Ref. Ares(2020)103486 - 08/01/2020

Information and Communication Technologies (ICT)

Programme

Project Nº: H2020-ICT-2016-1-732105

D6.3: Ocean Monitoring Demonstrator (Final Version)

Lead Beneficiary:	9 – AS
Work Package:	6
Date:	31-December-19
Distribution – Confidentiality:	[Public]

Abstract:

This document contains the description of the M36 Ocean Monitoring demonstrator. The description is based on the skeleton defined in D6.1 and includes the scope and purpose of the demonstrator, the description of the demonstrator itself and the accomplished results.

© 2019 CERBERO Consortium, All Rights Reserved.

Disclaimer

This document may contain material that is copyright of certain CERBERO beneficiaries, and may not be reproduced or copied without permission. All CERBERO consortium partners have agreed to the full publication of this document. The commercial use of any information contained in this document may require a license from the proprietor of that information.

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.

Except as otherwise expressly provided, the information in this document is provided by CERBERO to members “as is” without warranty of any kind, expressed, implied or statutory, including but not limited to any implied warranties of merchantability, fitness for a particular purpose and non-infringement of third party’s rights.

CERBERO shall not be liable for any direct, indirect, incidental, special or consequential damages of any kind or nature whatsoever (including, without limitation, any damages arising from loss of use or lost business, revenue, profits, data or goodwill) arising in connection with any infringement claims by third parties or the specification, whether in an action in contract, tort, strict liability, negligence, or any other theory, even if advised of the possibility of such damages.

The technology disclosed herein may be protected by one or more patents, copyrights, trademarks and/or trade secrets owned by or licensed to CERBERO Partners. The partners reserve all rights with respect to such technology and related materials. Any use of the protected technology and related material beyond the terms of the License without the prior written consent of CERBERO is prohibited.

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
Evgeny Shindin	IBM

The list of authors does not imply any claim of ownership on the Intellectual Properties described in this document. The authors and the publishers make no expressed or implied warranty of any kind and assume no responsibilities for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information contained in this document.

Document Revision History

Date	Ver.	Contributor (Beneficiary)	Summary of main changes
23-Oct-19	0.1	AS	1st Draft
15-Nov-19	0.2	AS	2nd Draft
12-Dec-19	0.3	AS	3rd Draft
16-Dec-19	1.0	TNO, UNISS	Internal reviews by Joost Adriaanse (TNO) and Francesca Palumbo (UNISS)
19-Dec-19	1.1	AS	Feedback incorporation
20-Dec-19	1.2	INSA	Internal review by Karol Desnos (INSA)
23-Dec-19	1.3	AS	Feedback incorporation
23-Dec-19	1.4	UNISS	Internal review by Francesca Palumbo (UNISS)
24-Dec-19	1.5	AS	Feedback incorporation, quality assurance, added appendix
30-Dec-19	1.6	AS, IBM	Minor refinements based on TNO feedback, Details on AOW IBM

Table of contents

1. Executive Summary

This document describes the M36 Ocean Monitoring demonstrator. The document builds on and is a continuation of D6.9, which described the M18 Ocean Monitoring demonstrator as one of three CERBERO use-case scenarios.

The greyed-out text comes from D6.9 and is included in the document to preserve self-consistency of the document, and to help show the difference between M18 and M36 demonstrators.

The M18 demonstrator comprised the run-time and design-time components and demonstrated the initial version of surface and underwater (shallow water) physical camera system prototype, initial version of data storage system, data fusion models, OM configuration assessments based on high level models, and adaptive image enhancement methods.

The M36 demonstrator is a follow-up of the M18 demonstrator and adds to M18 OM the final surface and subsea camera system (deep water) physical prototype, a hub for managing adaptive cameras, the data storage and retrieval system, two image fusion models for object detection and tracking, and secure communication between the demonstrator components, the optimisation model for image enhancement, and adaptivity of the object detection and tracking model. It uses additional sensors such as temperature and pressure to facilitate system's adaptivity.

The M36 demonstrator is based on:

- CERBERO methodologies: **Adaptation Loop, KPIs**
- CERBERO tools: **DynAA, AOW (by TNO and IBM respectively)**
- CERBERO technologies: **Fusion Models (by AS)**

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.

Related Documents

- *D1.3 – Open Data Plan (Final version*
    - *A big data set of 85 million records has been made public by the Ocean Monitoring use case to facilitate further research and collaboration*
- D2.1 – Description of Scenarios (Final version)
    - The Ocean Monitoring demonstrator will be based on the use case scenario defined in D2.1

- D2.2 – Technical Requirements (Final version)
    - The development of the demonstrator will contribute to satisfy and validate the requirements listed in D2.2
- D3.1 – Modelling of KPI (Final version)
    - The addressed KPIs are based on the generic list of KPIs as defined in D3.1
- D3.3 – Cross-layer Modelling Methodology for CPS (Final version) –
    - The methodology applied for cross layer modelling of the CPS of the Ocean Monitoring demonstrator is described in D3.3
- D4.1 – Multilayer Adaptation (Final version)
    - The deliverable is based on D4.3 and describes the adaptation approaches and data fusion models used to enable adaptivity
- D5.2 – Framework Components (Final version)
    - The framework used for setting up and interfacing CERBERO tools in the (M36) demonstrator are based on the framework components described in D5.2
- D6.1 – Demonstration Skeleton (Final version)
    - The generic skeleton used to build the Ocean Monitoring demonstrator is described in D6.1

2. Scope and purpose

In this document the M36 demonstrator of the Ocean Monitoring use case is described in order to demonstrate and validate the CERBERO methodology, technologies, and tools for cross-layered and adaptive CPS. The OM demonstrator is an example of run-time and design-time system. It uses various KPIs to evaluate how close the CPS currently is to its expected goals/performance.

The M36 OM demonstrator extends the functionality of M18 demonstrator. It incorporates additional sensors such as temperature, and pressure sensors. It comprises ruggedized deep-sea camera system with the additional sensors, object detection and tracking capabilities, and sophisticated storage and retrieval component. This extended functionality is especially valuable in underwater/subsea and surface scenarios such as surface ship detection and tracking, and subsea ocean monitoring.

The M36 demonstrator provides the second iteration of the demonstrator. The development was based on the configuration assessment using DynAA tool, an optimisation model for image enhancement using AOW, the underwater camera prototype, storage and retrieval model, integrating the (M18) demonstrator components, and data fusion models to enable image enhancement and adaptivity.

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

Table 2-1 – OM User to Technical requirements mapping with related assessment strategy (M36 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.	M18 initial prototype. *M36 surface and deep underwater final prototypes.* M18 first version of data storage system. *M36 data storage and retrieval system for surface and subsea scenarios.*

		On demand task dependent Data Fusion. Secure communication.	M18 data fusion models for image fusion and retrieval. *M36 data fusion for object detection, recognition and tracking.* *M36 secure communication between demonstrator components.*
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 real time applications.	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 using CERBERO technology. *M36 incremental prototyping using CERBERO methodologies (adaptation loop, KPIs), tools (DynAA, AOW), and technologies (fusion models).*
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. *M36 optimisation model and solution for image enhancement* *M36 adaptation of object's visual characteristics for the tracking purposes based on motion and deep learning object detection* *M36 adaptation based on readings from temperature*

			and pressure sensors

In Table 4 of D2.1 the user requirements are mapped to technical requirements. In this document we are describing how the validation is performed and when. The various elements and functionalities of the demonstrator are distributed between M18 and M36.

OM requirements were prioritised as specified in Table 4-5 from D2.4. During working on the plan of prototypes it became evident that there were more challenges for subsea aspects of the use cases, and arguably fewer technologies available for these than the sea surface use cases. Also, some techniques and tools used for subsea could also be deployed on the sea surface. To minimize risk and give more flexibility in work (a flexibility that was needed because of the iterative process inherent in the customer discovery approach to exploitation), we prioritized the subsea over sea surface requirements. Similarly, to minimize risk and allow for more time to meet the challenges, we upgraded the priority of visual sensing or adaptive camera systems related requirements.

3. Description of the Ocean Monitoring demonstrator

The OM demonstrator is both a run-time and design-time demonstrator. It involves the developed models and software as well as developed physical prototypes. It can be placed in between the other two use cases, the more FPGA hardware-oriented Planetary Exploration use case and the Smart Travelling software and system-of-system oriented use case. The OM high level solution components are depicted in Figure 3-1.

The OM solution providing the underlying infrastructure for the M36 demonstrator includes: the adaptive camera, the hub, ROV components, an umbilical cable, and an information storage system for data archiving (Figure 3-1). The adaptive camera enables enhanced-vision and can be used with other sensors such as pressure and temperature. The hub integrates with the information retrieval system capable of analysing and indexing video streams for storage and retrieval purposes. Its functionality can also include video enhancement. The ROV components are based on open platform technologies that enable both hardware and software technological extensions and flexibility. The adaptive camera and ROV components need to connect via umbilical cables to the hub. The data and information storage system is there to enable retrieval and archiving of collected information from sensors such as camera, temperature, and pressure.
The adaptive camera and the hub implement data fusion models for image enhancement and object detection and tracking. The hub helps maintain overall situational awareness based on sensor information. The adaptive camera and the hub have been developed by using the DynAA and AOW tools at design time for optimisation purposes. The demonstrator uses sensors such as camera, temperature, and pressure for enhanced situational awareness and sensing of the environment. The sensors, tools, and technologies such as data fusion models enable different notions of adaptation of the demonstrator. The demonstrator supports both autonomous local and externally managed adaptation.

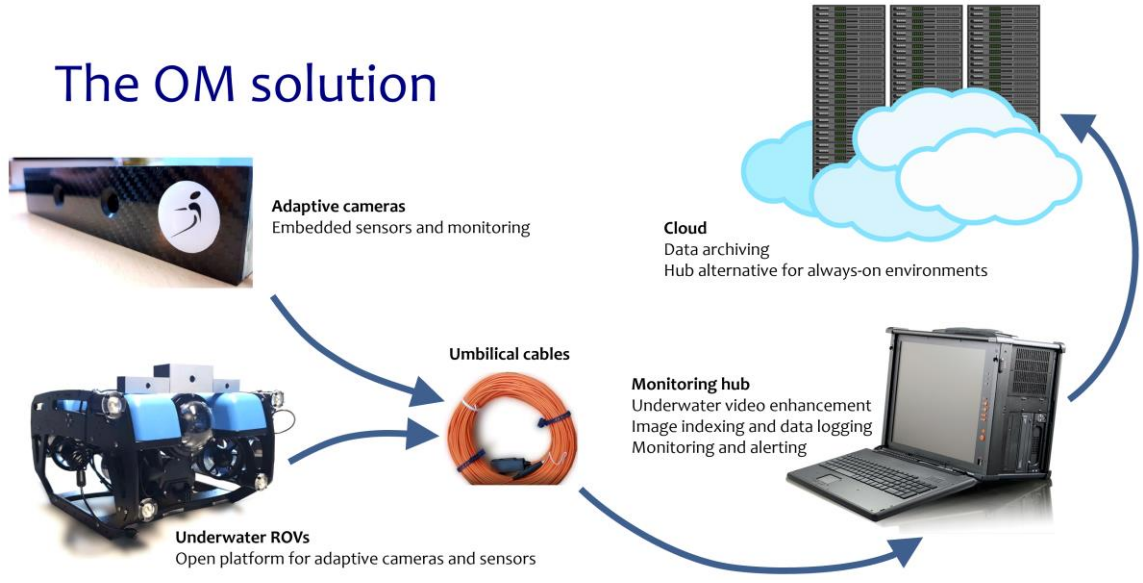


Figure 3-1.The Ocean Monitoring Solution.

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.

The M36 OM demonstrator shows selected aspects of the overall OM solution, described above. Derived from these high-level CERBERO requirements three demonstrator goals accomplished for the M36 demonstrator (see D2.1):

1. *Adaptive camera prototypes development for marine robots.*
   1.1 Development of new surface and deep underwater adaptive camera prototypes.
   1.2 Development and implementation of final data storage and retrieval system for surface and subsea scenarios.
   1.3 Development and implementation of information fusion models for object detection, recognition and tracking.
   1.4 Secure communication between demonstrator components.
2. *Incremental prototyping using CERBERO methodologies (**Adaptation Loop**, **KPIs**), tools (**DynAA**, **AOW**), and technologies (**Fusion Models**)* – respectively from WP3, WP4, and WP5
3. Adaptation approaches.
   3.1. *Development and implementation of optimisation model and application of CERBERO technologies and tools to finding the solution for image enhancement.*
   3.2. *Adaptation of object characteristics for tracking purposes based on motion and deep learning detector.*
   3.3. *Adaptation based on temperature and pressure sensors.*

The first goal is accomplished by the development of the adaptive camera system final physical M36 prototypes for surface and deep underwater scenarios. The camera can withstand a few hundred meters pressure depth and can adapt 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 M36 demonstrator can detect and track moving objects based on the fusion of colour-based and frame difference frameworks. It can also detect, recognise, and track objects such as ships based on the fusion of convolutional neural network and a tracking model.

The second goal is the development of 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. The OM demonstrator uses CERBERO methodology, tools and technologies to facilitate the incremental prototyping process.

The third goal relates to the development of image enhancement techniques for the adaptive camera system. The M36 OM demonstrator formulates and solves the optimisation problem for the fusion model for image enhancement to find the optimal weights representing the importance of different enhancement techniques. The demonstrator also adds the adaptation of detected object's characteristics for tracking purposes. Moreover, additional sensors such as temperature and pressure sensors facilitate the demonstrator's adaptivity to changing environmental conditions.

Figure 3-2 shows the main components of the M36 demonstrator. The colour coding is as follows:

- Green background represents the run-time components of the demonstrator.
- Orange background represents design-time components of the demonstrator.
- Red background represents CERBERO tools.
- Blue background represents the physical components.

The demonstrator consists of the following components:

- Surface and deep underwater *camera system prototype* that adapts its vision to changing environmental conditions and adapts object characteristics based on object's motion and other detected features for tracking purposes. This has been researched, developed, and implemented for the OM use case.
- *Hub* that manages a collection of adaptive cameras, provides indexing, retrieval, user and control interfaces. The functionality and complexity of the hub depends on installation context.
- *Information storage system* for storing and retrieving images, videos, and other data obtained from sensors. This system is adapted and extended for the OM use case.
- *Data fusion models* for image enhancement and retrieval, and object detection, recognition, and tracking. This has been researched, developed, and implemented for the OM use case. As stated in the project proposal – information fusion enables the adaptivity of the M36 demonstrator.
- *Temperature and pressure sensors* for monitoring the environmental conditions. These are integrated for the OM use case.
- *Secure communication* between the OM demonstrator components.
- *CERBERO tools and technologies* used to develop the M36 OM demonstrator.

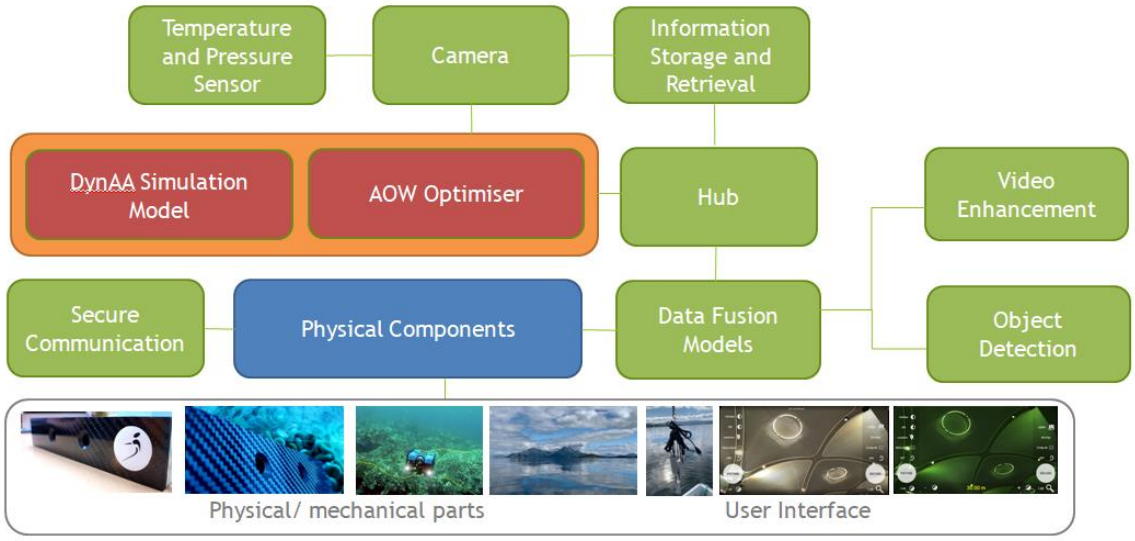


Figure 3-2. Components of the M36 OM demonstrator.

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

Here is a short recap of the different elements for M36 implementation. Note that since these are organized along the pattern of the CERBERO adaptation cycle, we will expand upon this later in Section 3.1.3 below:

1. KPI Estimator (a model): represents a way to evaluate how close the CPS currently is to its expected goals/performance. Information for the KPIs is derived from sensors such as optical, temperature, and pressure sensors.
2. Manager: defines if adaptation is needed, according to the distance between the evaluated KPIs and the final system goal. Adaptation can also be managed at local level.
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.

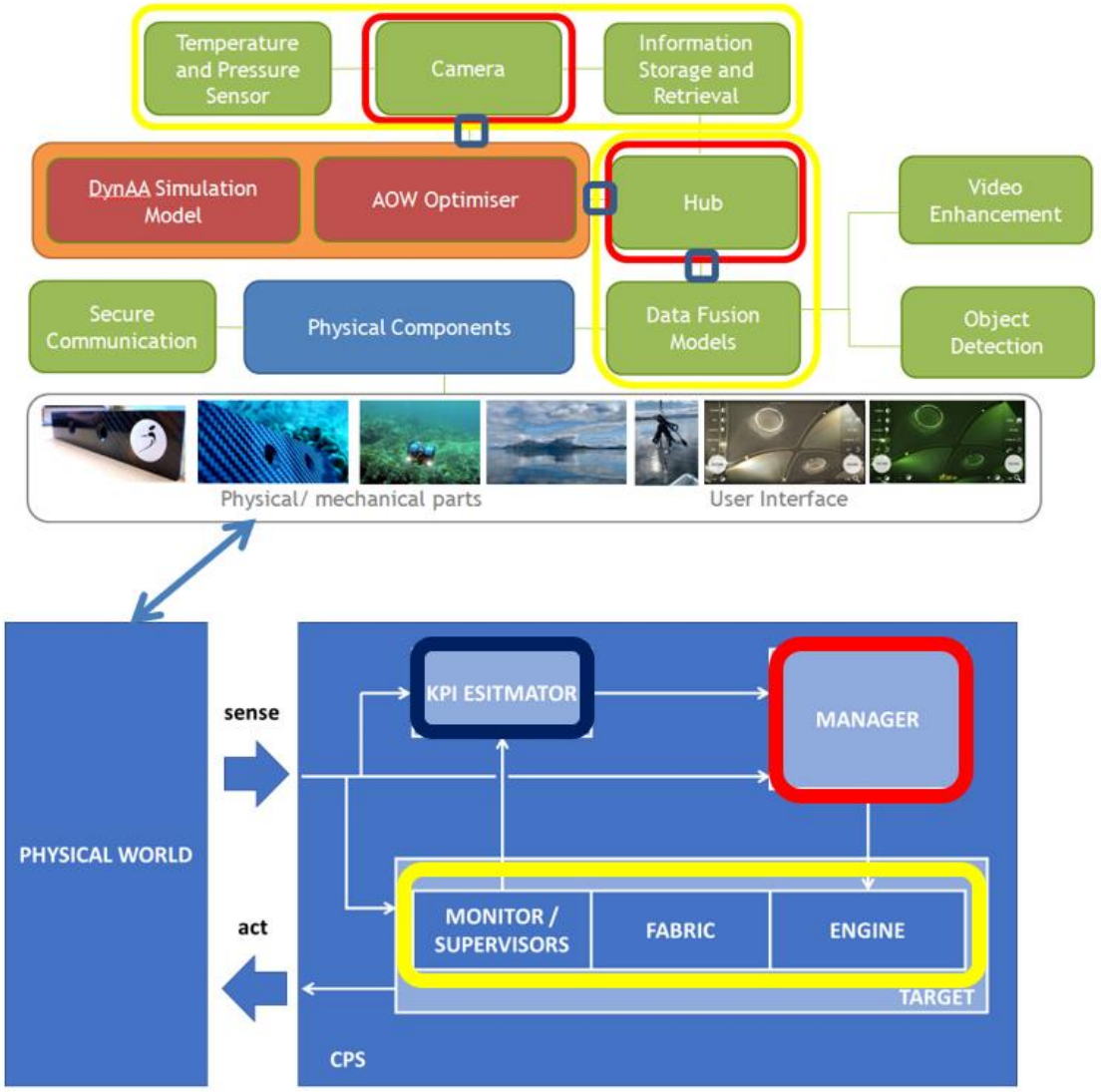


Figure 3-3. Ocean Monitoring use-case - CERBERO skeleton mapping. Notice the color-based mapping between the two diagrams.

3.1. Functionalities

The implementation of the M36 Ocean Monitoring demonstrator functionalities (according to the goals and requirements of D2.1) is divided into the following parts:

1. DynAA-based simulation of the OM components for different configurations assessment, and AOW-based optimisation model for adaptive image enhancement (OM2).
2. Physical prototypes of the Camera System (OM1, OM2).
3. Data fusion to enable adaptivity (OM1, OM3).

4. Image enhancement (OM1).
5. Object detection and tracking (OM1).
6. Adaptation approaches (OM3).
7. Information storage and retrieval system (OM1).

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

This section presents the developed DynAA simulation models which were used to assess alternative design configurations choices to ensure that the final hardware platform meets the required KPIs.

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. Physical prototypes of the camera system

The M18 demonstrator included multi-lens camera prototypes with the focus on their applications for image enhancement and navigation purposes.

The M18 initial physical prototype of the Camera System comprising 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, such as battery level.

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

Figure 3-6. Stereo calibration.

Figure 3-7. Stereo rectification.

The M36 camera prototype adds secure wireless communication, temperature, and pressure sensors, image storage and retrieval, and object detection and tracking, and the ability to work with a hub. Its architecture consists of sensors, camera and hub processors, control interface, and Solr. Figure 3-8 presents the overall architecture of the prototype.

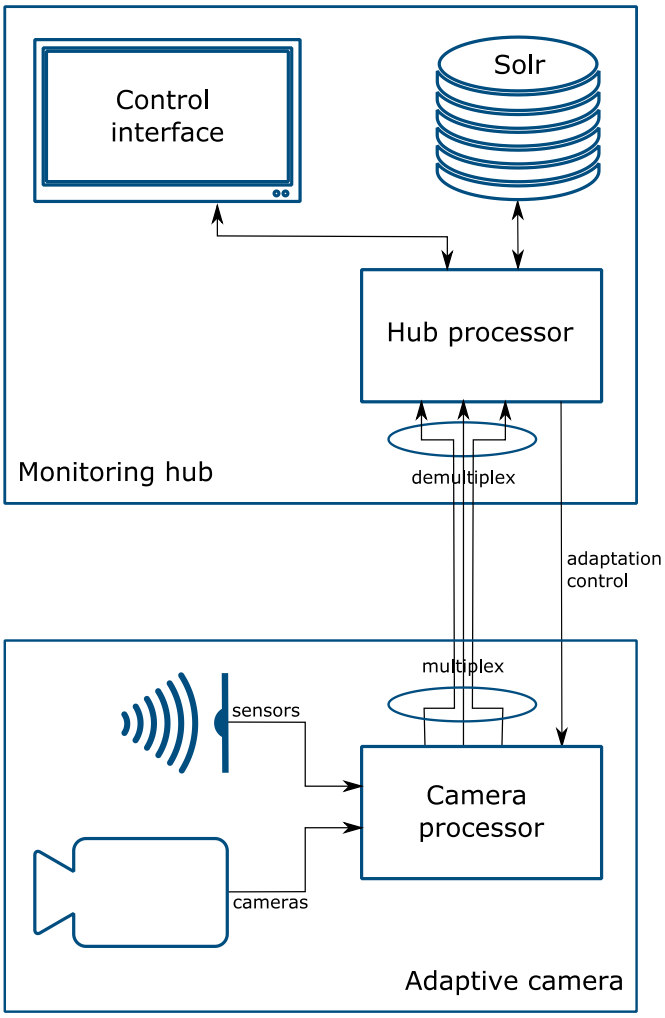


Figure 3-8. Camera system architecture.

There are two major components shown in **Figure** 3-8: a monitoring hub and an adaptive camera. The monitoring hub can manage a collection of adaptive cameras, provides indexing, retrieval, user and control interfaces. The adaptive camera provides sensor and video interfaces. These are designed to meet the user requirements in common ocean monitoring scenarios. For example, in a live monitoring setup, the two will be constantly connected, providing live data. In an assessment situation, especially one involving usage at depth, the two may be disconnected for some time. This means the adaptive camera was developed for both autonomous local and externally managed adaptations.

In the OM demonstrator, both video and other sensor data can be multiplexed and combined using a generic video data container format such as Matroska – this allows the sensor data to be sequenced, stored, retrieved, and displayed alongside regular camera data. In the reverse direction, adaptation commands can be sent from the controller/hub to the adaptive camera – typical commands might be requests to stop and start recording, modify lighting, switch lenses to ones more suitable for the visual environment.

The M36 subsea camera prototype is a ruggedized deep-sea camera system with an umbilical cord to allow for a live preview of the recorded footage.

3.1.3. The adaptation cycle.

Within Figure 3-8, there is a hierarchical adaptation process that implements the CERBERO adaptation pattern. In the M36 demonstrator, the adaptation cycle is implemented using an API that provides the following interfaces:

- **Manager** — responsible for processing incoming sense and monitor data, integrating it with model information, and making recommended actions to the engine.
- **Engine** — responsible for decision-making, processing received actions, and then making changes to fabrics
- **Monitors** and **sensors** — we support two different kinds of monitor: a pull mode monitor, which is polled for data, and a push mode sensor, which automatically updates the manager with sense data as it becomes available.
- **Fabrics** — are the interfaces to effectors and are responsible for applying changes to underlying hardware.
- **Models** — allow predictions to be used to estimate KPIs. Models can be developed through mathematical models, machine learning, or even simple rules. They allow the manager to make inferences about sense data and use them in decision-making on how to act.

There are separate C and Java implementations of these APIs, with C used in more embedded contexts, and Java where more performance is available. The Java implementation uses traditional classes and interfaces, in C it uses headers and structures to a similar effect. The main difference is that application logic in Java is added by implementing classes; in C by putting function pointers into linked structures.

To see this in more detail, let's consider one simple application of the adaptation cycle in an embedded system, in an adaptive camera which supports a light, and which changes the light's color to adapt to depth. The primary incoming data sensor is the camera, which is constantly receiving frames and multiplexing them for recording and transmission to the monitoring hub. While this is happening, periodically frames are sampled and decoded. For key frames (i.e., frames which do not require interpolation) they can be decoded directly and independently into a frame, which can then be processed for color. The monitor here decodes key frames, which may typically use a color system like YUV, maybe with chroma subsampling. This can then be processed by a monitor to generate a median color value, which is passed to the manager.

At this point, the manager can pass this value to a model, possibly along with information about the current lighting status, and use it to infer a depth. The manager can then compare that depth with the current lighting status, and use it to decide whether or not to change the lighting status. If it does, that status is passed to the engine.

The engine does two things: it updates the lights themselves, so that the external environment is modified. It can also modify the model with a representation of the new lighting status, so that the two remain consistent. To be safe, the engine may not make a full change, but may make a damped or smoothed change to the lighting, so that the risk of positive feedback and oscillations is reduced. To make this easier, the API provides fabrics

that interface to the underlying hardware (like the monitors and sensors do) so the engine passes the changes to fabric software components which provide that level of hardware interface.

At this point, one pass around the adaptation cycle is completed, and time will elapse before the next sensing phase begins another pass around the loop.

In the case of Ocean Monitoring, these loops are modelled as independent units that can be assembled into a hierarchy in the mode of 4D/RCS "nodes", where each node has a parent and may have children. Sense data from monitors and sensors can be passed to a parent node, and changes decided by an engine can be applied to child nodes. In the Ocean Monitoring use case, the adaptive camera forms a child node, and the monitoring hub forms a parent node. Video data, and other sensor data, is passed from the adaptive camera to the monitoring hub directly. In the Ocean Monitoring demonstrator, data is sent between the two components using ZeroMQ for sensor data, and video over HTTPS for video data. In the hub node, for example, the adaptation manager uses a model to adaptively select appropriate enhancement methods, as described next in Section 3.1.4

The monitoring hub's adaptation cycle may make configuration changes that are applied down to any of its attached adaptive cameras (there may be any number of adaptive cameras attached to a single hub). However, this happens alongside the adaptation system that happens entirely within each adaptive camera node.

3.1.4. Image enhancement methods

The M18 demonstrator incorporated image enhancement approaches to increase the situational awareness of the robot and its operator.

The M18 and M36 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 3-9 shows the results of histogram equalization.

The results of fusion of the original with the novel edge image are depicted in Figure 3-10. The fusion model de-noises the image and enhances its edges.

Figure 3-9. Underwater images and their corresponding histograms before and after histogram equalization.

Figure 3-10. Image enhancement by information fusion. Left - original image, right - enhanced image.

The M36 image enhancement uses the AOW optimiser to solve the optimisation problem that would lead to finding optimal weights associated with different enhancement techniques that would be fused together. The problem is formulated as follows:

The goal is to find the optimal combination of contributions of different image enhancement techniques. We use an automatic evaluation of the image quality based on the points of interest (key points). The points of interest are the most representative areas of sudden pixel intensity change. There are many approaches to automatic image/video quality assessment [Mohammadi 2014], with various methods based on key points detection [Oszust 2019], [Zhang 2015], [Lu 2014].

We want to maximize the assessed video quality using one of the established quality metrics over the weights corresponding to different image enhancement methods e.g. local and global histogram equalization, contrast enhancement, brightness adaptation, sharpening or blurring, etc.

$$\max_{w\in[0,1]} |w^T \cdot I| = \max_{w\in[0,1]} \left| (w_0, w_1, \dots, w_N) \cdot \begin{pmatrix} I_0 \\ I_1 \\ \vdots \\ I_N \end{pmatrix} \right| = \max_{w\in[0,1]} \left| \sum_{i=0}^{N} w_i \cdot I_i \right|$$

$$\sum_{i=0}^{N} w_i = 1$$

Here, the notation is as follows:

- w_i is a weight associated with an enhanced image I_i – (for example denoised)
- $|\cdot|$ is the assessed image quality
- N is the number of enhancement methods
- I_0 is the original image
- $\sum_{i=0}^{N} w_i \cdot I_i$ represents a type of image fusion as a linear combination of weighted images

The result of this optimization is an AOW-constructed design-time model that is ‘baked in’ as part of the adaptation model. This can be used by an adaptation manager to predict the impact of algorithm weights – essentially the fabric in this adaptation component. This allows the manager to select the appropriate combination of algorithms to ensure the best result dynamically. The AOW tool then allowed us to build the model needed to power one of the adaptation managers of the OM demonstrator.

3.1.5. Data fusion techniques for image enhancement, image retrieval, navigation, and object detection and tracking

The M18 demonstrator used various data fusion models for image enhancement, image retrieval, and navigation.

The information fusion techniques used in the M18 OM 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 3-11 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.

Figure 3-11. Notions of correlation in the context of information retrieval.

The M36 OM demonstrator incorporates two new data fusion techniques for the purpose of detecting and tracking objects. Both fusion techniques were designed and implemented from an adaptive systems viewpoint – i.e. to perform automatic detection and tracking of objects for obstacle and collision avoidance purposes.

The first one uses motion information to detect moving objects. The visual features related to detected moving objects are then extracted from the motion area. These visual features represent the colour distribution in the motion window, which are then used for tracking purposes. This is a combination of background subtraction and colour-based tracking frameworks. The diagram in Figure 3-12 illustrates the fusion concept.

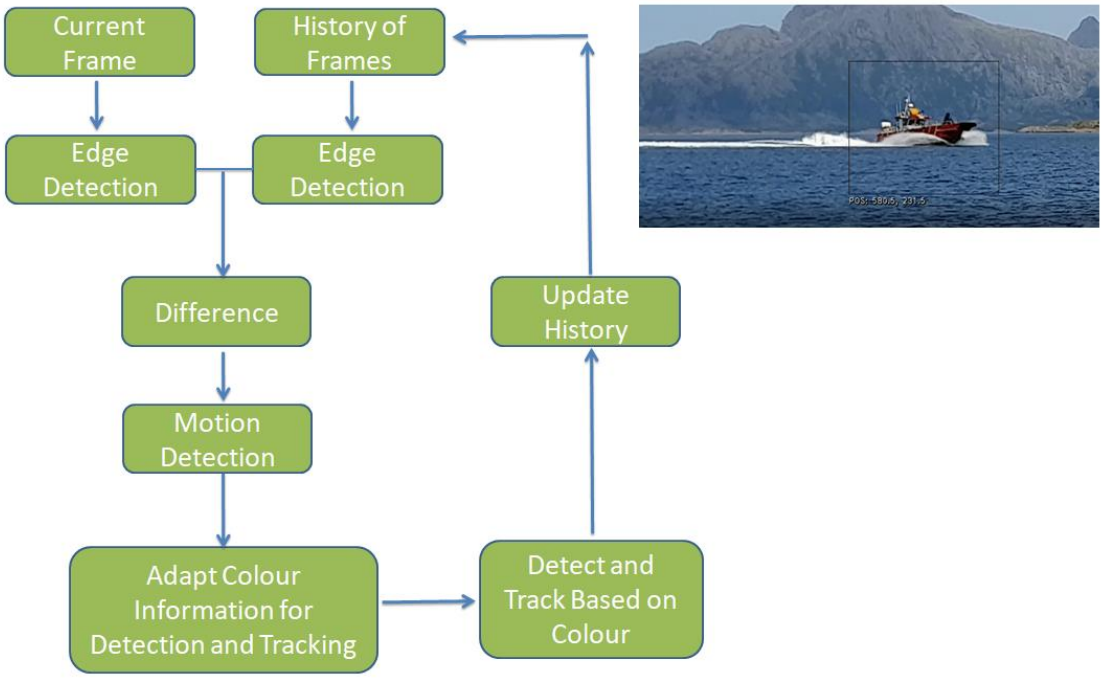


Figure 3-12. Background subtraction-based detection and color-based tracking fusion for object detection.

The second data fusion approach for object detection and tracking detects objects such as ships using convolutional neural networks. The information about the object's location is then fed to the tracker. This is a combination of deep learning-based object detector and correlation-based tracker. The diagram in Figure 3-13 illustrates the fusion concept.

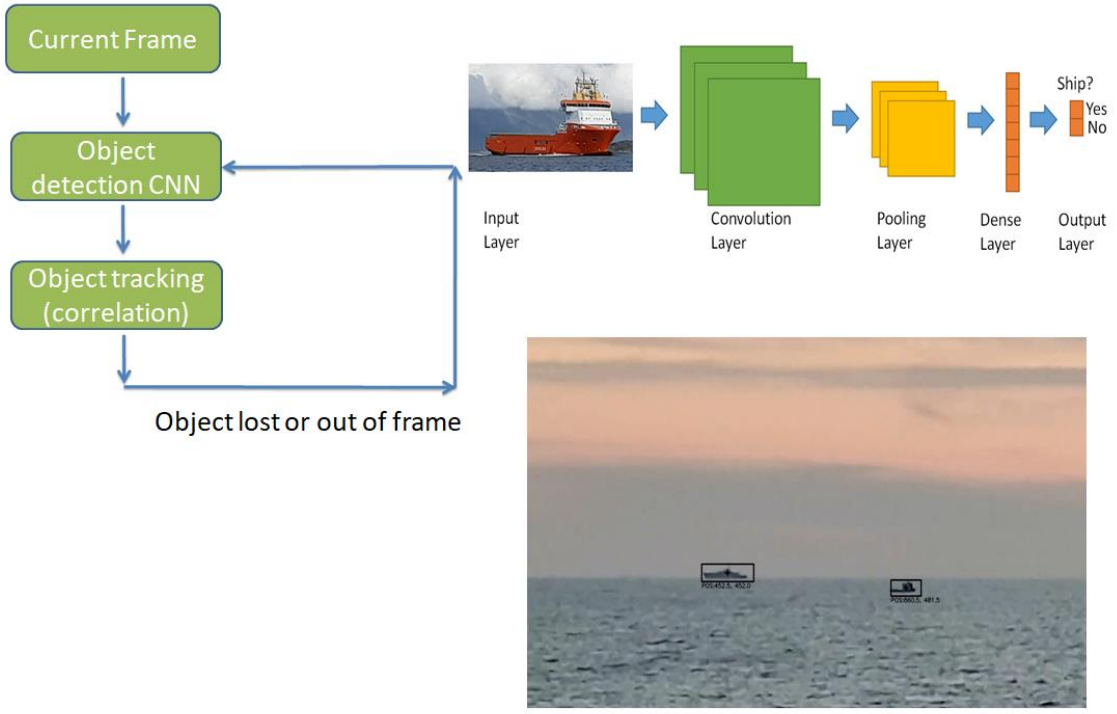


Figure 3-13. Deep learning-based detection and correlation based tracking fusion for object detection.

Figure 3-14. Image Enhancement based on Lighting Conditions.

3.1.6. Adaptation approaches

The M18 demonstrator used 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 3-14 presents different levels of image enhancement. The levels of image enhancement will correspond to camera's self-adaptation to different lighting conditions.

The M18 demonstrator allowed the user to trigger adaptation by requesting specific functionality. It also automatically adapted the levels of video enhancement to changing light conditions of the environment.

The M36 demonstrator uses additional sensors such as temperature, humidity, and pressure to trigger adaptation of the camera system.

The camera compensates for the colour loss at different water depth, as described above in Section 3.1.3. Based on the data from pressure sensor and the model describing which colour disappears at which depth, the demonstrator compensates for the lost colour in order to present a visually appealing image comprising a full range of colours.

The M36 OM demonstrator uses the temperature sensor to adaptively change the level of image de-noising. The higher the environmental temperature is the more noise would be generated by the optical sensors which can result in deterioration of the image quality [Lin 2010], [Abarca 2017]. The goal of this adaptation, similarly to the colour compensation at different depths, is to make the image quality relatively invariant to changes of temperature.

The M36 demonstrator adapts the tracker visual characteristics based on detected object's location. The first model adapts the colour characteristics of the tracker based on the object's motion. Thus, the colour information gets updated when the motion occurs. The second model updates the correlation-based tracker's characteristics using convolutional neural network. The adaption is triggered when the tracker loses the tracked object which can occur because of occlusion, drastic change of viewpoint, or the object getting out of the camera field view.

3.1.7. Information storage system

The M18 demonstrator developed and implemented data fusion models for the combination of text and visual features to improve the retrieval. Both text and images could be stored in Solr and used to retrieve relevant information.

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 3-15 shows the top results of image search by visual example. Figure 3-15 presents the storage format for textual and image indexes.

Figure 3-15. 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 vi
      "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 vi
      "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 beacl
      "_version_": 1524997319443873800
    },
```

Figure 3-16. Storage system of textual and visual information.

The M36 OM demonstrator incorporates the final version of the Storage and Retrieval System. The data from sensors is streamed in real time to the cloud and a browser. The data from sensors is stored alongside video data and is continuously indexed in Solr. The extracted features include visual features based on the "bag of visual words" framework. This makes stored data retrievable for user and the AI methods. The M18 demonstrator focused on the development of data fusion models and appropriate information object representations for storage and retrieval purposes. The M36 demonstrator integrates the adaptive camera with the storage and retrieval system so that the visual content of the keyframes extracted from the data stream can be constantly processed and indexed.

3.2. CERBERO Methodology, Technologies, and 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.

The M36 OM demonstrator follows the established CERBERO methodology such as the adaptation loop and the KPI modelling. It uses the following CERBERO technologies and tools: DynAA, Data Fusion Models, and AOW. The Data Fusion technology is used at runtime, and DynAA and AOW are used at design time.

The adaptation loop incorporated by the OM demonstrator allows for the adaptation break-down into separate components. To achieve this, the OM demonstrator uses a hierarchical version of the CERBERO adaptation cycle, shown in Figure 3-17. This extends the basic cycle with an inter-node communication model based on the Albus 4-D/RCS reference model, originally designed for autonomous vehicles. The extensions simply allow sense inputs to be relayed to parent nodes, and instructions for adaptation to be accepted from those same parent nodes. Each major block in Figure 3-17 has its own independent adaptation system, with the adaptive camera playing the 'child' role, and the monitoring hub the 'parent' role.

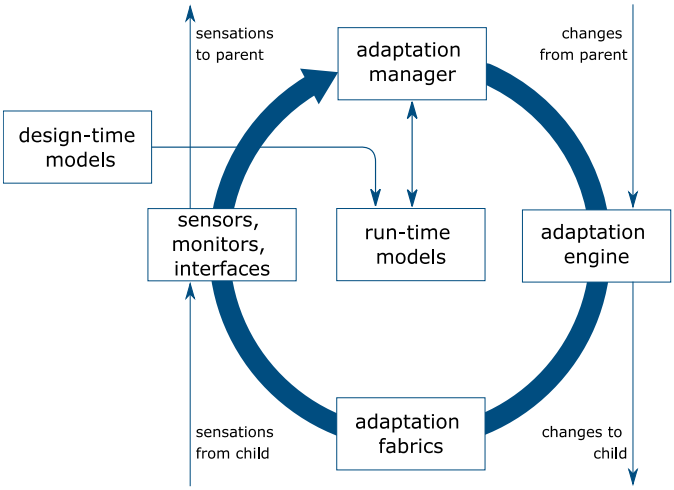


Figure 3-17. The OM demonstrator updated adaptation cycle.

Note that the OM demonstrator also incorporates design-time models as part of its run-time modelling. This is a key feature of the adaptive camera, which uses an optimization model built using AOW as part of its adaptive algorithm selection technology. Put simply: we use AOW to build a model that determines which techniques work best to improve video quality, depending on features of the raw video such as: brightness, contrast, amount of blue, and so on. The adaptation manager can then use this to adapt the video processing algorithm selection (a fabric, in CERBERO terms) dynamically.

3.3. Development and deployment environment

Tools used by the OM demonstrator include:

- **Java development environment** – especially for the hub processor, interfacing to Solr, DynAA-based adaptation, and so on. We have developed a Java API for the adaptation cycle, and this provides the major high-level framework for the hub processing node.
- **C development** – for the adaptive camera. In many respects, this uses a similar architecture, again based on the CERBERO adaptation cycle. Using C does reduce developer productiveness (one of the most critical KPIs for the OM demonstrator) but does allow more flexibility and efficient use of embedded hardware.
- **OpenCV** – for most of the video processing technology. OpenCV has both a Java interface and C interface, so it allows us to make the code relatively portable. However, the Java interface does not generally support any accelerators such as GPUs, due to constraints imposed by Java memory management. Fortunately, acceleration is generally transparent, after migration to C, GPUs and other accelerators become viable.
- **DynAA** – at design time, to validate the processing throughput for the alternative COTS technology platforms at proof-of-concept stage. This correctly helped us to identify the primary KPIs that were particularly critical to technology selection.

Once selected, these model predictions remained crucial to guiding later technical decisions.

- **AOW** – to build optimizing models of video processing techniques. We prepared a large dataset of images from videos in a wide range of ocean monitoring scenarios, assessed them using processing-cheap metrics like brightness, colour points, contrast, and histograms, and used that to pre-build a static model of which techniques work best to optimize for quality. This model is then baked into the run-time model for the adaptive camera. For background, AOW (Architecture Optimization Workbench) is a tool for System of System (SoS) and System level multi-objective multi-view cross-level design using mathematical programming techniques. AOW allows models of metrics, views and levels using algebraic mixed-integer linear equations and inequalities. It is extended also for models using linear differential equations and inequalities as well. Using AOW, system (or SoS) designers can simultaneously find the optimal system (or SoS) architecture and control, including routing and reconfiguration decisions. Here, for the OM use-case it is used for designing and optimizing image enhancement methods.
- **ffmpeg libraries** – for the adaptive camera core logic, directly interfacing to video camera sensors, multiplexing, container management and so on. Note that, as our work at M18 showed, by far the most limiting resource for the OM demonstrator is video processing capacity, and much the simplest solution to this – and the one adopted by the OM demonstrator at M36 – is to pass through encoded data wherever possible, minimizing decode-encode cycles which have a high cost in both quality and performance. There are several ffmpeg libraries for different purposes, e.g., managing devices, codecs, container formats.

Our minimum version of the adaptive camera (as in Figure 3-17) comprises a camera sensor and a minimum temperature, pressure, and humidity sensors, which are multiplexed into the data stream.

One note regarding ffmpeg: it provides an exciting opportunity for further use of the CERBERO tools, especially those like ARTICo^3. While it is not exactly "open" in the sense that new modules can't be added to it without recompiling, it is relatively easy to write new filters using a standard API and bundle them into a custom build. It also uses a dataflow pattern extensively, especially for its filters. It is, therefore, possible to write C filter API wrappers to drive ARTICo^3 accelerators, and package them transparently into ffmpeg video processing pipelines. This is a great strategy for the OM demonstrator, as it allows accelerator development to be relatively decoupled from the rest of the demonstrator architecture.

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:

[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.

- *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 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 3-4 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

[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.

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.

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

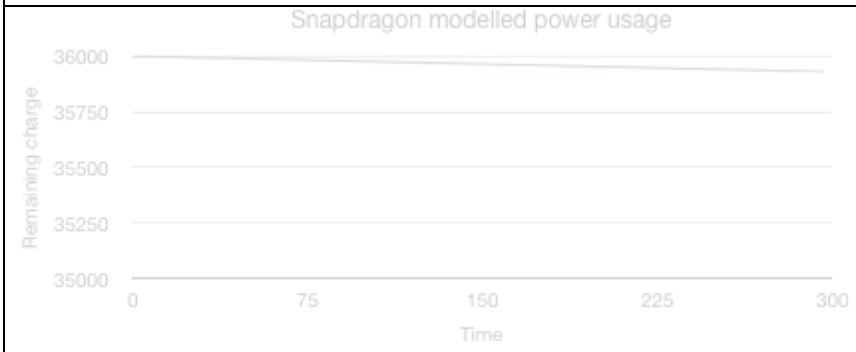
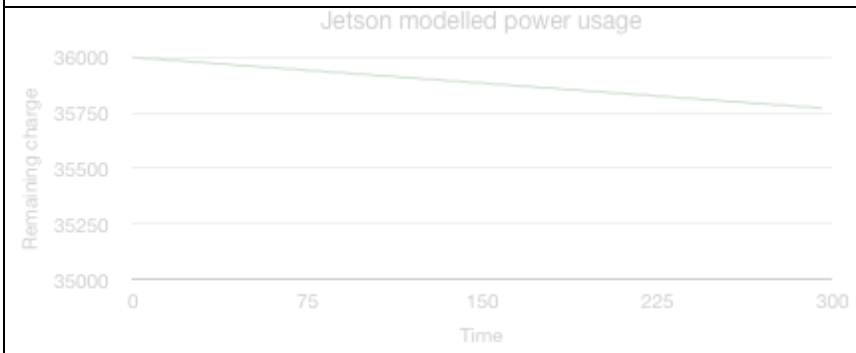
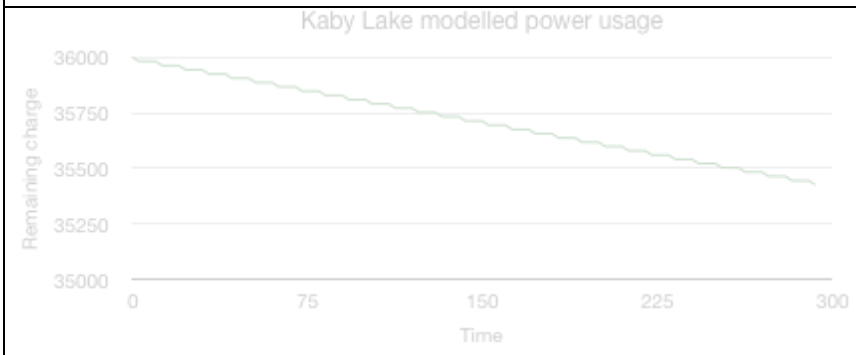
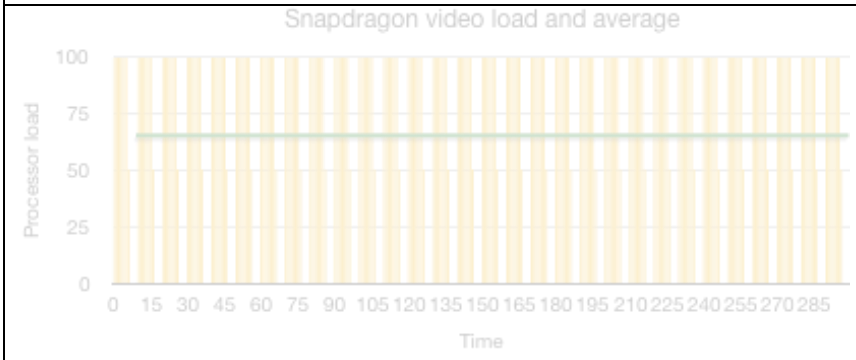
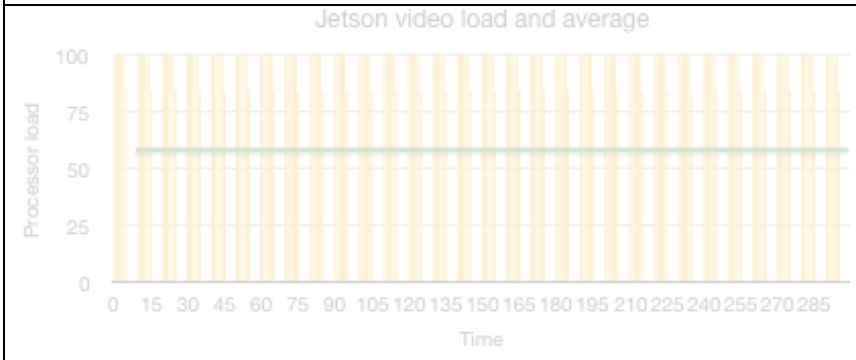
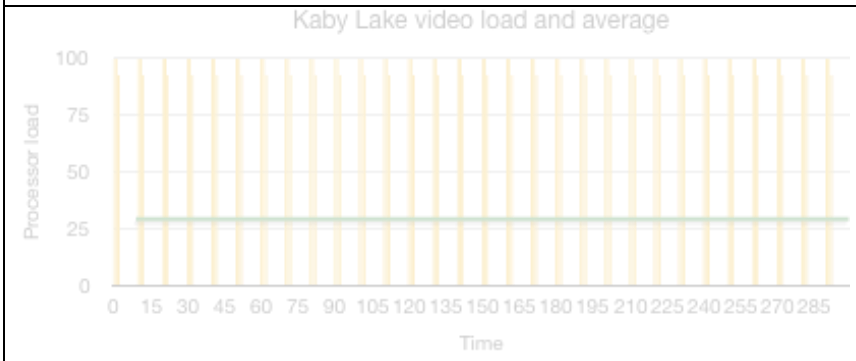
Platform results	Notes
	Based on 1 x 4k camera, video processed in 10 second blocks. Five minutes of video requires around 70 coulombs, or 20 mAh.
	Based on 1 x 4k camera, video processed in 10 second blocks. Five minutes of video requires around 230 coulombs, or 65 mAh.
	Based on 1 x 4k camera, video 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 key-points detected before and after the adaptive image enhancement.

The OM use case for M36 tested: OM system functionalities; the deep-water camera prototype; the algorithms and possible hardware boards they can run on, along with video data throughput to the cloud; the data fusion model for image enhancement optimisation. The M36 Ocean Monitoring demonstrator tests the following OM system functionalities:

- Final physical surface and subsea prototypes of the Adaptive Camera System
- Fusion for image enhancement in the context of AOW based optimisation
- Data fusion techniques for object detection and tracking
- Adaptation approaches: temperature sensor based adaptive image de-noising, pressure sensor based adaptive colour loss compensation
- Information storage, indexing, and retrieval system. Indexing of incoming data stream from sensors. On-the-fly visual features extraction for query by visual example.

The underwater camera prototypes underwent ruggedization testing in harsh Arctic waters in Norway. They were submerged to the specified depths and then retrieved and tested to check if they were still functioning properly. The prototypes were able to withstand the submersion to depths of up to 320 meters. The further testing plans include testing of the prototypes at depths of 1000 meters. Figure 4.1 shows pictures from the open ocean waters tests.

Figure 4-1. Open waters camera testing.

The adaptive camera algorithms needed to be tested on several boards as the camera sensor excludes the board. The adaptive camera modules were tested on high-end hardware Intel Kaby Lake and Snapdragon boards (from 820 up to 865), and also on low-end embedded quad-to-octa core processor boards. The former category was found highly suitable for the adaptive camera deployment for more multiple lenses. The latter category was found

suitable for adaptive camera with single lens. GPUs and other special purpose accelerators, e.g. for video encoding are important to use for video processing and data throughput. GPIO ports are also needed for integration with multiple actuators and analogue sensors. USB 3 was required to obtain high data throughput - especially for 4k and 8k streaming.

In general, we found that high-end hardware boards performed well for image processing and also data throughput, whereas low-end hardware boards had issues relating to computational power and data buses. Overcoming these issues (considering KPIs such as image quality and data throughput), when it comes to low-end embedded hardware, remains a priority for future exploitation, but in the interim, we have adopted a workaround: bypassing video encoding data wherever possible, eliminating unnecessary decode-encode cycles, which have a high cost in both quality and data throughput. We have concluded that all tested camera sensors are capable of real time or near real time streaming to video cloud service and browser.

In summary, we experimented with a range hardware boards from pure GPU boards, via a combination of GPU and multi-core arm processors, to pure multi-core arm processors during the development phases, because video processing requires high-end and optimised heterogeneous boards for real-time processing. Further hardware consideration details can be found in the Appendix. Providing enhanced computer vision for marine robots and human operator requires high-end processing and encoding, whereas less computational CPS-tasks, such as marine robot propulsion and control was accomplished by using low-end embedded hardware.

The M36 demonstrator tests the data fusion model for image enhancement optimisation with AOW CERBERO tool. A big-data set consisting of 85 million records was generated for the test and optimisation purposes, and aspects of it has been made available as open data [D1.3] on the CERBERO project web site to stimulate further research collaborations. An excerpt from the dataset is presented in Table 4-3. The evaluated KPI is the automatically measured image quality.

Table 4-3. Excerpt from dataset for AOW based optimization for image enhancement.

filename	pixels	brightness	entropy	w1	w2	w3	w4	w5	w6	w7	w8	keypoints
video.0001.mp4.frame.0001.jpg	2073600	0.562066251	-131605543.8	0	0	0	0	0	0	0	1	1
video.0001.mp4.frame.0001.jpg	2073600	0.562066251	-131605543.8	0	0	0	0	0	0	0.1	0.9	0
video.0001.mp4.frame.0001.jpg	2073600	0.562066251	-131605543.8	0	0	0	0	0	0	0.2	0.8	0
video.0001.mp4.frame.0001.jpg	2073600	0.562066251	-131605543.8	0	0	0	0	0	0	0.3	0.7	0
video.0001.mp4.frame.0001.jpg	2073600	0.562066251	-131605543.8	0	0	0	0	0	0	0.4	0.6	0
video.0001.mp4.frame.0001.jpg	2073600	0.562066251	-131605543.8	0	0	0	0	0	0	0.5	0.5	0
video.0001.mp4.frame.0001.jpg	2073600	0.562066251	-131605543.8	0	0	0	0	0	0	0.6	0.4	1
video.0001.mp4.frame.0001.jpg	2073600	0.562066251	-131605543.8	0	0	0	0	0	0	0.7	0.3	4
video.0001.mp4.frame.0001.jpg	2073600	0.562066251	-131605543.8	0	0	0	0	0	0	0.8	0.2	14
video.0001.mp4.frame.0001.jpg	2073600	0.562066251	-131605543.8	0	0	0	0	0	0	0.9	0.1	33
video.0001.mp4.frame.0001.jpg	2073600	0.562066251	-131605543.8	0	0	0	0	0	0	1	0	54
video.0001.mp4.frame.0001.jpg	2073600	0.562066251	-131605543.8	0	0	0	0	0	0.1	0	0.9	1
video.0001.mp4.frame.0001.jpg	2073600	0.562066251	-131605543.8	0	0	0	0	0	0.1	0.1	0.8	0
video.0001.mp4.frame.0001.jpg	2073600	0.562066251	-131605543.8	0	0	0	0	0	0.1	0.2	0.7	0
video.0001.mp4.frame.0001.jpg	2073600	0.562066251	-131605543.8	0	0	0	0	0	0.1	0.3	0.6	0
video.0001.mp4.frame.0001.jpg	2073600	0.562066251	-131605543.8	0	0	0	0	0	0.1	0.4	0.5	0
video.0001.mp4.frame.0001.jpg	2073600	0.562066251	-131605543.8	0	0	0	0	0	0.1	0.5	0.4	0
video.0001.mp4.frame.0001.jpg	2073600	0.562066251	-131605543.8	0	0	0	0	0	0.1	0.6	0.3	1
video.0001.mp4.frame.0001.jpg	2073600	0.562066251	-131605543.8	0	0	0	0	0	0.1	0.7	0.2	4
video.0001.mp4.frame.0001.jpg	2073600	0.562066251	-131605543.8	0	0	0	0	0	0.1	0.8	0.1	12

4.2. Results

In the Table 4-4 below, in accordance with D6.1, the results from the demonstrator development activity are shown.

Table 4-4. Demonstration results. M36 results are shown in italic.

ID	Goal	M18 and M36 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. M18 Development of the initial Data Storage System for efficient storage and retrieval of different type of information. *M36 Final Data Storage and Retrieval System with on-the-fly indexing of streamed data.* M18 Development of data fusion models for information retrieval based on combination of features, and for image enhancement purposes. *M36 Data fusion models for object detection and tracking.* M18 Development of the initial version of the physical prototype of Adaptive Camera System. Multiple cameras

		produce images of the quality of expensive cameras, thus reducing the cost. *M36 final version of adaptive camera: surface – wireless camera streaming to cloud and browser; subsea – ruggedized deep water with colour loss compensation.* 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. M18 incremental prototyping of the OM demonstrator based on the DynAA simulation models and the KPIs such as video processing throughput, java performance, battery performance, and storage performance. *M36 incremental prototyping of the OM demonstrator based on DynAA and AOW models and the KPIs such as video quality, throughput and response time tests for live streaming with different compression methods: H264, Mjpeg, and as a raw data.* 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 M18 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. *The M36 adaptation focuses on the environmental triggers. The demonstrator uses data from optical, temperature, pressure, and humidity sensors to adapt the image quality.*

4.3. Feedback

A number of challenges for runtime configuration were identified in the previous version of the deliverable, namely D6.9.

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 M36 demonstrator uses CERBERO methodology, technology, and tools in order to present a marine solution for ocean monitoring purposes. It incorporates an adaptive camera and the hub component, additional sensors, live streaming to cloud and browser, stream indexing, storage, and retrieval system, hybrid object detection and tracking, and new adaptation functionalities such as colour compensation for colour loss at different depths. Different aspects of the demonstrator have been successfully tested to prove the validity of the proposed solution in the context of CERBERO project.

6. References

[Abarca 2017]	Abarca, A., Xie, S., Markenhof, J., and Theuwissen, A. (2017). Temperature Sensors Integrated into a CMOS Image Sensor, In Proceedings of Eurosensors 2017, Paris, France, 3–6 September 2017.
[CERBERO 2018]	http://www.cerbero-h2020.eu
[IMAGECLEF 2018]	https://www.imageclef.org
[MIRFLICKR 2018]	https://press.liacs.nl/mirflickr/
[D1.3]	CERBERO D1.3 Open Data Management Plan (Final)
[D2.4]	CERBERO D2.4 Description of Scenarios (Final)
[D2.7]	CERBERO D2.7 Technical Requirements (Final)
[D3.4]	CERBERO D3.4 Modelling of KPI (Final)
[D6.7]	CERBERO D6.7 Demonstration Skeleton (Final)
[Zhang 2015]	Zhang, Xiang & Wang, Shiqi & Ma, Siwei & Gao, Wen. (2015). Towards Accurate and Efficient Image Quality Assessment with Interest Points. 164-170. 10.1109/BigMM.2015.58.
[Lin 2010]	Lin, D., Wang, C., and Wei, C. (2010). Quantified Temperature Effect in a CMOS Image Sensor, in IEEE Transactions on Electron Devices 57(2):422 – 428, IEEE Xplore, March 2010, DOI: 10.1109/TED.2009.2037389
[Lu 2014]	Lu, Wenjun & Li, Congli & Shi, Yongchang & Sun, Xiaoning. (2014). Image quality assessment based on SIFT and SSIM. Communications in Computer and Information Science. 437. 1-7. 10.1007/978-3-662-45498-5_1.
[Mohammadi 2014]	Mohammadi, P., Ebrahimi-Moghadam, A. and Shirani, S., 2014. Subjective and objective quality assessment of image: A survey. arXiv preprint arXiv:1406.7799.
[Oszust 2019]	Oszust, M., 2019. Local Feature Descriptor and Derivative Filters for Blind Image Quality Assessment. IEEE Signal Processing Letters, 26(2), pp.322-326.

7. Appendix: Some hardware consideration details regarding the OM use case

We researched and developed several camera prototypes using both Snapdragon, Intel Kaby Lake, and more Low-end Embedded Hardware boards. The various Snapdragon processors that we tested on were: Snapdragon 820, 821, 835, 845, 855, and 865. The Intel processors that we used were Intel Core i7/ Intel Kaby Lake, and the embedded hardware ranged from NVIDIA GPU development boards and down to quad core development boards. The various firmware we deployed on were Ubuntu LTS, Debian, Windows 10, and also Android version 7 to 10.

Below are the hardware tests for camera prototypes grouped according to type of hardware.

High-end hardware tests for camera: These included the use of Intel Kaby Lake and Snapdragon boards with CSI and USB data buses for both one and multiple lenses. We were able to stream and record both 4K and 8K footage using H.264 and HEVC compression at the same time as performing some live processing of the video stream. The results were positive and confirmed the choice of using both DynAA and AOW as tools for design time purposes, and also the tests confirmed the implemented Fusion Models.

Low-end embedded hardware tests for camera: These included testing of a range of different camera sensors for comparison purposes and to measure throughput and response time KPIs for live streaming to a video cloud service with different input video compression methods. These are H264, MJPEG, and raw data. We present some of these results in Table 8-1. This was tested using ffmpeg to transcode and reformat video data from these different devices, packing them to FLV formatted H.264 over RTMP for live relay through the prototype monitoring hub.

Table 8-1. Data throughput example streaming to cloud service

	H264	**Raw video**	**MJPEG**
UHD 920	N/A	Yes, 10 Fps	Yes, 15 Fps
Logitech	N/A	Yes, 14 Fps	Yes, 14 Fps
Spedal L920	N/A	Yes, 10 Fps	Yes, 15 Fps
Advent	N/A	Yes, 9 Fps	Yes, 9 Fps
Adaptive underwater camera – mini version	N/A	Yes, 9 Fps	Yes, 7 Fps

The H.264 entries reflect that almost no USB camera systems support H.264 at all, and many that do (notably Logitech ones) use a proprietary container format. The remaining performance measures show that encoding data throughput was the primary critical KPI

(matching the analysis performed using DynAA at M18). We did, however, take the platforms claims for their ability to encode H.264 too literally – in practice these all depended on directly connected cameras, typically using a CSI bus, with tight limits on the availability of sensors. This suggests there are multiple data bottlenecks, rather than just one.

Overcoming these issues, when it comes to low-end embedded hardware, remains a priority for future exploitation, but in the interim, we have adopted a workaround: bypassing video encoding data wherever possible, eliminating unnecessary decode-encode cycles, which have a high cost in both quality and data throughput. We have concluded that all tested camera sensors are capable of real time or near real time streaming to video cloud service and browser.