Information and Communication Technologies (ICT)

Programme

Project Nº: H2020-ICT-2016-1-732105

D6.9: Ocean Monitoring (Ver. I)

Lead Beneficiary:	9 – AS
Work Package:	6
Date:	31-July-18
Distribution - Confidentiality:	[Public]

Abstract:

This document contains the description of the M18 Ocean Monitoring demonstrator. The description is based on the skeleton defined in D6.7 and includes the scope and purpose of the demonstrator, the description of the demonstrator itself and the accomplished results till now.

Disclaimer

The CERBERO Consortium is the following:

Num.	**Beneficiary name**	**Acronym**	**Country**
1 (Coord.)	IBM Israel – Science and Technology LTD	IBM	IL
2	Università degli Studi di Sassari	UniSS	IT
3	Thales Alenia Space Espana, SA	TASE	ES
4	Università degli Studi di Cagliari	UniCA	IT
5	Institut National des Sciences Appliques de Rennes	INSA	FR
6	Universidad Politecnica de Madrid	UPM	ES
7	Università della Svizzera italiana	USI	CH
8	Abinsula SRL	AI	IT
9	Ambiesense LTD	AS	UK
10	Nederlandse Organisatie Voor Toegepast Natuurwetenschappelijk Onderzoek TNO	TNO	NL
11	Science and Technology	S&T	NL
12	Centro Ricerche FIAT	CRF	IT

For the CERBERO Consortium, please see the http://cerbero-h2020.eu web-site.

Document Authors

The following list of authors reflects the major contribution to the writing of the document.

Name(s)	**Organization Acronym**
Leszek Kaliciak	AS
Stuart Watt	AS
Ayse Goker	AS
Hans Myrhaug	AS

Document Revision History

Date	**Ver.**	**Contributor (Beneficiary)**	**Summary of main changes**
21-June-18	0.1	AS	1st Draft
23-June-18	0.2	AS	2nd Draft
25-June-18	0.3	AS	3rd Draft
28-June-18	0.4	AS	4th Draft
29-June-18	0.5	IBM, UNISS	Review, 5th draft
03-July-18	0.6	AS	6th Draft
03-July-18	0.7	UNISS	Review, 7th draft
03-July-18	0.8	AS	8th Draft
22-July-18	0.9	AS	9th Draft
31-July-18	1.0	AS	Version for submission

Table of contents

1. Executive Summary 5
1.1. Structure of Document 5
1.2. Related Documents 5
2. Scope and purpose 6
3. Description of the Ocean Monitoring demonstrator 8
3.1. Functionalities 12
3.1.1. Simulation of the OM components 12
3.1.2. Initial version of the physical prototype of the camera system 13
3.1.3. Image enhancement methods 15
3.1.4. Data fusion techniques for image enhancement, image retrieval, and navigation 17
3.1.5. Adaptation approaches 18
3.1.6. Information storage system 18
3.2. Tools 20
3.3. Development and deployment environment 20
4. Tests, Results and Feedback 23
4.1. Tests 23
4.2. Results 26
4.3. Feedback 27
5. Conclusion 29
6. References 30

1. Executive Summary

This document describes the M18 Ocean Monitoring demonstrator. The document is a preliminary version of D6.3, due by M36, which will describe the final Ocean Monitoring demonstrator as one of the CERBERO use-case scenarios.

1.1. Structure of Document

In Section 2 the scope and purpose of the demonstrator are described. Section 3 includes the description of the developed demonstrator. In Section 4 the tests, results and feedback are presented. Section 5 concludes and Section 6 provides the references.

1.2. Related Documents

- D2.4 – Description of Scenarios (Ver. 2)
  Demonstration follows the use case scenario description defined in D2.4.
- D2.7 – Technical Requirements (Ver. 2)
  Demonstration validates technical requirements defined in D2.7.
- D5.6 – Framework Components (Ver. I)
  Adopted components of the CERBERO framework are described in D5.6.
- D6.7 – Demonstration Skeleton (Ver 1)
  Skeletons description guides the implementation in demonstration activities.

2. Scope and purpose

This document describes the Ocean Monitoring M18 demonstrator, developed to demonstrate and validate the CERBERO technologies and tools. The M18 demonstrator provides the first iteration of the planned demonstrator, with the focus on the development of adaptive camera prototype and the use of a part of CERBERO technologies.

In the next iteration (M36) the functionality of the OM demonstrator will be further extended.

The drivers of demonstration activities have been defined in D2.7. In Table 2-1 an excerpt of Table 2 of D2.7 is provided.

Table 2-1 – OM User to Technical requirements mapping with related assessment strategy (M18 activities are shown in Italic)

User requirement	Technical requirement	Validation demonstration	Planned month
OM1. Provide complete design cycle from system level design to HW/SW co-design and implementation of Ocean Monitoring robot using adaptable COTS. **Need**: reduction of energy consumption and costs, increase reuse in other projects, while keeping or improving safety and security level and maintenance costs.	5, 6 (see section 4.5 and Table 4 in section 4.6 in D2.7)	*Development of Adaptive Camera based on COTS (Commercial Off The Shelf) HW with OEM firmware.* *Data storage according to mission needs* *On demand task dependent Data Fusion.* Secure communication. Building Battery and Motor functionality from generic components.	M18 initial prototype M18 first version of data storage system M18 data fusion models for image fusion and retrieval
OM2. Develop integrated “open” toolchain environment for development of Ocean Monitoring robots with focus on incremental prototyping. **Need**: facilitate development cycles, reduce time to market, and increase reuse, quality and verification level by incremental prototyping from high level of abstraction directly to working	1, 2, 3, 7, and 8 (see section 4.5 and Table 4 in section 4.6 in D2.7)	*Incremental prototyping of Adaptive Camera components from high level models.*	M18 configuration assessment based on high level models using CERBERO technology

real time applications.			
OM3. Development of a (self-)adaptation methodology with supporting tools. **Need**: Efficient support of functional adaptivity, according to system, human and environment triggers.	6 and 20 (see section 4.5 and Table 4 in section 4.6 in D2.7)	*Develop adaptive image enhancement methods for Adaptive Camera.* Multi-objective navigation and motor control modules with run time adaptation for Autopilot.	M18 adaptive image enhancement methods

In Table 4 of D2.7 the user requirements are mapped to technical requirements. In this document we are describing how the validation will be performed and when. The various elements and functionalities of the demonstrator are distributed between M18 and M36.

3. Description of the Ocean Monitoring demonstrator

The high level CERBERO requirements related directly to the Ocean Monitoring use case are the following (see D2.7):

1. (OM1) Provide complete design cycle from system level design to HW/SW co-design and implementation of Ocean Monitoring robot using adaptable COTS.
2. (OM2) Develop integrated “open” toolchain environment for development of Ocean Monitoring robots with focus on incremental prototyping.
3. (OM3) Development of a (self-) adaptation methodology with supporting tools.

Derived from these requirements are the three main goals which we want to accomplish during the development of the M18 demonstrator (see D2.4):

1. Development of adaptive camera system and development of new task-dependent information fusion techniques. Development of efficient information fusion models for information storage and retrieval purposes.
2. Incremental prototyping of OM components from high level models. Use of CERBERO infrastructure to reduce the cost of the assessment of different design time and runtime configurations.
3. Development of adaptive image enhancement methods.

The first goal is accomplished by the development of the adaptive camera system initial physical prototype comprising two high-definition cameras. The camera is capable of adapting to different visibility conditions and user preferences. The developed and implemented data fusion techniques and data storage system enable adaptive image enhancement methods and efficient storage and retrieval of relevant information based on combined data.

The second goal is the development of initial high-level models of the OM components using the modelling support of the CERBERO infrastructure to assess alternative design configurations and to ensure that the final hardware platform will meet the KPIs required in its final runtime configuration.

The third goal relates to the development of image enhancement techniques for the adaptive camera system.

Figure 3-1 shows the main components of the M18 demonstrator consisting of:

- Adaptive Camera System which can be also used as navigational sensors - images of different quality depending on the number of active cameras or depth maps for distance measurement for obstacle avoidance.
- Information storage system for storing and retrieving images, videos, and other data obtained from sensors.
- Data fusion models, from WP4, for image enhancement and retrieval.

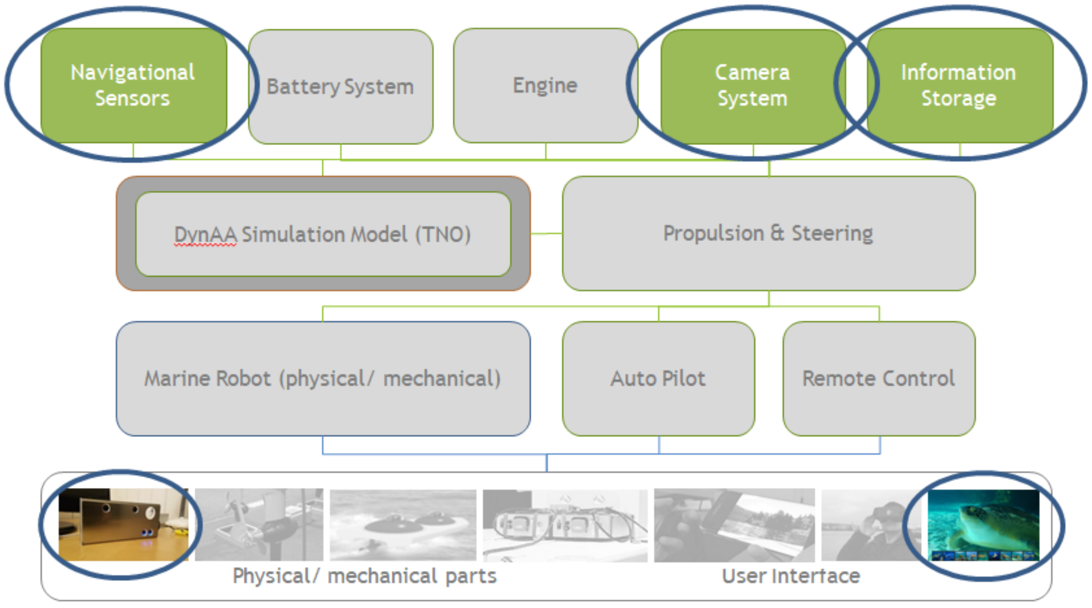


Figure 3-1. Highlighted components of the M18 OM demonstrator.

Components of M18 demonstrator are shown in colour in Figure 3-1.

CERBERO demonstrators, for all the use-cases, follow the generic CERBERO skeleton defined in D6.7. Therefore, it is important to be able to identify the skeleton parts in the demonstrator overview presented in Figure 3-1, which could help in future reuse of the developed CERBERO tools. The graphical mapping is provided in Figure 3-2.

Here is a short recap of the different element for M18 implementation:

1. KPI Estimator: represents a way to evaluate how close the CPS currently is to its expected goals/performance.
2. Manager: defines if adaptation is needed, according to the distance between the evaluated KPIs and the final system goal.
3. Target: composed of three different elements: the adaptation engine, the fabric and the monitors. The engine is a target dependent element which physically put in place the actions to execute the decisions of the manager in terms of adaptivity, the reconfigurable fabric carries out the functional tasks according to the taken decisions, and the HW monitors and the SW supervisors sense the fabric status.

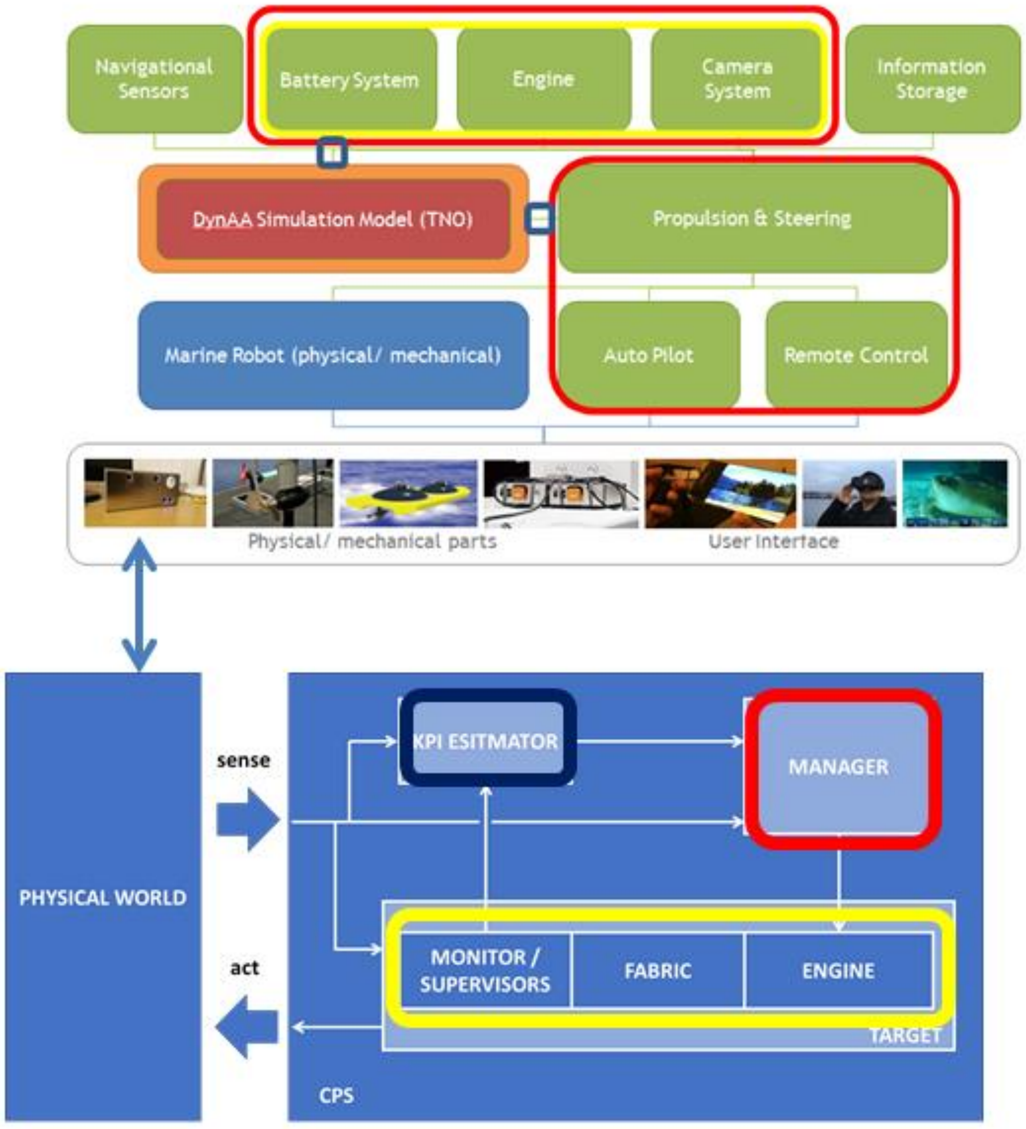


Figure 3-2. Ocean Monitoring use-case - CERBERO skeleton mapping. Notice the colour-based mapping between the two diagrams.

The M18 OM demonstrator follows the CERBERO adaptation loop (Figure 3-3).

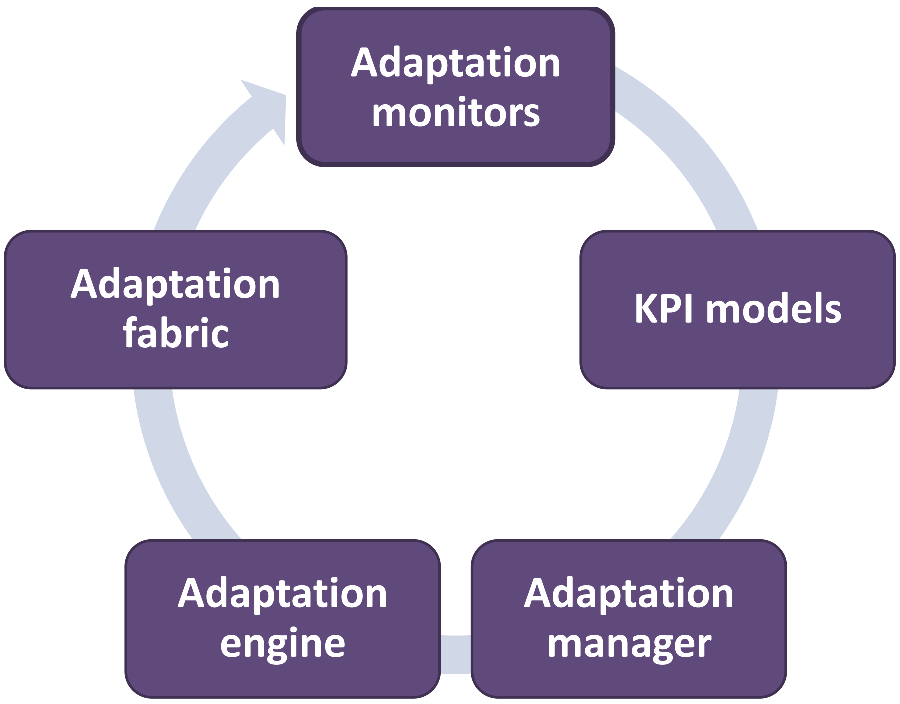


Figure 3-3. CERBERO Adaptation Loop.

The M18 demonstrator is an example of user commanded, environment triggered, and context aware adaptation. Context describes an aspect of a real world situation. One type of adaptation is user commanded resulting in change of functionality, e.g. distance measurement vs image de-noising. Another type of adaptation is an automatic adaptation to changing environmental lighting conditions resulting in different levels of image enhancement and illumination correction.

The M18 demonstrator aspects and the adaptation loop components can be mapped as follows:

Adaptation monitors:

- Camera sensors monitoring the environment.
- Illumination sensor.

KPI models:

- User perceived image quality, ranked feature.

- Image quality measured automatically by illumination sensor and estimated from image histogram.
- Low prototype cost by using low cost components.
- Appropriate for real-time response time.

Adaptation manager:

- Uses different types of image fusion to enable context aware adaptivity.
- Environment triggered adaptation based on lighting measurements of illumination sensor and estimated from histogram. The adaptation is continuous, lighting measurements scaled to [0, 1], linear weighted combination of original and enhanced image with weights w, 1-w; w is related to lighting numerical value from [0, 1].

Adaptation engine:

- Changes the image fusion model e.g. distance measurement vs image de-noising.
- Changes the level of image enhancement/lighting correction.

Adaptation fabric: changed functionality and enhanced video adapted to the current situation/context.

3.1. Functionalities

The implementation of the M18 Ocean Monitoring demonstrator functionalities (according to the goals and requirements of D2.7) is divided into the following parts:

1. Simulation of the OM components for different configurations assessment.
2. Initial version of physical prototype of the Camera System
3. Image enhancement methods
4. Data fusion techniques for image enhancement, image retrieval, and navigation
5. Adaptation approaches
6. Information storage system

In the following paragraphs the developed functionalities of different parts of the demonstrator are described in more detail.

3.1.1. Simulation of the OM components

The developed DynAA simulation models are used to assess alternative design configurations choices to ensure that the final hardware platform will meet the KPIs required in its final runtime configuration. The detailed description of the activity is presented in Section 3.3.

This assessment was necessary because there are significant competing constraints in the final platform, including:

- Video processing throughput
- Java performance
- Battery performance
- Storage performance

The primary approach to modelling was to define a new kind of DynAA 'Node', which included the camera sensor and a video processing pipeline to model image processing and data fusion.

The overall models represent a subset of the processing requirements designed as depicted in Figure 3-4.

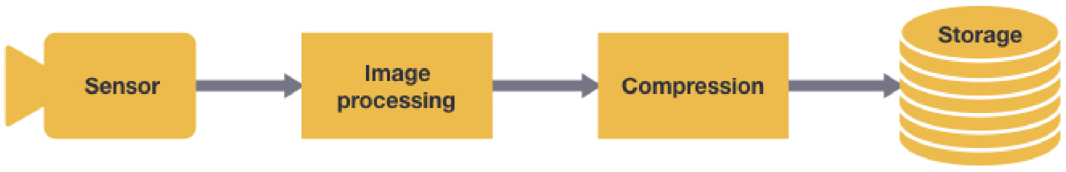


Figure 3-4. Model flow for design-time assessments.

3.1.2. Initial version of the physical prototype of the camera system

The initial physical prototype of the Camera System comprises two cameras working in tandem is shown in Figure 3-5. The algorithms for the cameras working in tandem require stereo calibration and rectification of both cameras. The calibration process finds the intrinsic and extrinsic camera parameters and removes the radial distortion from the images (Figure 3-6). Rectification of the cameras re-projects image planes onto a common plane parallel to the line between optical centres (Figure 3-7).

The use of two cameras in the M18 demonstrator has three different applications: 1) adaptive incremental video quality levels depending on the number of cameras active, 2) depth maps calculation, and 3) stereoscopic images.

All three applications of the camera system can be used for, e.g., enhanced navigational capabilities – enhanced remote control or computer vision (stereo images in virtual reality headset, super-resolution imaging) and obstacle avoidance (disparity/depth maps).

The prototype Camera System incorporates different adaptation approaches: user-triggered adaptation, adaptation to changing visibility conditions, and adaptation to internal state.

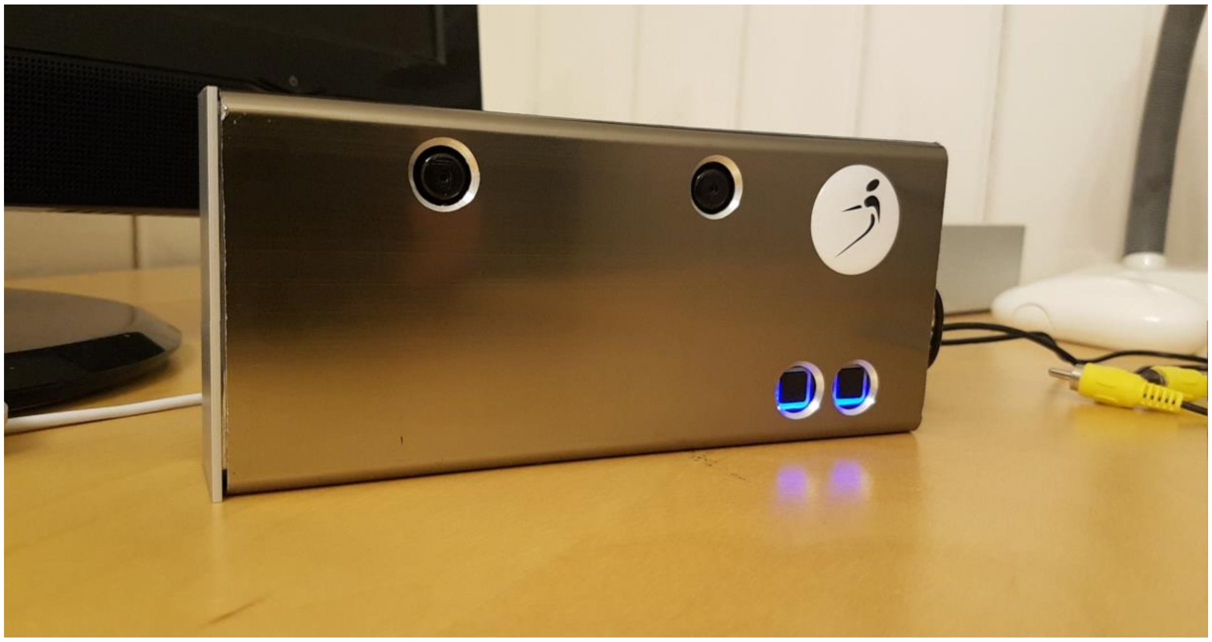


Figure 3-5. First Prototype of the Adaptive Camera System.

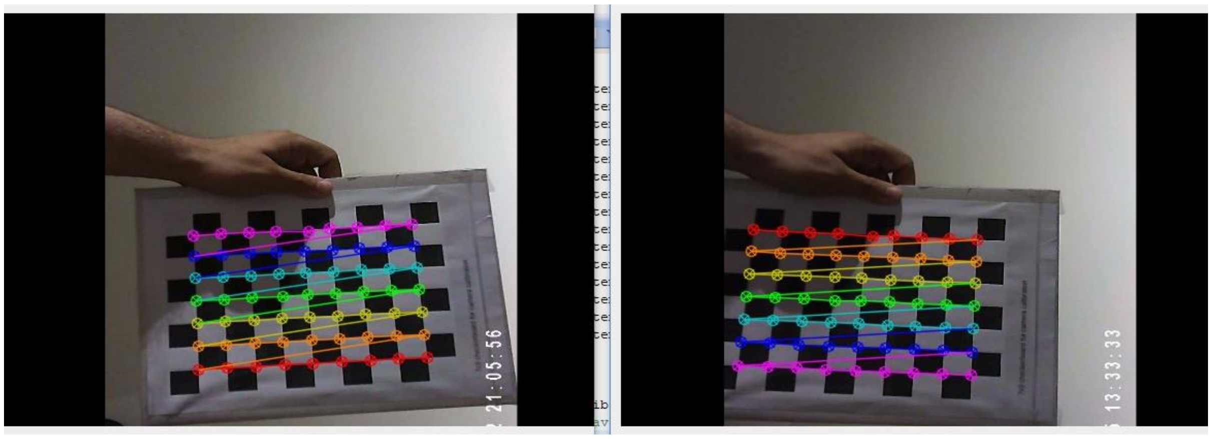


Figure 3-6. Stereo calibration.

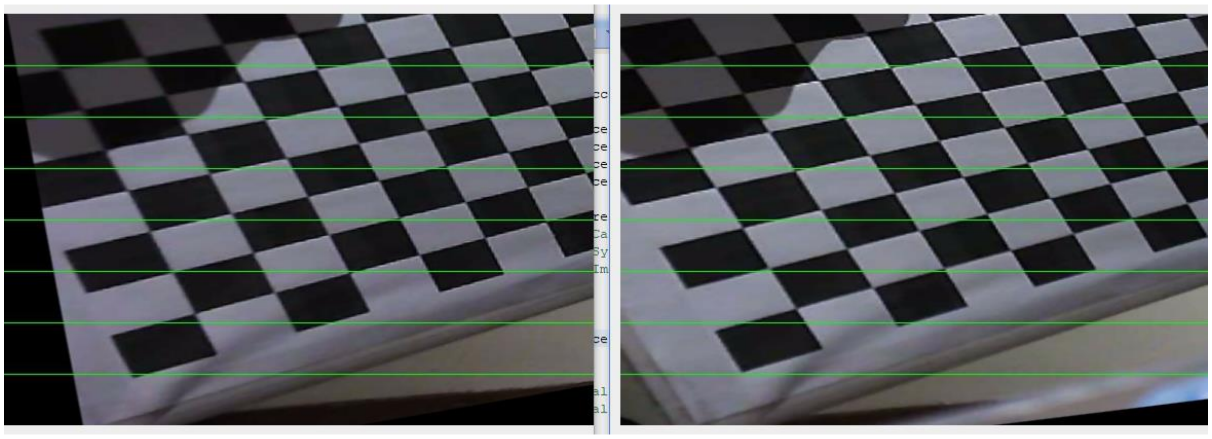


Figure 3-7. Stereo rectification.

3.1.3. *Image enhancement methods*

The Adaptive Camera System uses the following image enhancement methods: fusion of our edge detector with the original image, histogram equalization, and image de-nosing.

Figure 3-8 shows the results of histogram equalization.

The results of fusion of the original with the novel edge image are depicted in Figure 3-9. The fusion model de-noises the image and enhances its edges.

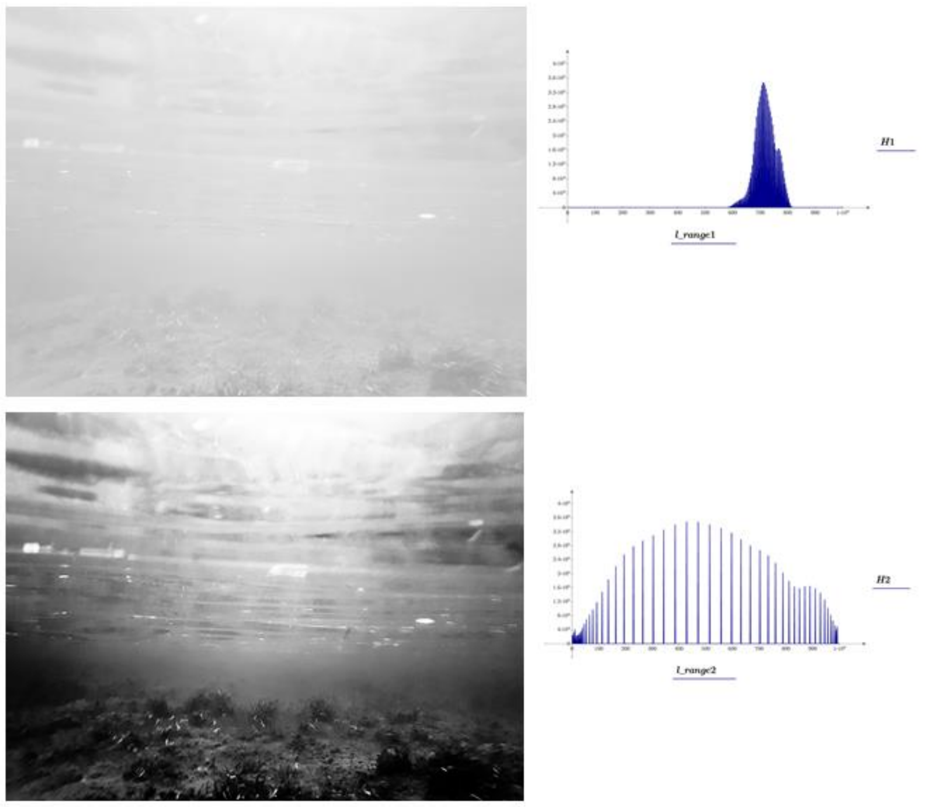


Figure 3-8. Underwater images and their corresponding histograms before and after histogram equalization.

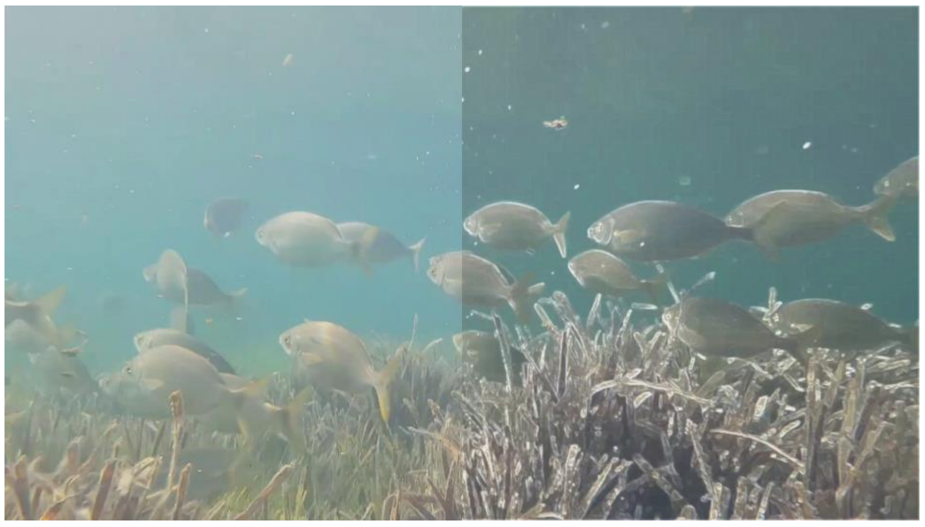


Figure 3-9. Image enhancement by information fusion. Left - original image, right - enhanced image.

3.1.4. Data fusion techniques for image enhancement, image retrieval, and navigation

The information fusion techniques used in the demonstrator have been developed as part of the T4.3 activities from WP4. The OM demonstrator uses different information fusion models in order to:

- *Enhance images*: the edge detector image is fused with the original one to sharpen the image, remove the noise, and enhance the edges of objects.
- *Enhance images*: images from different cameras are fused together in order to improve the overall image quality (super-resolution).
- *Retrieve fused data*: stored data (e.g. images) need to be retrievable based on combination of different information related to them. The proposed unified information storage and retrieval framework uses different data fusion models to perform the information search.
- *Enhance navigation*: fused images from the cameras are used to find the disparity/depth maps that can be used for obstacle avoidance.

Figure 3-10 shows different notions of correlation captured in the data fusion model in the context of information retrieval. The developed unified information retrieval model represents a complete information retrieval solution and uses tensors and the notion of co-occurrence to combine the so-called feature spaces e.g. visual and textual.

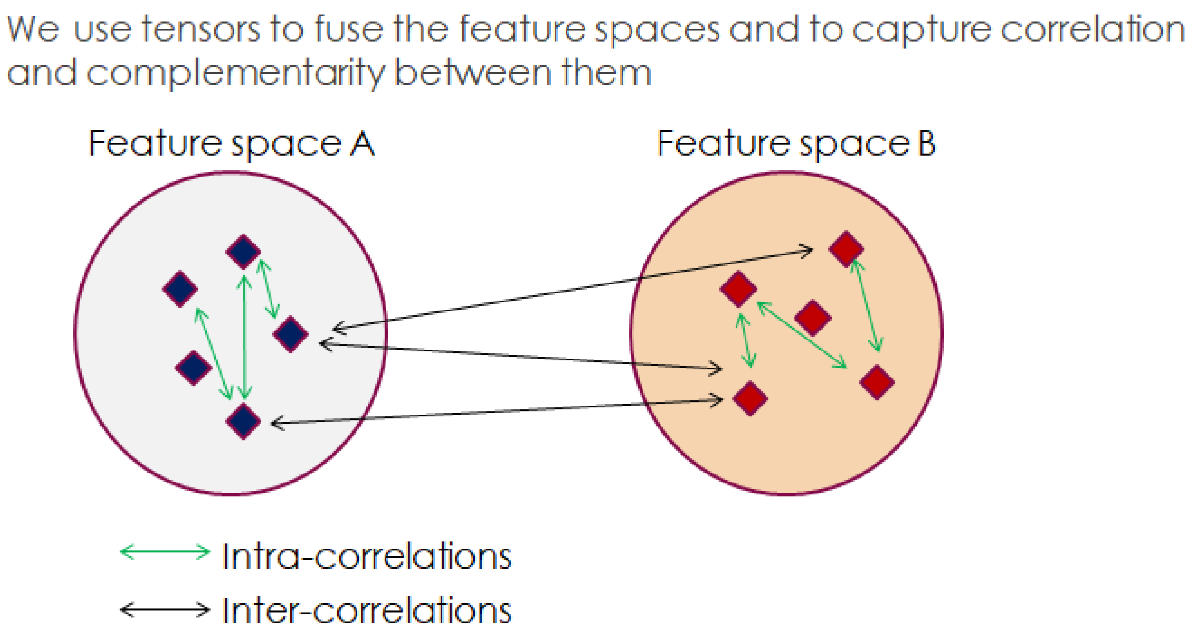

We use tensors to fuse the feature spaces and to capture correlation and complementarity between them
Feature space A
Feature space B
Intra-correlations
Inter-correlations

Figure 3-10. Notions of correlation in the context of information retrieval.

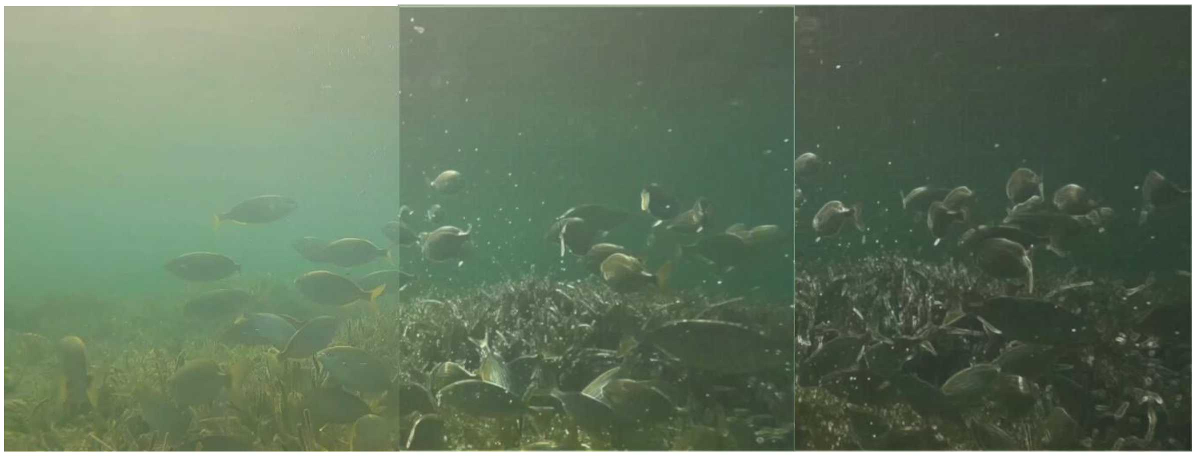


Figure 3-11. Image Enhancement based on Lighting Conditions.

3.1.5. Adaptation approaches

The M18 demonstrator uses the following adaptation approaches:

- Adaptation to user preferences: the user can trigger the adaptation of the camera system by requesting a different functionality which requires different image fusion model.
- Adaptation to visibility conditions based on the measurements from the illumination sensor. The fusion model for image enhancement adaptively changes the weights associated with the influence of the edge detected image and the original one. The poorer the visibility conditions, the higher the level of image enhancement

Figure 3-11 presents different levels of image enhancement. The levels of image enhancement will correspond to camera’s self-adaptation to different lighting conditions.

3.1.6. Information storage system

The M18 demonstrator stores the information such as multimedia in a specific way that allows for efficient retrieval of relevant data. For example, apart from the images themselves, an index is also created and stored. The index stores the data associated with the multimedia objects in a vector form. What is interesting is that the index for the textual data as well as the index for images is represented in a uniform way. This allows for efficient information retrieval based on similar principles.Figure 3-12 shows the top results of image search by visual example. Figure 3-13 presents the storage format for textual and image indexes.

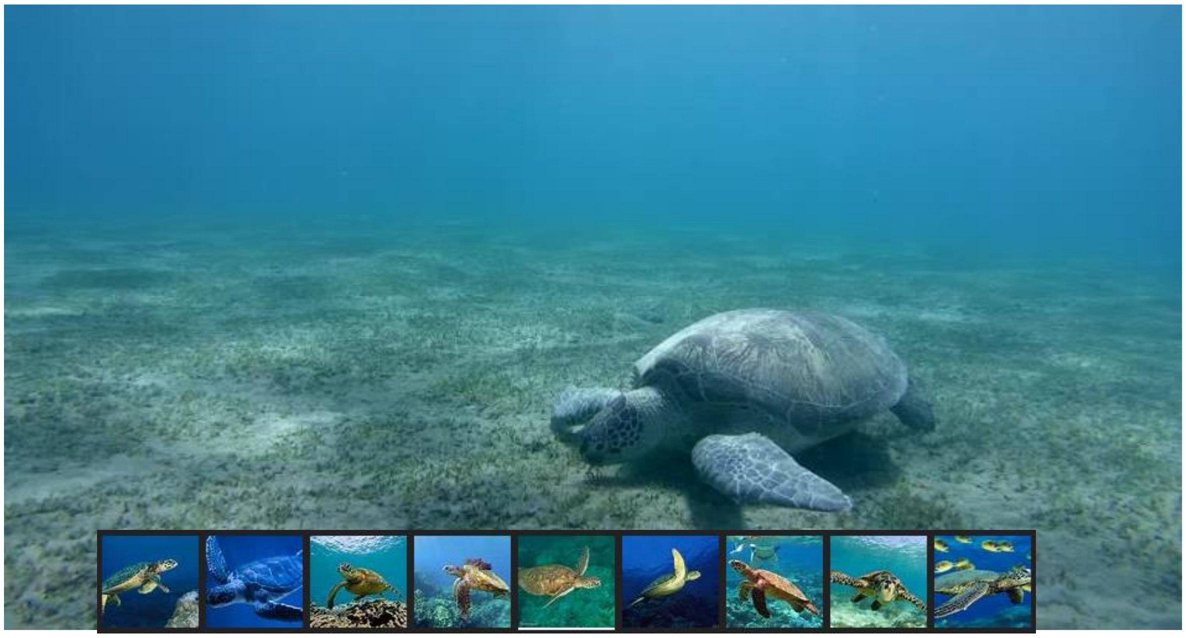
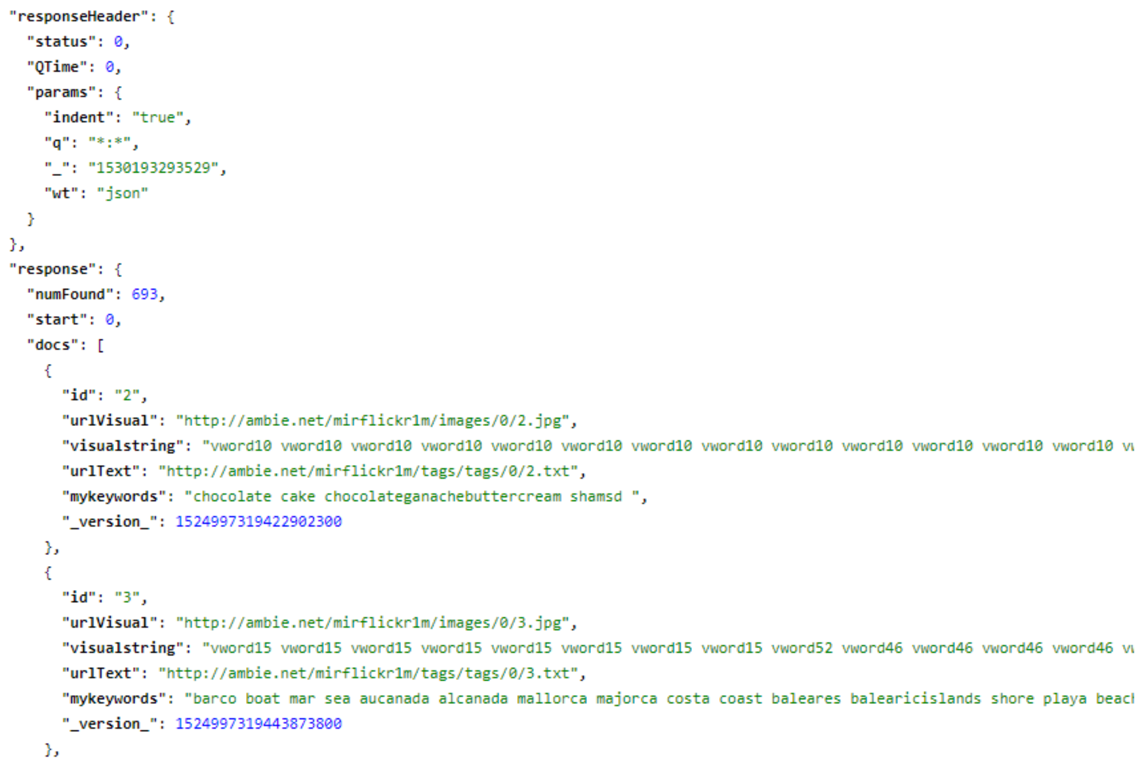


Figure 3-12. Content-based image retrieval.



```
"responseHeader": {
  "status": 0,
  "QTime": 0,
  "params": {
    "indent": "true",
    "q": "*:*",
    "_": "1530193293529",
    "wt": "json"
  }
},
"response": {
  "numFound": 693,
  "start": 0,
  "docs": [
    {
      "id": "2",
      "urlVisual": "http://ambie.net/mirflickr1m/images/0/2.jpg",
      "visualstring": "vword10 vword10 vword10 vword10 vword10 vword10 vword10 vword10 vword10 vword10 vword10 vword10 vword10 vw
      "urlText": "http://ambie.net/mirflickr1m/tags/tags/0/2.txt",
      "mykeywords": "chocolate cake chocolateganachebuttercream shamsd ",
      "_version_": 1524997319422902300
    },
    {
      "id": "3",
      "urlVisual": "http://ambie.net/mirflickr1m/images/0/3.jpg",
      "visualstring": "vword15 vword15 vword15 vword15 vword15 vword15 vword15 vword15 vword52 vword46 vword46 vword46 vword46 vw
      "urlText": "http://ambie.net/mirflickr1m/tags/tags/0/3.txt",
      "mykeywords": "barco boat mar sea aucanada alcanada mallorca majorca costa coast baleares balearicislands shore playa beac
      "_version_": 1524997319443873800
    },
```

Figure 3-13. Storage system of textual and visual information.

3.2. Tools

The tools used in the development of the M18 demonstrator are mainly Java-based development environments, and the CERBERO DynAA tool (see D5.6 and the following Section 3.3 for details).

The demonstrator also uses SOLR which is a powerful, scalable and fault-tolerant search engine based on the Apache Lucene. SOLR is written in Java, provides distributed search and index replication, full text search, hit highlighting, faceted search, real-time indexing, dynamic clustering, database integration, NoSQL features, and rich document handling. We have developed a method to also store visual features in SOLR so that it can be used to retrieve images based on visual content.

3.3. Development and deployment environment

A key part of the development process has involved using DynAA to assess alternative design configurations to ensure that the final hardware platform will meet the KPIs required in its final runtime configuration. At M18, we assessed three alternative design configurations, as set out in D2.4. Briefly, these were:

- Snapdragon 835 / 845 (Android, ARM-based, 8 cores of CPU, Adreno GPU, USB C, USB 3)
- Intel i7 Kaby Lake (Intel x64-based, 8 cores of CPU, Intel GPU, USB 3)
- NVidia Jetson TX1 (ARM-based, 4 cores of CPU, NVidia Maxwell GPU, USB 3)

This assessment was necessary because there are significant competing constraints in the final platform, including:

- Video processing throughput (memory speed, GPU, video compression performance, camera interface bandwidth).
- Java performance (on Intel x64 and ARM, and the Android-based Snapdragon uses a different VM than the standard Java used by the Intel and NVidia platforms).
- Battery performance (platforms use different voltages and have varying power needs).
- Storage performance (generally less of an issue, as processed data is very much smaller than the raw sensor data, therefore we don’t consider it further here).

Although all three candidate platforms can deploy the necessary logic, it was not clear which, if any, of these alternatives was best suited for the final implementation – and whether any architectural extensions or additional components would be required to implement the processing load required by the image enhancement and processing load needed by the image enhancement (see 3.1.3) and data fusion (see 3.1.4) components described above. Comments about the comparison are presented below.

To reduce the cost of this assessment, we used DynAA at design time to model – roughly – each of the three different platforms, and measure their ability to handle the processing loads required. Originally we also considered using AOW to optimize, but we found that our initial configuration design is generally not an optimization problem. Instead, it is about whether solutions are viable or not, rather than how to make a better solution. This is particularly true when cost is a priority, as the space of alternative configurations is very highly constrained by the available platforms.

During this process, we also evaluated and explored the challenges of using DynAA at runtime to manage adaptation. On the whole, it performed well, and seemed generally better suited to runtime adaptation against physical environments than it did against the complexities of stream processing on modern architectures, such as GPUs and FPGAs, which are substantially different to conventional computing architectures.[1]

The primary approach to modelling was to define a new kind of DynAA node, which included the camera sensor and a video processing pipeline to model basic image processing and data fusion, and then to assess whether, when run, these models would backlog with video data. We also modelled power requirements, but very naively, as all three platforms were within viable power constraints.

The model implementations were integrated using a standard Maven build against DynAA version 1.2.0-SNAPSHOT, commit 884809ef130 on the master branch at: git@ci.tno.nl:DynAA/DynAA.git.

To adapt DynAA to this modelling task, we implemented a number of new model classes:

- *VideoProcessor*, and a subclass *CompressionProcessor* – these roughly parallel the built-in *Processor* class, but don't use integer or floating-point operations per second. Instead, they use kilobytes per second, and model tasks in terms of their transformation of quantities of data into other quantities of data. This is more consistent with the patterns of stream processing. By and large, the algorithms' performances can be estimated naively. For example, for compression, standard implementations of H.264 (and H.265) have a performance that is approximately linear by input bitrate. It is typical for there to be several *VideoProcessors* connected together, especially when there is dedicated hardware support for video compression, for example.[2] Note that as with the standard DynAA components, we did not attempt to model many subtleties, e.g., data fusion, processors with multiple cores, GPU internals, and so on.
- *VideoProcessingSegment* – a parallel version of *ProcessingSegment*, designed to work with *VideoProcessor* and its subclasses, again defining processing in terms of kbps rather than CPU-style operations.
- *CameraSensor* – a subclass of *Sensor* that produces a given amount of data per second. The three platforms allow cameras to be connected in different ways: two (NVIDIA and Snapdragon have direct CSI-2 connectors, where Kaby Lake uses USB) and they differ in the number of sensor processors. All can handle two

[1] This should probably have been expected. By its very nature, stream processing is intrinsically adaptive and less sequential. The DynAA model framework would benefit from further work building components and unit types to ease this process.

[2] The limit of this approach is when using a GPU. These typically have a fairly large number of cores (in the case of the Nvidia Jetson, it's 256 cores, but it may be up to ten times that number). When several algorithms are being used for different parts of an image processing pipeline, the load on a GPU is not easy to model due to very significant impacts of multiple levels of cache memory, GPU memory size, and so on. This is not well documented, as it typically involves proprietary technology internally.

cameras, the NVIDIA can theoretically handle up to six. They vary, therefore, in the amount of camera sensor data generated per second.

- *Storage* – essentially identical to the model basic *Memory*, but separated to eventually permit power and data throughput limits to be represented. This will gradually extend into a model of the Apache SOLR components described earlier in section 3.2.

Using these components, the design time assessment of the different architectures was completed by implementing three distinct parametric models, -using data derived from the specification sheets (reducing the CPU and GPU specifications by 25-30% to allow for system overheads) and then modelling a common video processing load for a camera sensor component over a simulated period of five minutes. If the process load completed within the period specified, that indicated that the given architecture could handle the load expected. This was a pass/fail assessment of modelled performance. The results can be found in Section 4.

The modelling approach was as follows. We identified a common pattern in line with Figure 3-3, above, and implemented an abstract Java test class which allowed performance parameters to be injected by specific instances. This abstract test class implemented: (a) a camera sensor, (b) a CPU, (c) a separate video processing unit, writing to (d) storage. Then, for each hardware platform we implemented a subclass which set component parameters (like compression throughput capacities) derived from public specifications and benchmarking results. We also added a number of DynAA components that allowed timing and battery measures at different points in the process, using additional Java logging to include them in the test output (DynAA uses this extensively). We finally used scripts to translate the model log output files into CSV files for import into a spreadsheet and charting.

4. Tests, Results and Feedback

4.1. Tests

The M18 Ocean Monitoring demonstrator tests the following OM system functionalities:

- Simulation of the OM components
- Initial physical prototype of the Camera System
- Image enhancement methods
- Data fusion techniques for image enhancement, image retrieval, and navigation
- Adaptation approaches
- Information storage system

The initial simulation of the OM components assessed alternative design configuration choices to ensure that the final hardware platform will meet the required KPIs. After running assessments of all three different platforms - Snapdragon 835 / 845, Intel i7 Kaby Lake, and NVidia Jetson TX1, the differences between them were not material, as shown in Table 4-1. According to the models, all three were capable of processing the expected video load with processing capacity to spare. Note that, particularly for GPUs, there may be very substantial adaptation logic built in beyond what could be modelled, and the modelling of algorithm processing load was relatively naïve, so the precision of these assessments is not perfect. However, the overhead tolerances provide enough room for confidence that all architectures are viable.

Accordingly, the Snapdragon architecture was selected, as it requires substantially less power than the alternatives (see Table 4-2), and provides additional sensing capabilities that reduce the need for additional component integration.

The simulation can later be extended by addition of other components. In future, the simulation will also test the adaptive capabilities of components such as the camera system, by progressively adding more camera sensors to improve the image quality, for example.

The initial version of the physical camera system validates the stereo-calibration and stereo-rectification process needed to accomplish the other tasks, such as image de-noising, stereoscopic images, and depth maps calculation. The validation took the form of visual inspection of images and how well the images could be aligned after calibration and rectification.

The information fusion approaches such as the adaptive relevance feedback model for the combination of features in the context of relevance feedback are tested on publicly available data collections – ImageCLEF and MIRFlickr. Information storage system demonstration component validates the system's capability to efficiently store both textual and visual information in a uniform format. We were able to utilise SOLR's efficient text retrieval capability to also search based on visual content of image objects.

The image enhancement methods and adaptivity will be initially tested by comparing the number of keypoints detected before and after the adaptive image enhancement.

Table 4-1 – DynAA modelled power usage for different configurations.

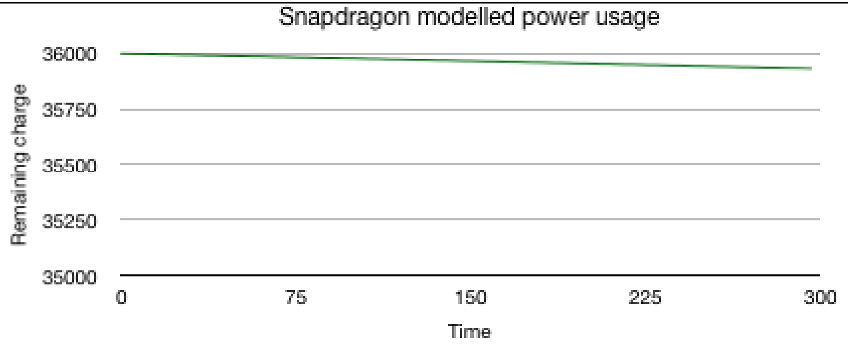
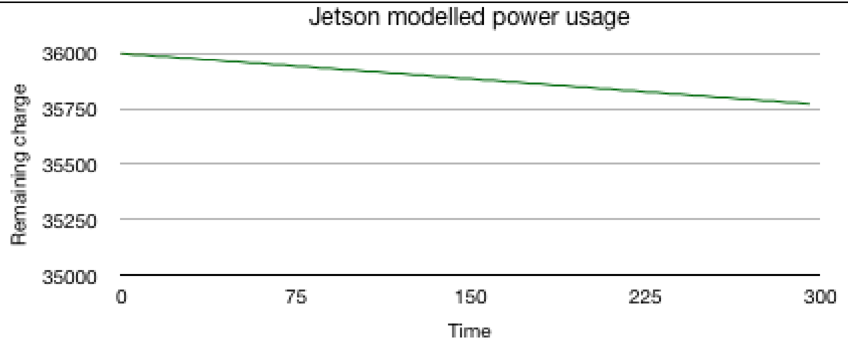
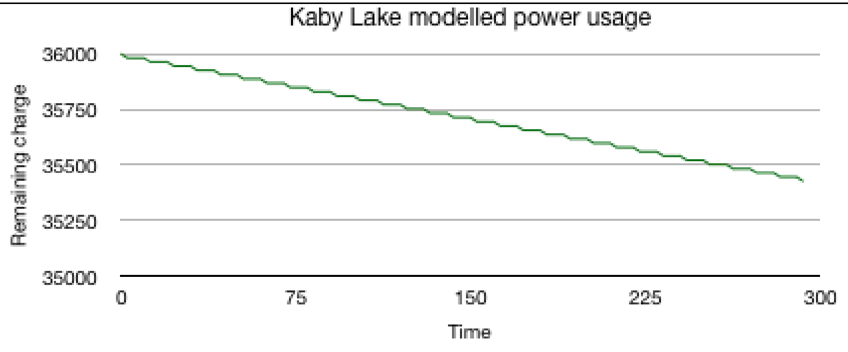
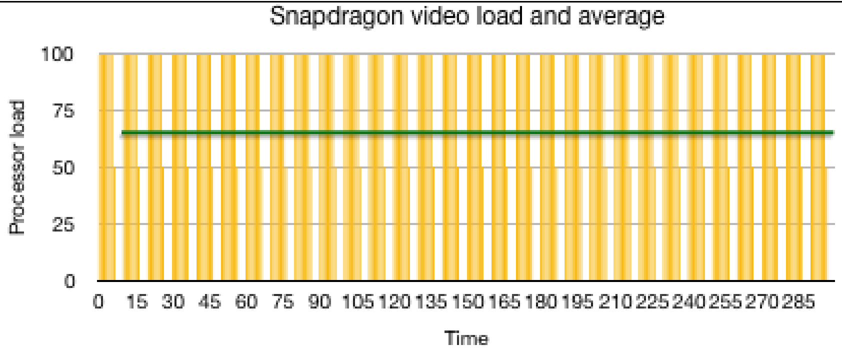
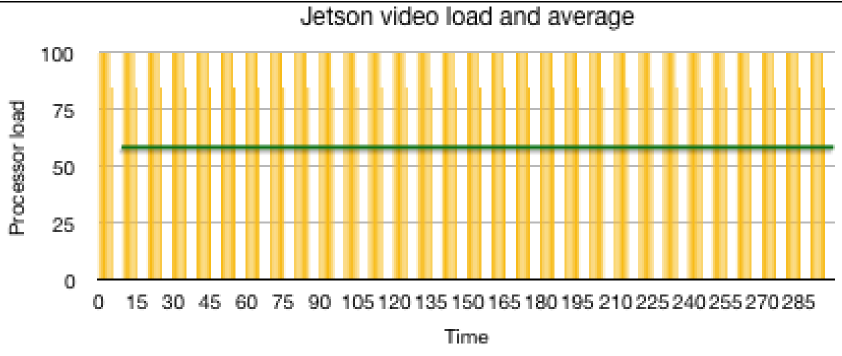
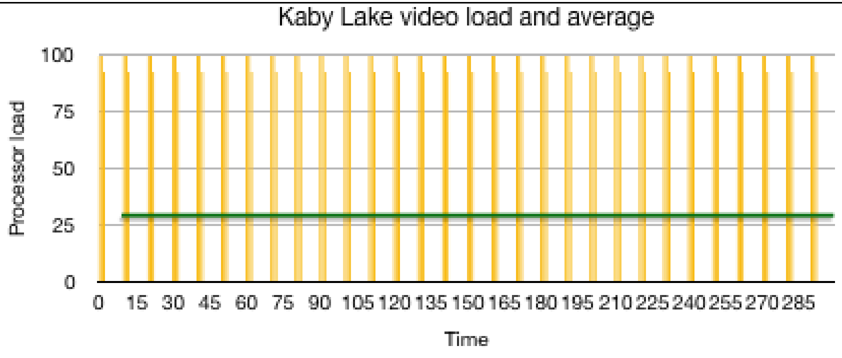
Platform results	Notes
	Based on 1 x 4k camera, video processed in 10 second blocks. Five minutes of video requires around 70 coulombs, or 20 mAh.
	Based on 1 x 4k camera, video processed in 10 second blocks. Five minutes of video requires around 230 coulombs, or 65 mAh.
	Based on 1 x 4k camera, video processed in 10 second blocks. Five minutes of video requires around 575 coulombs, or 160 mAh.

Table 4-2 – DynAA model results for video processor load task.

Platform results	Notes
	Based on 1 x 4k camera, video processed in 10 second blocks. Moving average video load is 65%.
	Based on 1 x 4k camera, video processed in 10 second blocks. Moving average video load is 58%.
	Based on 1 x 4k camera, video processed in 10 second blocks. Moving average video load is 29%.

The simulation can later be extended by addition of other components. In future, the simulation will also test the adaptive capabilities of components such as the camera system, by progressively adding more camera sensors to improve the image quality, for example.

The initial version of the physical camera system validates the stereo-calibration and stereo-rectification process needed to accomplish the other tasks, such as image de-noising, stereoscopic images, and depth maps calculation. The validation took the form of visual inspection of images and how well the images could be aligned after calibration and rectification.

The information fusion approaches such as the adaptive relevance feedback model for the combination of features in the context of relevance feedback are tested on publicly available data collections – ImageCLEF [IMAGECLEF 2018] and MIRFlickr [MIRFLICKR 2018].Information storage system demonstration component validates the system's capability to efficiently store both textual and visual information in a uniform format. We were able to utilise SOLR's efficient text retrieval capability to also search based on visual content of image objects.

The image enhancement methods and adaptivity will be initially tested by comparing the number of keypoints detected before and after the adaptive image enhancement.

4.2. Results

In the Table 4-3 below, in accordance with D6.7, the results from the demonstrator development activity are shown.

Table 4-3 – M18 demonstration results.

ID	Goal	M18 demonstrator results
OM1	OM1. Provide complete design cycle from system level design to HW/SW co-design and implementation of Ocean Monitoring robot using adaptable COTS.	Reduction of costs, increase of reuse in different simulation scenarios. *Development of the Data Storage System for efficient storage and retrieval of different type of information* *Development of data fusion models for information retrieval based on combination of features, and for image enhancement purposes* *Development of the initial version of the physical prototype of Adaptive Camera System. Multiple cheap cameras produce images of the quality of more expensive cameras, thus reducing the cost* *The Camera System, Data Storage System, and the Data Fusion Models can be reused across all the use-cases*
OM2	OM2. Develop integrated "open" toolchain environment for development of Ocean Monitoring robots with focus on incremental prototyping.	Reduce time of development, verification, integration, along with the related costs, exploiting a library of reusable components/metrics integrated by common framework in different levels of abstraction. Incremental prototyping. *Incremental prototyping of the OM robot based on the DynAA simulation models and the KPIs such as video processing throughput, java performance, battery performance, and storage performance* *Components such as the camera system can also be reused in different context,*

		outside of the OM use-case
OM3	OM3. Development of a (self-)adaptation methodology with supporting tools.	Efficient support of functional adaptivity, according to system, human and environment triggers. *The adaptation of the camera system can be triggered by system, human, and environment. The adaptation will result in different number of activated cameras and different levels of image enhancement*

4.3. Feedback

While DynAA worked well for the configuration design task, we identified a number of challenges that are expected to cause challenges for runtime integration:

1. **The DynAA primary engine uses a singleton pattern.** This means, essentially, that only one model can run in a JVM at any one time. Because the CERBERO model permits a hierarchical approach with multiple models, the current implementation requires that each model access involves clearing out the old model, adding a new one, initializing and running it, as a blocking operation. This is a substantial architectural limitation, as spreading the models into distinct JVMs is a far more complex way of assembling a system. Singletons are often considered an anti-pattern for this exact reason, and addressing this should be a priority in the next stage of development activity.
2. **Missing unit types for computation and data quantities.** DynAA uses the JSR 363 units of measurement API extensively. For physical quantities, such as luminance, wavelengths, pressures, and battery capacities, this is perfect and a significant asset. Where it proved a challenge was modelling computable quantities, like kilobytes per second, frames per second, and storage quantities like megabytes. Even though these don't fit into standard SI units, they are not strictly dimensionless. Since a significant part of DynAA's role in the Ocean Monitoring case will be adaptation of the compute platform, supporting these data types would make DynAA much easier to use.
3. **Improved entity name display.** Generally, DynAA didn't reflect entity names in its logs, which made debugging harder. To overcome this, OM subclasses typically override the toString() method, to make entity names reflect component roles. We'd suggest that standard DynAA classes should also do this by default.
4. **Exception handling.** DynAA uses a wide range of Exception subclasses, all of which derive directly from the Java Exception class. This made exception handling more challenging, as there was no way to easily catch all DynAA exceptions distinct from other Java exceptions. We suggest introducing new intermediate classes for DynAA exceptions – ideally, one for model construction time, and a second for model run time, e.g., ModelException and ModelRuntimeException.

5. **Versioning.** We used a snapshot building of DynAA (1.2.2-SNAPSHOT) as there weren't any other version tags. This makes the models vulnerable to API changes as DynAA develops. Tagging versions, using (for example), maven-release-plugin, would make it easier to decouple DynAA development from projects downstream.
6. **Documentation.** DynAA's examples and unit tests were generally good, but more would be helpful, especially when it comes to extending the unit type use associated with JSR 363.

Of these, only points 1 and 4 are dependent on upstream changes to DynAA, and only point 1 involves significant effort.

5. Conclusion

The M18 demonstrator addresses some of the key aspects of the OM system: re-usability, reduction of cost, incremental prototyping. It required development of the initial simulation model of different OM components to asses alternative design configuration choices, development of early physical camera system, calibration and rectification of the camera system, development and implementation of the information fusion models for image enhancement and image retrieval, development of the data storage system that can store and efficiently retrieve textual and visual type of information, and the camera adaptation approach.

The physical camera component will be an important part of the marine robot that will be used for navigation and collection of data purposes. The storage and retrieval system will allow us to create even more elaborate AI models that can re-use the collected data as a prior knowledge. Development of the initial simulation of the OM components helped to acquire the necessary skills to work with DynAA tool. The next steps will involve addition of more OM components to the model.

6. References

[CERBERO 2018]	http://www.cerbero-h2020.eu
[IMAGECLEF 2018]	https://www.imageclef.org
[MIRFLICKR 2018]	https://press.liacs.nl/mirflickr/
[D2.7]	CERBERO D2.7 Technical Requirements (Ver. II)
[D2.4]	CERBERO D2.4 Description of Scenarios (Ver. II)
[D3.4]	CERBERO D3.4 Modelling of KPI (Ver. I)
[D6.7]	CERBERO D6.7 Demonstration Skeleton (Ver 1)