research and Requirement Analysis			
Activity Type	MGT			
Participant Number	1	2	3	
Participant Short Name	AA	OT	UA	
Person-weeks per participant:	26 weeks	26 weeks	26 weeks	

Objectives

- Determining the project scope, gathering information about the project, determining requirements and analysis.

Description of work
Task-1: Determine the project scope
Task-2: Research about similar works
Task-3: Making a requirement analyses.
Task-5: Determine the main modules.

Deliverables

- Documentation of project description, its scope and requirements.

Work Package Number	WP-2.1	**Start Date:**		14.12.2015
Work Package Title	Data Model Design			
Activity Type	SUPP			
Participant Number	1	2	3	
Participant Short Name	AA	OT	UA	
Person-weeks per participant:	20 week	20 week	20 week	

Objectives

- Determining data structures that can be commonly used with computer vision algorithms.

Description of work
Task-1: Decide data structures
Task-2: Discuss the functionality

Deliverables

- Data model.

Work Package Number	WP-2.2	**Start Date:**		21.12.2015
Work Package Title	Domain Model Design			
Activity Type	SUPP			
Participant Number	1	2	3	
Participant Short Name	AA	OT	UA	
Person-weeks per participant:	19 weeks	19 weeks	19 weeks	

Objectives

- Determine the classes and their attributes.

Description of work
Task-1: Determine the classes and their attributes.
Task-2: Determine the relationships among classes

Deliverables

- Domain model.

Work Package Number	WP-3.1	**Start Date:**		04.01.2016
Work Package Title	Renderscript Implementation			
Activity Type	SUPP			
Participant Number	1	2	3	
Participant Short Name	AA	OT	UA	
Person-weeks per participant:	10 weeks	10 weeks	10 weeks	

Objectives

- Implement Harris Corner Detection Algorithm with Renderscript.

Description of work

Task-1: Renderscript implementation.

Deliverables

- Renderscript sample application

Work Package Number	WP-3.2	**Start Date:**		21.03.2016
Work Package Title	Java Implementation			
Activity Type	SUPP			
Participant Number	1	2	3	
Participant Short Name	AA	OT	UA	
Person-weeks per participant:	3 weeks	3 weeks	3 weeks	

Objectives

- Implement Harris Corner Detection Algorithm with Java.

Description of work

Task-1: Java implementation.

Deliverables

- Java sample application

Work Package Number	WP-3.3	**Start Date:**		11.04.2016
Work Package Title	OpenCV Implementation			
Activity Type	SUPP			
Participant Number	1	2	3	
Participant Short Name	AA	OT	UA	
Person-weeks per participant:	5 weeks	5 weeks	5 weeks	

Objectives

- Implement Harris Corner Detection Algorithm with OpenCV.

Description of work

Task-1: OpenCV implementation.

Deliverables

- OpenCV sample application

Work Package Number	WP-4	**Start Date:**		18.01.2016
Work Package Title	Testing			
Activity Type	SUPP			
Participant Number	1	2	3	
Participant Short Name	AA	OT	UA	
Person-weeks per participant:	18 weeks	18 weeks	18 weeks	

Objectives

- Test applications on different devices with different resolutions.

Description of work

Task-1: Compare the performances of each implementation.

Deliverables

- Comparison results

2.3.5 Summary Effort Table

P. short name	WP-1	WP-2.1	WP-2.2	WP-3.1	WP-3.2	WP-3.3	WP-4	Total person weeks
AA	26	20	19	10	3	5	18	101
OT	26	20	19	10	3	5	18	101
UA	26	20	19	10	3	5	18	101

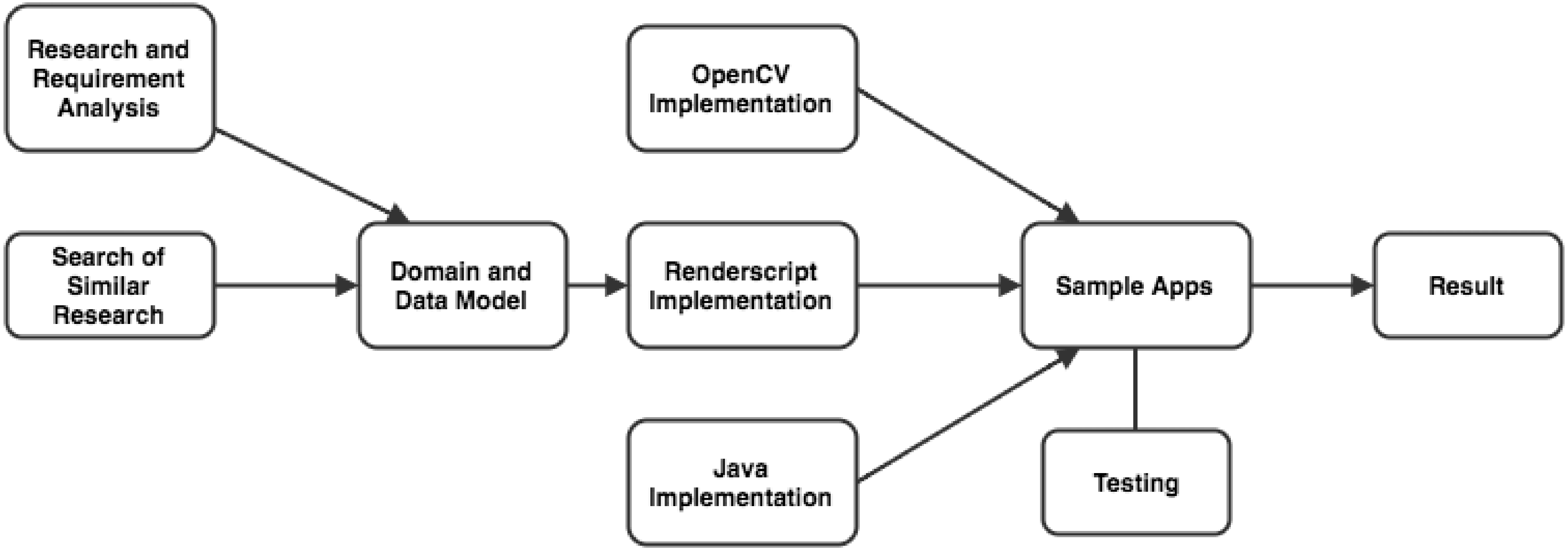


Figure 1: svg image

2.4 Pert Diagram

2.5 Risk Management

The development team is unexperienced on computer vision field, so some learning period is required in multiple phases of the project. Especially developing modules that contain algorithms having steep learning curve will take remarkable time in the process. For this reason it is crucial to comply with the schedule.

3 Methodology and Analysis

3.1 Feasibility Study

reVision is an open source and non-profit project so there is no need to examine the financial feasibility. The project is analyzed only in operational and technical sides.

3.1.1 Operational Feasibility

Computer vision developers feel the lack of a decent computer vision library on Android platform. Creating a high performance computer vision application on Android takes too much effort. This project, if succeeds will become a ready-to-use and collaborative tool for Android developers.

3.1.2 Technical Feasibility

Widely used libraries for computer vision and GPU computing such as OpenCV, OpenCL, and CUDA, cannot be used on Android platform because of incompatibilities of devices and lack of drivers. For this reason a new GPU computing engine named Renderscript was presented. But Renderscript does not have a complete computer vision library.

reVision uses Renderscript for data-parallel GPU computations. Since Renderscript technology is hardware and version independent among all recent Android devices, reVision can also be used universally on every Android device. The runtime frame-rates vary between different devices but reVision is ensured to work at the most efficient way that the device is capable of.

3.2 The Process Model and Its Particular Adaptation

According to R. S. Pressman [5] there are several process model types including linear sequential model, prototype model, evolutionary models and etc. In real life most them are used according to the project necessities and requirements.

reVision is being developed using incremental model which is an evolutionary model. Every release is published after a work cycle which includes all the phases of the software development process. Each release extends the project with at least one module or functionality with its sample application.

Incremental model is suitable for this project because the project team is not experienced in computer vision field. The incremental model gives the option of refactoring the design and evolving the project through time to the team, which other models would not give.

Development cycle can simply be clarified with following steps; Requirement analysis, Planning, Design, Implementation and Testing. Every development iteration of reVision include each one of those steps.

3.3 Functional and Non-Functional Requirements

3.3.1 Functional Requirements

Filtering: Image filtering is required for other modules to provide initial data. It also can be used as standalone to get filtered image.

Feature Detection: Feature detection is one of the outputs of the library which provides marked image matrix of the detected features such as edges, corners, blobs.

Object Tracking: Object tracking is one of the features of the library which detects the object and tracks the object in the given stream.

```
uchar4 __attribute__((kernel)) harris(const uchar4 in,
  uint32_t x, uint32_t y)
{
    float c = 0.04;
    float4 convXpixel =
      rsUnpackColor8888(rsGetElementAt_uchar4(convX, x, y));
    float4 convYpixel =
      rsUnpackColor8888(rsGetElementAt_uchar4(convY, x, y));

    float Ix = convXpixel.r * gMonoMult[0] + convXpixel.g *
      gMonoMult[1] + convXpixel.b * gMonoMult[2];
    float Iy = convYpixel.r * gMonoMult[0] + convYpixel.g *
      gMonoMult[1] + convYpixel.b * gMonoMult[2];

    float Ixx = Ix * Ix;
    float Iyy = Iy * Iy;
    float Ixy = Ix * Iy;
    float cornerResponse =
      (Ixx*Iyy - Ixy*Ixy - c*(Ixx+Iyy)*(Ixx+Iyy));

    if(cornerResponse < harrisThreshold ) {
        cornerColorRGB.r = 0;
        cornerColorRGB.g = 255;
        cornerColorRGB.b = 0;
        return cornerColorRGB;
    } else {
        return in;
    }
}
```

Listing 1: Harris Kernel from the first prototype

3.3.2 Non-Functional Requirements

Performance Requirements: The response time problem were the thing that triggered the creation of reVision project. Without reasonable performance, reVision could not be considered successful. To achieve the desired performance, developers implement and use all performance related modules such as implementing Async Model or using Renderscript framework. The performance is the single most important non-functional requirement of reVision.

Reliability Requirements: The responses of reVision modules must be correct and precise.

Usability Requirements: The function arguments and the Android communication modules must be easy to understand and to use. Since it is a library, every function has to be well documented. Also the sample applications have to be complete and ready to test for users.

3.4 Use Cases

reVision is an experimental project to provide analysis results of Computer Vision implementations of different technologies. It is not intended to be used by end users. Therefore there is no use case to share.

4 Planned Solution/Product

reVision is planned to be an open source computer vision library that can operate on all Android devices. By using the power of Renderscript, it will provide ready to use computer vision algorithm implementations with their sample applications.

Computer vision algorithms will be implemented in Renderscript. When the Renderscript was presented, the goal of it had been to bring a higher performance API to Android developers [1]. So, the goal of reVision is to present a higher level, closer to Renderscript performance, and easy-to-use API to Android developers.

Sample applications are planned to exemplify all features of the library in order to guide the developers on their developing process. Figure 1 indicates a couple of pictures from first prototype of reVision. Image on the bottom is a Harris Algortihm applied form of the top.

clean.png detected.png

Figure 2: First prototype of reVision: Implementation of Harris Corner Detection

Eventually, reVision will be a solution to absence of an easy-to-use, and well performanced library operates on all Android devices.

5 Related Work/Similar Solutions

There are several libraries about the vision. One of the most significant of them is OpenCV.

opencv.png

OpenCV is released under a BSD license and hence its free for both academic and commercial use. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform. Adopted all around the world, OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 9 million. Usage ranges from interactive art, to mines inspection, stitching maps on the web or through advanced robotics [4].

Another computer vision library is the Qualcomm's FastCV™ [6]. FastCV™ library offers a mobile-optimized computer vision (CV) library which includes the most frequently used vision processing functions for use across a wide array of mobile devices, even mass-market handsets. Middleware developers can use FastCV to build the frameworks needed by developers of computer vision apps; Qualcomm's Augmented Reality (AR) SDK is a good example. Developers of advanced CV application can also use FastCV functions directly in their application. FastCV will enable you to add new user experiences into your camera-based apps like:

fastcv.jpg

- gesture recognition
- face detection, tracking and recognition
- text recognition and tracking
- augmented reality

Also Cuda™ [7] can be mentioned as a similar work. CUDA™ is a parallel computing platform and programming model invented by NVIDIA. It enables dramatic increases in computing performance by harnessing the power of the graphics processing unit (GPU). To use CUDA on your system, you will need the following:

- Android development device with a CUDA-capable GPU
- A supported version of Linux to cross-compile
- NVIDIA CodeWorks for Android with CUDA support

6 Impact

In the impact section, realistic constraints of the project will be mentioned. Besides being a thesis project, reVision is a nonprofit project as a matter of course, aims to make the Android world a better place for computer vision application developers. Thus, there are several constraints about the project. When compared to similar works, reVision is maintained by students instead of groups of experienced developers or big companies. So, this situation limits the scope of the project, but it also motivates its participants about the dream of taking part in a project that can be a milestone as well.

reVision is also an opportunity to learn new technologies and methodologies for its developers. It focuses on the computer vision and its applications using mobile devices, and data-parallel GPU computation. So, project team participants get a chance to improve themselves in these topics. The project will help the developers to understand the formal procedures of writing formal reports. Nearly all the software engineering processes are revisited.

All in all, reVision is a hardware independent computer vision library with companion sample applications for Android which will be an open source and GPU-ready, and strives to make Android developers more productive on computer vision projects.

References

[1] Introducing Renderscript - http://android-developers.blogspot.com.tr/2011/02/introducing-renderscript.html

[2] Renderscript - http://developer.android.com/intl/es/guide/topics/renderscript/compute.html

[3] Harris, C. and Stephens, M. 1988. Acombined corner and edge detector. In Fourth Alvey Vision Conference, Manchester, UK, pp. 147151.

[4] OpenCV - http://opencv.org/

[5] Roger S. Pressman, Software Engineering A Practitioner's Approach, McGraw-Hill, 2001, pp. 26-39

[6] FastCV - https://developer.qualcomm.com/software/fastcv-sdk

[7] Cuda - http://docs.nvidia.com/gameworks/content/technologies/mobile/cuda_android_main.htm