# D5.2 -

# INTEGRATION LAYER AND MULTIMODAL INDEXING OF HETEROGENEOUS DATA

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 883302.

## Deliverable Information

**Work Package:** WP5

**Deliverable Number:** D5.2

**Date of Issue:** 31/08/22

**Version Number:** v0.2

**Nature of Deliverable:** Report

**Dissemination Level:** PU

**Author(s):** Maria-Eirini Pegia (CERTH), Ilias Gialampoukidis (CERTH), Nikos Dourvas (CERTH), Konstantinos Ioannidis (CERTH)

**Keywords:** Information systems, Multimedia and multimodal retrieval, Supervised learning

**Abstract:** This deliverable discusses the Integration Layer and Multimodal Indexing service that gathers the heterogeneous data available in ISOLA, stores them and allows quick and efficient search to them. This service is connected to other services via Kafka. As far as heterogeneity is concerned, it is related to the use of different modality data, e.g., image, geo-location, time, etc.. The service uses a supervised image retrieval method for retrieving similar data, which is tested and validated against several datasets, and a friendly Graphical User Interface (GUI) for easily accessing and testing the retrieval system.

| Document History | | | |
|---|---|---|---|
| **Date** | **Version** | **Stage – remarks** | **Contributors** |
| 14/06/22 | 0.1 | ToC | Maria-Eirini Pegia (CERTH), Ilias Gialampoukidis (CERTH), Nikos Dourvas (CERTH), Konstantinos Ioannidis (CERTH) |
| 29/07/22 | 0.2 | First draft of the doc | Maria-Eirini Pegia (CERTH), Ilias Gialampoukidis (CERTH), Nikos Dourvas (CERTH) |
| 22/08/22 | 0.3 | Internal Review | George Alexopoulos (CLS), Sebastian Simonsen (PRO) |
| 31/08/22 | 1.0 | Final Draft | Maria-Eirini Pegia (CERTH), Ilias Gialampoukidis (CERTH), Nikos Dourvas (CERTH) |

**Disclosure Statement:** The information contained in this document is the property of the ISOLA consortium and it shall not be reproduced, disclosed, modified or communicated to any third parties without the prior written consent of the abovementioned entities.

# Executive Summary

This deliverable captures the progress realized within the context of the task 5.2 “Integration Layer and Multimodal Indexing” of the ISOLA project. The main focus of the task is the development of a service that allows efficient indexing and retrieval of heterogeneous information. Apart from that, proper connection to the ISOLA platform is realized which involves linking related input and output services.

Starting with the indexing and retrieval method that is being developed, an extensive study of the state-of-the-art methodologies on the domain of multimodal retrieval is realized. In particular, this study focuses on hashing methods, because they need less storage and are faster due to the use of hash-indexed data. Then, the novel BiasHash approach with a multimodal fusion approach is described. BiasHash is a supervised image retrieval method, which learns to project images to hash codes using a Bayesian-Ridge Regression framework. In addition, it incorporates a late fusion approach for combining hash codes for different modalities (including image, time and location) into one unified hash code, which enhances the multimodal search procedure.

In the sequel, the method is evaluated against three publicly available vessel datasets (MarDCT, SeaDronesSee and SeaShips) and against the ISOLA dataset. The evaluation on the public datasets involves experiments of the BiasHash method, and two other state-of-the-art methods on these datasets and for different code lengths by using commonly used performance metrics. Furthermore, experiments on ISOLA dataset are presented which involve visual evaluation of the produced results. From the experimental analysis it is shown that BiasHash outperforms the two state-of-the-art methods on all datasets. Finally, from the experimental analysis, it is shown that BiasHash outperforms the two state-of-the-art methods on all datasets.

Furthermore, the service is dockerised for reducing the effort and the risk of problems with application dependencies. Finally, to be connected to the ISOLA system, a presentation of the pipeline including the connection with other ISOLA services (like, face recognition service, object detection service and social media analysis service) is presented. The connection between services is done via the distributed messaging system Apache Kafka. Then, the data are indexed and stored into a non-relational DB, such as MongoDB, which is opted for flexibility reasons, as it can handle heterogeneous data. A detailed description of the record of each collection from MongoDB and the Kafka messages exchanged between services that concerns this service are included.

# Table of Contents


Executive Summary ..... 4
Table of Contents ..... 5
List of Tables ..... 7
List of Figures ..... 7
List of Acronyms ..... 10
1 Introduction ..... 12
2 Related work ..... 13
3 Multimodal Indexing Framework ..... 17
3.1 BiasHash method ..... 17
3.2 Methodology and ISOLA data ..... 21
4 Dataset ..... 22
4.1 ISOLA Dataset ..... 22
4.2 Publicly available datasets ..... 32
4.2.1 MarDCT dataset ..... 32
4.2.2 SeaDronesSee dataset ..... 34
4.2.3 SeaShips dataset ..... 37
5 Experiments ..... 39
5.1 Evaluation Metrics ..... 39
5.2 Parameters ..... 40
5.3 ISOLA Dataset ..... 40
5.3.1 Experiments ..... 40
5.3.2 Conclusions ..... 42
5.4 MarDCT dataset ..... 42
5.4.1 Experiments ..... 42
5.4.2 Conclusions ..... 45
5.5 SeaDronesSee dataset ..... 45
5.5.1 Experiments ..... 46
5.5.2 Conclusions ..... 51
5.6 SeaShips dataset ..... 51
5.6.1 Experiments ..... 51
5.6.2 Conclusions ..... 53
6 Integration of the Multimodal Indexing service to ISOLA ..... 53
7 Conclusions ..... 56

Acknowledgements ........................................................................................................56
References........................................................................................................................57

# List of Tables

Table 1. List of acronyms. ....................11

Table 2. The AP@k of BiasHash method for code lengths 16, 32, 64, 128 bit and for any combination of visual and temporal modality on MarDCT dataset ....................44

Table 3. The AP@k of BiasHash method for code lengths 16, 32, 64, 128 bit and for any combination of visual and temporal modality on MarDCT dataset ....................45

Table 4. The AP@k of BiasHash method for code lengths 16, 32, 64, 128 bit and for any combination of visual, temporal and spatial modality on SeaDronseSee dataset....................49

Table 5. The AP@k of BiasHash method for code lengths 16, 32, 64, 128 bit and for any combination of visual, temporal and spatial modality on SeaDronseSee dataset. ....................51

Table 6. The AP@k of BiasHash method for code lengths 16, 32, 64, 128 bit and for visual modality on SeaShips dataset ....................52

Table 7. The AP@k of BiasHash method for code lengths 16, 32, 64, 128 bit and for any combination of visual and temporal modality on SeaShips dataset....................53

# List of Figures

Figure 1. Project architecture based on the Integration Layer and Multimodal Indexing of Heterogeneous Data ....................13

Figure 2. Semantic Gap....................14

Figure 3. Curse of Dimensionality....................14

Figure 4. Supervised and Unsupervised Learning ....................15

Figure 5. Proposed framework for BiasHash ....................18

Figure 6. Example for construction of a temporal feature from a datetime. ....................22

Figure 7. Fields and values of the ACCELI_MISSION collection ....................23

Figure 8. An example of a document from the ACCELI_MISSION collection....................23

Figure 9. Fields and values of the CERTH_ACT collection....................24

Figure 10. An example of a document from the CERTH_ACT collection ....................25

Figure 11. . Fields and values of the CERTH_OBJ collection ....................26

Figure 12. An example of a document from the CERTH_OBJ collection ....................27

Figure 13. Fields and values of the CERTH_OBJ_QUERIES collection ....................28

Figure 14. An example of a document from CERTH_OBJ_QUERIES collection ....................28

Figure 15. Fields and values of the IDMG_FACE collection ....................29

Figure 16. An example of a document from the IDMG_FACE collection....................29

Figure 17. Fields and values of the OMST_UUV collection ....................30

Figure 18. An example of a document from the OMST_UUV collection....................30

Figure 19. Fields and values of the OMST_UUV_QUERIES collection....................31

Figure 20. An example of a document from the OMST_UUV_QUERIES collection .............31
Figure 21. An example of a document from the SIMAVI_MOBILE collection ........................31
Figure 22. Fields and values of the SIMAVI_MOBILE collection..........................................32
Figure 23. An example of a document from the SIMAVI_MOBILE collection ........................32
Figure 24. Some images from MarDCT dataset ..................................................................33
Figure 25. Histogram of classes for MarDCT dataset .........................................................33
Figure 26. Histogram of classes for query (left) and train (right) set for MarDCT dataset......33
Figure 27. Fields and values of the MarDCT collection........................................................34
Figure 28. . An example of a document from the MarDCT collection ..................................34
Figure 29. Some images from SeaDronesSee dataset.........................................................35
Figure 30. Histogram of classes for SeaDronesSee dataset.................................................35
Figure 31. Histogram of classes for query (left) and train (right) set for SeaDronesSee dataset ..............................................................................................................................35
Figure 32. Fields and values of the SeaDronesSee collection .............................................36
Figure 33. An example of a document from the SeaDronesSee collection...........................37
Figure 34. Some images from SeaShips dataset.................................................................37
Figure 35. Histogram of classes for SeaShips dataset.........................................................38
Figure 36. Histogram of classes for query (left) and train (right) set for SeaShips dataset ....38
Figure 37. Fields and values of the SeaShips collection ......................................................38
Figure 38. An example of a document from the SeaShips collection. ..................................39
Figure 39. Image example from CERTH_OBJ collection .....................................................40
Figure 40. Example of visual similarities for a given query from decision support service.....41
Figure 41. An image from OMST_UUV collection.................................................................41
Figure 42. An example of visual similarities of a query from decision support service for OMST_UUV data ..............................................................................................................41
Figure 43. An example of visual dissimilarities of a query from decision support service for OMST_UUV data ..............................................................................................................42
Figure 44. The first 10 results from BiasHash method for code lengths 16, 32, 64, 128 bits and for any combination of visual and temporal information on MarDCT dataset ........................43
Figure 45. The first 10 results from BiasHash, SSAH and FCMH methods for any combination of visual and temporal information on MarDCT dataset for 64 bit ........................................44
Figure 46. The first 10 results from BiasHash method for 16bits and for any combination of visual, temporal and spatial information in SeaDronesSee dataset for 128bit.......................46
Figure 47. The first 10 results from BiasHash method for 32bits and for any combination of visual, temporal and spatial information in SeaDronesSee dataset for 128bit.......................46
Figure 48. The first 10 results from BiasHash method for 64bits and for any combination of visual, temporal and spatial information in SeaDronesSee dataset for 128bit.......................47

Figure 49. The first 10 results from BiasHash method for 128bits and for any combination of visual, temporal and spatial information in SeaDronesSee dataset for 128bit .......................47

Figure 50. The first 10 results from BiasHash, SSAH and FCMH methods for any combination of visual and temporal information on SeaDronesSee dataset for 128bit .............................48

Figure 51. The first 10 results from BiasHash method for code lengths 16, 32, 64, 128 bits and for visual information on SeaShips dataset ..............................................................52

Figure 52. The first 10 results from BiasHash, SSAH and FCMH methods for any combination of visual and temporal information on SeaDronesSee dataset for 64bit ...............................53

Figure 53. Kafka architecture for Integration Layer and Multimodal Indexing service ............54

Figure 54. An example from a Kafka message, which the service can read from message bus ........................................................................................................................54

Figure 55. An example from a Kafka message, which the service can produce to message bus ........................................................................................................................55

Figure 56. Example with the overall architecture of the service ..........................................56

# List of Acronyms

| Acronym | Meaning |
|---|---|
| ACCELI_MISSION | Collection name of MongoDB for UAV from mission drone service and service's name for Kafka |
| AGAH | Adversary Guided Asymmetric Hashing |
| AP@k | Average Precision at k |
| BiasHash | Bayesian-Ridge Semantic Preserving Hashing |
| CENTRIC_DECISION | Decision support name for Kafka |
| CERTH_ACT | Collection name of MongoDB for data from activity recognition service and service's name for Kafka |
| CERTH_OBJ | Collection name of MongoDB for data from object detection service and service's name for Kafka |
| CSQ | Central Similarity Quantization |
| DADH | Deep Adversarial Discrete Hashing |
| DPH | Deep Priority Hashing |
| GUI | Graphical User Interface |
| IBM_DRONES | Collection name of MongoDB for UAV drones after modifications of object detection service and service's name for Kafka |
| IDMG_FACE | Collection name of MongoDB for face detection data and service's name for Kafka |
| FCMH | Fast Cross-Modal Hashing |
| FDMFH | Fast Discrete Matrix Factorization Hashing |
| GSPH | Generalized Semantic Preserving Hashing |
| HCOH | Hadamard Codebook based Online Hashing |
| KDLFH | Kernel-based Discrete Latent Factor Hashing |
| MFB | Multimodal Factorized Bilinear |
| MTFH | Matrix Tri-Factorization Hashing Framework |
| LAH | Label-Attended Hashing |
| LAGNH | Lightweight Augmented Graph Network Hashing |
| LCMH | Linear Cross-Modal Hashing |
| LSSH | Latent Semantic Sparse Hashing |
| OMST_UUV | Collection name of data from UUV service and service's name for Kafka |
| OMST_UUV | Collection name of data from queries of UUV data from decision support service and service's name for Kafka |
| PCDH | Pairwise Correlation Discrete Hashing |
| prec | Precision |
| prec@k | Precision at k |
| ReHash | Rank-embedded Hashing |
| RKHS | Reproducing Kernel Hilbert Space |
| SePH | Semantic Preserving Hashing |
| SIMAVI_MOBILE | Collection name of Twitter data from social media service and service's name for Kafka |
| SPAT | Spatial modality |
| SSAH | Self-Supervised Adversarial Hashing Network |
| STH | Self-taught Hashing |

| | |
|---|---|
| **TEMP** | Temporal modality |
| **UAV** | Unmanned Aerial Vehicle |
| **UUV** | Unmanned Underwater Vehicle |
| **VIS** | Visual modality |

**Table 1**. List of acronyms.

# 1 Introduction

The task "Integration Layer and Multimodal Indexing of Heterogeneous Data" (KR07) addresses efficient indexing and retrieval of heterogeneous data that are within ISOLA. The outcome of this task is a service that is responsible for consuming data from other services, processing them accordingly, indexing them and eventually allowing fast retrieval through a Graphical User Interface (GUI). The service uses a MongoDB database for storing the multimodal ISOLA data in JSON format. Depending on the source of information, as input we receive either images and their metadata (i.e., spatial and temporal information) or text and its metadata (i.e., temporal information), and as output we push the data to other ISOLA services. Figure 1 provides a general overview of the service, its position with regards to the ISOLA architecture, its connection with other services (the relevant tasks are also identified) and the flow of data to and from it.

Thus, the service may receive as input either of the following data:

- mobile and social data, data from passengers' moves without using any personal information and instead using the userID from social media service (see Figure 1, item "1");
- detected visual objects (like, skiff) originating from object detection service (see Figure 1, item "3");
- detected abnormal behaviour data (like, fighting) originating from the activity recognition service (see Figure 1, item "4");
- data from the bottom of the ship and from the bottom of the sea originating from the underwater vehicle service (see Figure 1, item "5");
- passenger data originating from the face recognition service (see Figure 1, item "2").

It should be noted that all the aforementioned input data are accompanied by metadata (e.g., datetime, spatial information and produced visual features) according to their type (See Figure 1, item "6").

The original data in case of text along with their original metadata and the produced metadata by KR07 are stored in a MongoDB (Figure 1, item "7"), while the original images/ videos are stored in a dedicated repository (Figure 1, item "8").

After that, the service provides according to the query, which can be either of the aforementioned inputs, different results (Figure 1, item "9"). The outcome of the service along with the initial information (Figure 1, item "10") is served to the Ontologies service and decision support service. The service is also responsible for inserting and updating the records in MongoDB. The data exchanges among the different services are realized through the Kafka's distributed system.

Regarding social media information the service detects if a specific alert keyword exists in social data, and if it exists then it produces an alert and sends it to the Ontologies service and decision support service. Specifically, it processes the TwitterID and the TwitterText for finding specific keywords from the predefined alert keywords (like, piracy and attack), while personal data are not part of the analysis. Finally, the service does not require to store this data for long-term use.

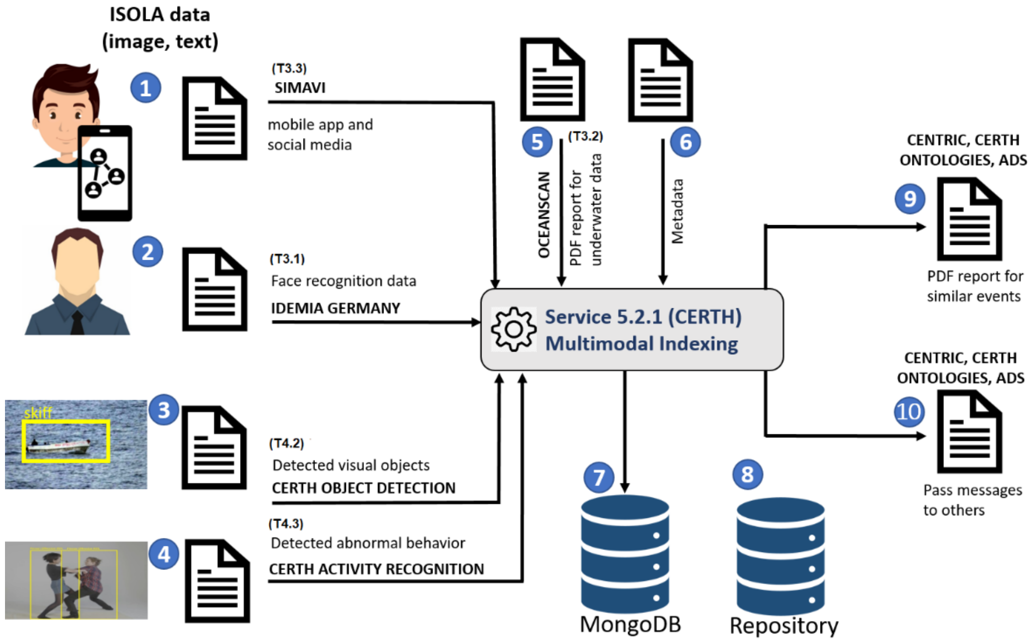


**Figure 1.** Project architecture based on the Integration Layer and Multimodal Indexing of Heterogeneous Data

This deliverable is structured as follows. Section 2 presents relevant recent works and Section **Error! Reference source not found.** gives details of the proposed multimodal retrieval method. The dataset and the experimental results are presented in Section 4 and Section 5, respectively. Section **Error! Reference source not found.** contains the framework of the service. The paper concludes with a brief summary in Section 7.

# 2 Related work

Multimodal retrieval is the field of study concerned with searching, browsing and retrieving multimedia data available in different contexts like text, image, audio and video from database (Xie2020). Due to the massive generation of multimedia data around the world, multimodal retrieval attracts interest among researchers from many fields, like image processing, multimedia search, and computer vision. The main challenges are: (a) the semantic gap between the low-level feature representing and high-level semantics in the images (Figure 2), and (b) the curse of dimensionality (Figure 3), since feature descriptors usually have hundreds or even thousands of dimensions. Hash-based indexes offer reduced storage, by storing only compact binary codes in the index, and constant average response time, thus making them ideal for addressing the indexing task within ISOLA. Therefore, this work focuses on hashing methods in order to use fast search through hash-indexed data instead of inefficient exhaustive search.

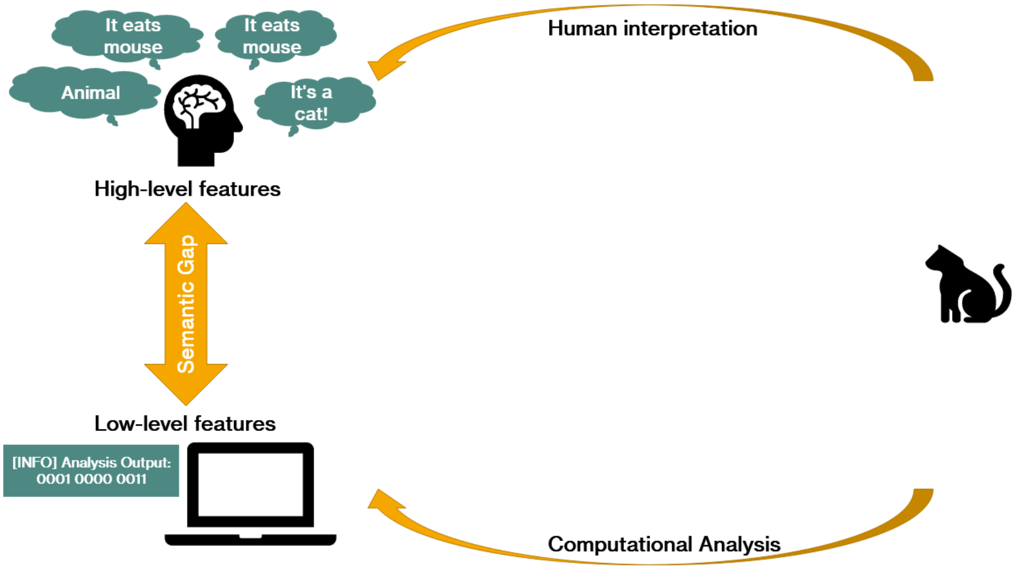


**Figure 2.** Semantic Gap

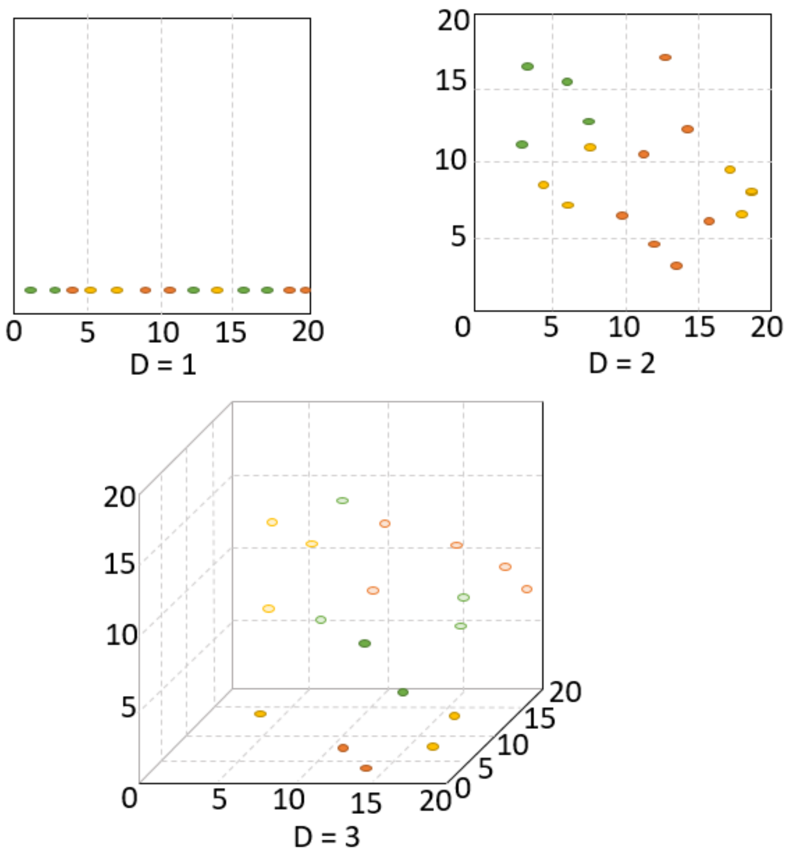


**Figure 3.** Curse of Dimensionality

Various hashing methods have been proposed for multimodal retrieval. Hashing approaches are categorized into single-view (Cao2018, Chen2020, Lin2014, Lin2018, Zhang2010, Zhen2016) and multi-view (Gu2019, Jiang2018, Li2018, Lin2015, Liu2019, Mandal2018, Zhou2014, Zhu2013). The former approaches use only one view, while the latter approaches importantly support many views/modalities (text, image, and video). Another categorization is based on the nature of the hashing functions used to generate the binary codes. Early approaches to hash-based indexing used manually-tuned hashing functions (e.g., Indyk1997), but more recent hashing approaches use either unsupervised learning (Zhen2016, Zhou2014, Zhu2013) or supervised learning (Cao2018, Chen2020, Gu2019, Jiang2018, Li2018, Lin2018, Lin2015, Liu2019, Mandal2018, Yuan2020) methods to generate the hash function, with the latter approach performing better.

In this section, some representative state-of-the-art unsupervised and supervised methods from the literature are selected. Figure 4 illustrates the basic procedure of a supervised and an unsupervised method. Specifically, the upper subfigure presents the procedure of learning to map annotated/labeled data to clusters and corresponds to supervised learning. In contrast, the bottom subfigure includes the procedure of learning the inner relationships of unlabeled data and grouping of them in clusters based on their inner structure of data.

Unsupervised hashing methods usually learn hash functions from data distribution in order to preserve the structures of training data. The Linear Cross-Modal Hashing (LCMH, Zhu2013) transforms each instance of training set into a k-dimensional approximation point (with k clustering) and maps the approximation points into Hamming space with the learnt hash functions, to match with the database binary codes. The Latent Semantic Sparse Hashing (LSSH, Zhou2014) learns latent semantic features for images and texts, respectively, with sparse coding and matrix factorization, and maps them to a joint abstraction space for generating unified hash codes. Finally, the SelfTaught Hashing (STH, Zhen2016) finds the optimal l-bit binary codes for all documents in the given corpus via unsupervised learning, and then trains l classifiers via supervised learning to predict the l-bit code for any query.

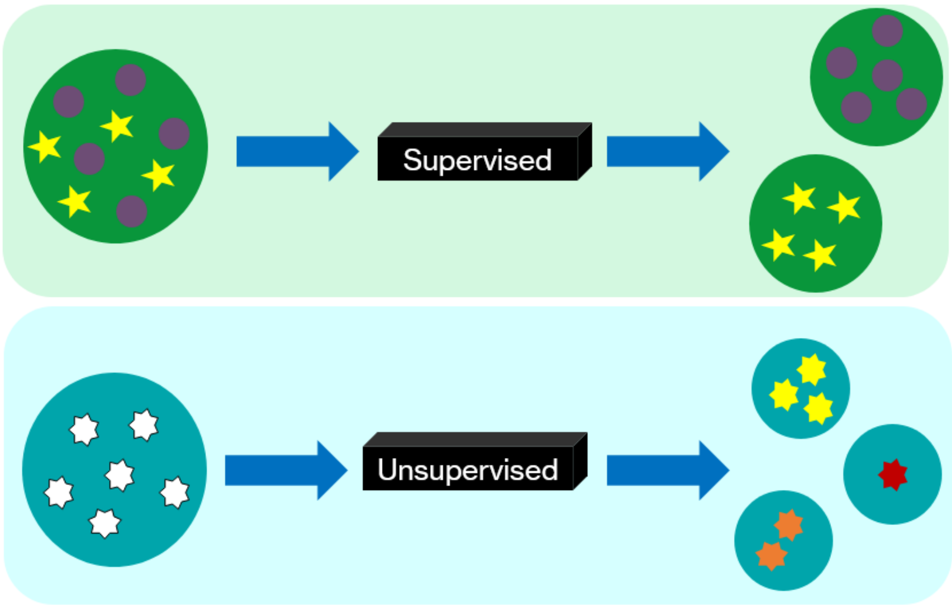


**Figure 4.** Supervised and Unsupervised Learning

Supervised methods, on the other hand, learn hash functions using supervised information. The supervised hashing methods can be splitted into three categories. The Error-free methods try to learn hash codes directly, while the transitive methods that uses transfer learning for learning compact hash codes, the Quantization methods and the Similarity matrix based methods use relaxation mechanism to learn hash codes (Lin2015). In particular, the

Quantization category uses quantization to obtain the final results, while the Similarity based category uses a matrix representation when learning hash functions.

There are hashing methods (Error-free methods) that try to direct find the hash codes without any relaxation procedure. The Kernel-based discrete latent factor model based cross-modal hashing (KDLFH, Jiang2018) is a discrete method which can directly learn the binary hash codes without continuous relaxation using a stochastic learning strategy. Furthermore, the Deep Priority Hashing (DPH, Cao2018) generates compact and balanced hash codes by jointly optimizing priority cross-entropy and quantization loss. Another method is the Rank-embedded Hashing (ReHash, Fu2020) that integrates the ranking metric into deep supervised hashing, which employs asymmetric supervision of deep learning for optimizing the compact codes projection. Moreover, the Matrix Tri-Factorization Hashing Framework (MTHF, Liu2019) aims to transfer knowledge from single-modal source domain to cross-modal target domain for promoting cross-modal retrieval. Chen, et al. proposed the Pairwise Correlation Discrete Hashing (PCDH, Chen2020), which uses the pairwise correlation of deep features and semantic information to generate discrete hashing codes.

There are also supervised methods that use adversarial learning or transfer knowledge. The Generalized Semantic Preserving Hashing (GSPH, Mandal2018) learns the optimum hash codes for the two modalities simultaneously, and then learns the hash functions to map from the features to the hash codes. Furthermore, the Adversary Guided Asymmetric Hashing (AGAH, Gu2019) uses adversarial learning guided multi-label attention mechanism to learn feature representation and generates binary codes from an asymmetric loss hash network. Finally, the Deep Adversarial Discrete Hashing (DADH, Bai2020) uses adversarial training for learning features across modalities and ensures the distribution consistency of feature representations across modalities.

Some methods try to solve the hard discrete optimization problem by relaxing the binary constraints and quantizing the solution to obtain the final results. Yuan, et al introduced the Central Similarity Quantization (CSQ, Yuan2020) that presents a global central similarity and encourages the hashing codes of similar images to arrive at the corresponding centers. Label-Attended Hashing (LAH, Xie2020) combines CNNs and Graph Convolution Network for generating image representation and label co-occurrence embeddings separately, adopts Multi-modal Factorized Bilinear (MFB) to fuse these vectors and learns the hash function with a loss function based on Cauchy distribution. Finally, the Fast Discrete Matrix Factorization Hashing (FDMFH, Zhao2021) utilizes matrix factorization to learn a latent semantic space and generates codes by rotating quantization and preserving with linear regression the original locality structure of training data. Moreover, the Lightweight Augmented Graph Network Hashing (LAGNH, Cui2021) extracts the inner structure of the image as the auxiliary semantics to enhance the semantic supervision of the unsupervised hash learning process

Other methods try to construct a simple similarity matrix when learning hash functions or binary codes. The Hadamard Codebook based Online Hashing (HCOH, Lin2018) utilizes a Hadamard matrix by minimizing the $l_2$ difference between hash-like output and the target hash codes with their corresponding labels (i.e., Hadamard loss). It trains the classification loss and Hadamard loss simultaneously. The Self-Supervised Adversarial Hashing Network (SSAH, Li2018) incorporates a self-supervised semantic network coupled with multi-label information, and carries out adversarial learning to maximize the semantic relevance and feature distribution consistency between different modalities. . In addition, the Fast cross-modal hashing (FCMH, Wang2021) takes both global and local similarities of data through global and local similarity embedding and solves the binary optimization problem by a well-designed group updating scheme. Finally, the Semantic Preserving Hashing (SePH, Lin2015) generates one unified hash code for all observed views of any instance. The Bayesian-Ridge-based Semantic

Preserving Hashing (BiasHash, Pegia2022) builds on top of the SePH method by using Bayesian regression as the utilized predictive model. The choice of SePH as baseline is based on the evaluation of training method is lower compared to other methods. In particular, the BiasHash uses the semantic probabilities of training data, approximates them with the learnt hash codes and then uses a Bayesian framework to learn these projection functions, motivated by the probability distribution that visual features tend to approximate.

Supervised methods perform better than the unsupervised methods in praxis. Apart from that, the supervised method BiasHash isn't greatly affected by the way the dataset is split and it outperforms the methods of the same category, as highlighted in the work of (Pegia2022). Only two methods were chosen, for comparison with BiasHash, as the most representative from the literature, SSAH (Li2018), FCMH (Wang2021) in terms of greater impartiality.

# 3 Multimodal Indexing Framework

## 3.1 BiasHash method

The approach that is incorporated in ISOLA is the Bayesian-Ridge-based Semantic Preserving Hashing (BiasHash, Pegia2022), which extends the Semantic Preserving Hashing (SePH, Lin2015). Figure 5 shows an overview of the SePH and the extensions proposed by the supervised hashing BiasHash framework. The light yellow boxes belong only to the BiasHash, while the other parts belong to both methods. Given that they are supervised methods, they both consist of two phases, the offline and the online phase. The offline phase corresponds to the training procedure, while the online phase corresponds to the testing procedure.

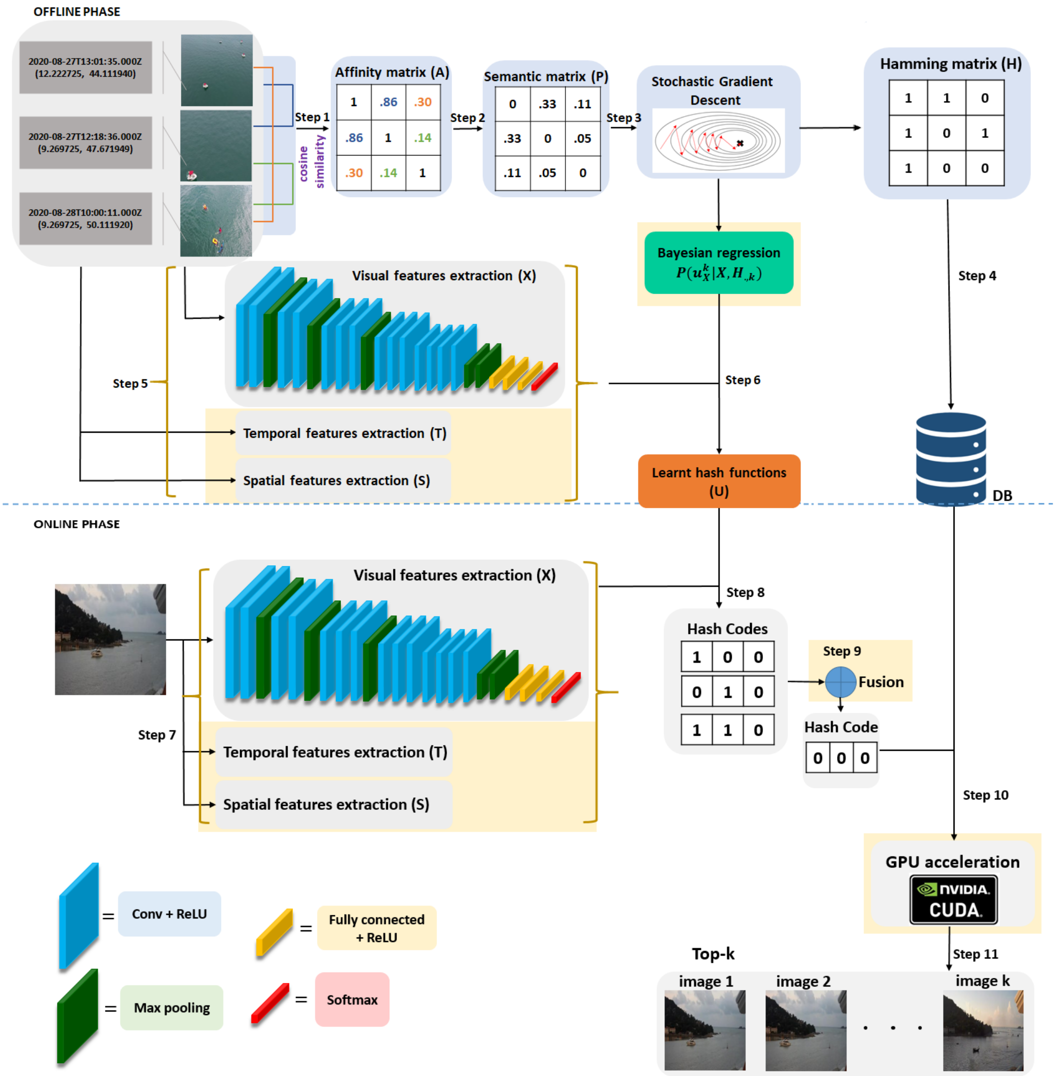


**Figure 5.** Proposed framework for BiasHash

In the offline phase, the affinity matrix is computed by training label vectors (Step1) and the semantic probabilities from affinity values (Step 2). Then, these values are projected to the learnt Hamming probabilities solving a minimization problem (Step 3). In the sequel, the Hamming vectors are stored in a database (Step 4). Then, the visual features are extracted from training images, the temporal features from the timestamps and the spatial features from the location (Step 5). Finally, the method learns the respective hash functions for each modality (Step 6) from each of the visual, temporal, spatial vectors to Hamming codes using Bayesian ridge regression.

In the online phase, for a given query, the approach extracts the visual feature (Step 7) and computes its Hamming code using the learnt hash functions (Step 8). After that, it combines the hash codes from different modalities using a fusion function and generates one unified hash code (Step 9). Finally it computes the Hamming distances between query and database codes in GPU (Step 10), ranks the results and returns the top k most relevant (Step 11).

In the sequel, more details on each step from the training phase are provided, starting with the notation.

- $O$ is the training set of size $|O| = n$, with $O_i$ its $i$ -th instance.
- $X, T, S$ corresponds to the visual, temporal and spatial features, respectively. Each of the aforementioned three real arrays has size $n \times d_x n \times d_t$, and $n \times d_s$, respectively.
- $L$ is the binary matrix of size $n \times l$ with $l$ the total number of labels, which denotes the semantic labels. For each instance $O_i$, the row $X_{i,.}$ , $T_{i,.}$, $S_{i,.}$and $L_{i,.}$ belongs to the visual, temporal, spatial and semantic feature vector, respectively.
- $A$ is the real matrix of size $n \times n$ that represents the affinity matrix of the training set.
- $H$ is a Binary matrix of size $n \times d_c$ that denotes the learnt hash codes of the training set of code length $d_c$. Each row of $H$ (i.e., $H_{i,.}$) corresponds to the projection of each semantic instance.
- $u_k^X$, $u_k^T$ and $u_k^S$ correspond to the learnt hash function of $k$-th bit, for $1 \leq k \leq d_c$ of visual, temporal and spatial modality, respectively.
- $u^F$ denotes the unified hash code, after the fusion of all available modalities.
- $h(.,.)$ is the Hamming distance between two hash codes of respective unified codes.
- $c^X$, $c^T$ and $c^S$ denote the hash codes of respective visual (X), temporal (T) and spatial (S) features, respectively and with $c_k^X$, $c_k^T$ and $c_k^S$ the $k$-th bit of hash code$c^x$, $c^t$ and $c^s$ of visual ($x$), temporal ($t$) and spatial ($s$) feature, respectively, and for $1 \leq k \leq d_c$.

The proposed method is built on SePH, thus the description of the SePH method is given initially and then the differences are outlined. Specifically, the SePH first computes the affinity matrix $A$ using the cosine similarity of corresponding vectors:

$$A = \frac{< L_{i,.} \, , \; L_{j,.} >}{\|L_{i,.}\| \|L_{j,.}\|}$$

**(1)**

Then the probabilities are computed as follows:

$$p_{i,j} = \frac{A_{i,j}}{\sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} A_{i,j}}$$

**(2)**

in semantic space $P$. The probabilities $q_{i,j}$ of instances in Hamming space $Q$ can be computed easily using the theorem that a Student t-distribution with one degree of freedom is utilised for transforming each pairwise Hamming distance into a probability. The theorem belongs to the work of van der Maaten and Hinton (Maaten2008) and the probabilities are:

$$q_{i,j} = \frac{\left(1 + h(H_{i,.}, H_{j,.})\right)^{-1}}{\sum_{k=1}^{n} \sum_{m=1, m \neq k}^{n} \left(1 + h(H_{k,.}, H_{m,.})\right)^{-1}}$$

**(3)**

The differences between $Q$ and $P$ can be measured using the Kullback-Leibler divergence. Therefore, the optimal hash code matrix $H$ of training set can be computed by minimizing it. Nevertheless, this minimisation problem belongs to the integer programming problems, which are NP-hard (Papadimitriou1981) and difficult to solve accurately. Therefore, the binary matrix $H$ is relaxed to the real valued matrix $\widehat{H}$ in Equation (4):

$$\Psi = \min_{\widehat{H}\in\mathbb{R}^{n\times d_c}} \sum_{i=1}^{n} \sum_{j=1,j\neq i}^{n} p_{i,j}\, log\frac{p_{i,j}}{q_{i,j}} + \frac{\alpha}{C}\left\|\left|\widehat{H} - I\right|\right\|_2^2$$

**(4)**

$\Psi$ is the minimization problem, $I \in \{1\}^{n\times d_c}$, $\alpha = 10^{-6}$ is a model parameter for weighting quantisation loss, $C = n \times d_c$ is a normalisation factor for tuning the parameter $\alpha$ for affecting less by the hash code length and the training size and

$$q_{i,j} = \frac{\left(1 + \left\|\widehat{H}_{i,.} - \widehat{H}_{j,.}\right\|_2^2\right)^{-1}}{\sum_{k=1}^{n}\sum_{m=1,m\neq k}^{n}\left(1 + \left\|\widehat{H}_{k,.} - \widehat{H}_{m,.}\right\|_2^2\right)^{-1}}$$

**(5)**

After that, the stochastic gradient descent (Ruder2017) is used for solving the unconstrained, non-convex optimisation problem (Equation (4)) and finding a locally optimal $\widehat{H}$. Then, the Hamming space matrix is computed:

$$H = sign(\widehat{H})$$

Then the kernel logistic regression is used for learning the hash function for visual modality, which projects the visual features to derived hash codes $H$. With learnt hash functions for indexing process, the hash codes of any unseen instance $z_u$ can be predicted.

During the online phase, and for each query the method computes its Hamming distance from each item in the retrieval set, sorts the retrieval set in ascending order based on this value and then chooses the top k items from the ordered set.

In particular, the k-th bit of the predictive visual hash code $c^X$ can be measured using kernel logistic regression

$$c_k^X = sign((\phi(X_{i,.}))\widehat{\Phi}^t)u^{(k)}$$

**(6)**

for $1 \leq k \leq d_c$ and $u^{(k)} \in \mathbb{R}^{d_x}$. $\Phi$ is the kernel feature matrix and $\phi(X_{i,.})$ is the transformation of visual feature $X_{i,.}$, in the Reproducing Kernel Hilbert Space (RKHS).

It should be noted that SePH uses Hamming distance to perform retrieval for query hash code $H_q$ from the retrieval hash codes:

$$h(H_q, H_i) = bit\_count(H_q \otimes H_i)$$

**(7)**

where the symbol $\otimes$ denotes the XOR operation between the bits of $H\_q$ and $H_i$, and $bit\_count$ sums the number of 1's in the binary result. Next, it ranks all the instances of the retrieval set based on their Hamming distances in an ascending order and chooses the top ones.

Similar with SePH, BiasHash computes the affinity matrix, and the semantic space matrix, as descripted above. However, BiasHash differs in the following points:

First, BiasHash uses a Bayesian Ridge Regression (Tipping2001) as predictive model for learning hash functions. Bayesian Ridge Regression is a linear model that uses probability distribution rather than point estimation of linear regression. The linear regression minimises loss, while the Bayesian version maximises the posterior probability by fitting a probabilistic model. This will give the model more flexibility on the way of splitting to training and testing set. Specifically, the linear model:

$$H_{.,k} = Z\, u_Z^k + \epsilon$$

$$\epsilon \sim N(O_n, \sigma^2 I_n)$$

$$H_{.,k} \in \{0,1\}^n$$

**(8)**

where $H_{.,k}$ is the k-th column of learnt hash codes $H$ and Z are the features for any modality (like, visual, temporal and spatial). It can be formulated as $H_{.,k} \sim N(Zu_Z^k, \sigma^2 I_n)$.

Second, BiasHash computes the posterior mean for each modality $Z$ using the iterative method of Tipping based on parameters updates used by MacKay (Tipping2001, Pedregosa2011) and set it to $u_Z^k$. After that it computes the hash codes for visual ($c^X$), temporal ($c^T$) and spatial ($c^S$) modality. Similar to SePH each bit $k = 1, \ldots, d_c$ of each hash code is computed by

$$c_k^Z = sign(Z_i, u_Z^{(k)})$$

**(9)**

Third, it fuses the hash codes using the fusion method:

$$c^F = (c^X \otimes c^T) \oplus (c^X \otimes c^S) \oplus (c^T \otimes c^S)$$

**(10)**

with $\oplus$ be the AND bitwise operator and $\otimes$ be the bitwise XOR operator.

In the testing phase the BiasHash extracts the visual, temporal and spatial feature for a given query. After that it uses the pretrained modality-specific hash functions to compute the corresponding hash code (Equation (9)). Next, it fuses the hash codes into one unified hash code (Equation (10)). Finally it computes the similarity scores with the indexed elements in the database.

## *3.2 Methodology and ISOLA data*

In this section, we focus on the application of BiasHash within the context of ISOLA and the different available modalities

Thus, as far as the visual representation of images is concerned, we produce the visual features of images using a pretrained VGG-16 with $d_x$ = 4096 for data from object detection service, activity recognition service and underwater vehicle service.

Regarding the spatial feature of a location = (longitude, latitude, altitude), this is computed as a 3D vector ($d_s = 3$).

Finally, as far as the temporal feature is concerned, the timestamp, which is captured in UTC (like, YYYY-MM-ddTHH: mm:ss.fffZ) is represented as a 203-D vector ($d_t = 203$). Specifically the first four coordinates of the temporal feature belongs to the 4 digits of the year, the next 12 digits to the one-hot-encoding for month, the next 31 digits to the one-hot encoding for day, the next 24 to the one-hot-encoding for hours, the next 60 to the one-hot encoding for minutes, the next 60 to the one-hot-encoding for seconds and the last 12 digits to microseconds. **Error! Reference source not found.** presents an example of the extraction of the temporal feature from a datetime.

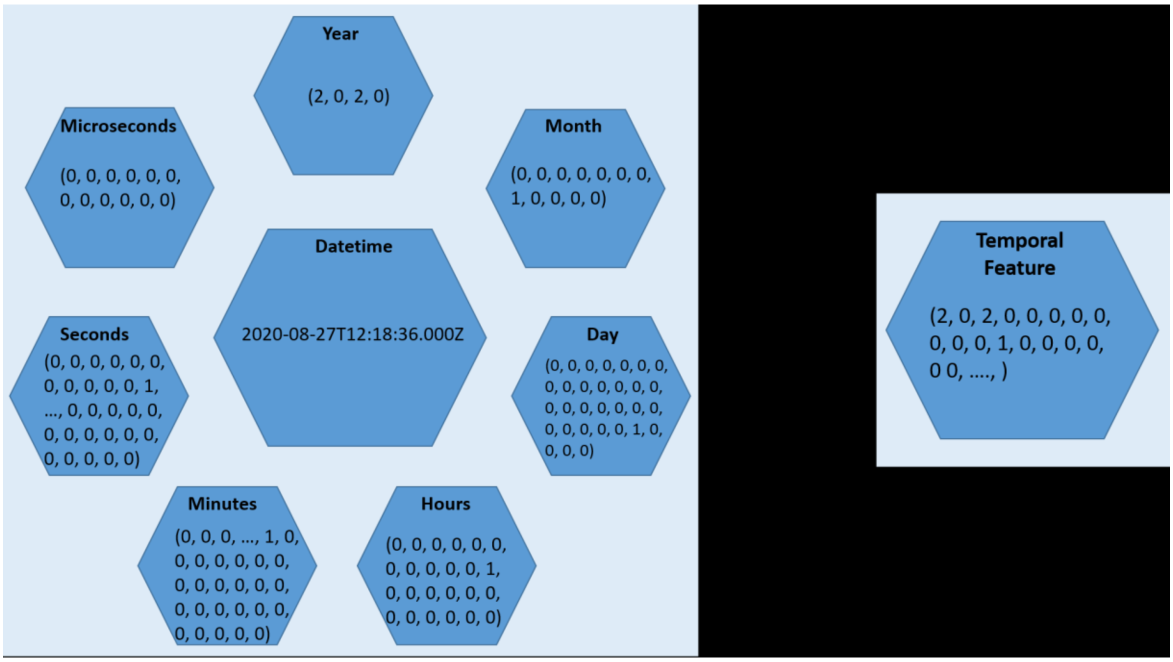


**Figure 6.** Example for construction of a temporal feature from a datetime.

# 4 Dataset

In this section, we present several datasets on which the BiasHash and other state-of-the-art methods are being evaluated in the context of ISOLA. Specifically, the methods are evaluated against the ISOLA dataset (i.e., data retrieved from other services), and several other datasets (i.e., MarDCT, SeaDronesSee, SeaShips).

## 4.1 ISOLA Dataset

The ISOLA dataset contains a plethora of different types of heterogeneous data. Specifically, there are eight collections, which are stored in the MongoDB database. The collections are ACCELI_MISSION, CERTH_ACT, CERTH_OBJ, CERTH_OBJ_QUERIES, IDMG_FACE,

OMST_UUV, OMST_UUV_QUERIES and SIMAVI_MOBILE. Some information concerning the data come from the messages of other partners through Kafka and others are produced from this service. The choice of these collections came from the data model with the collaboration of all partners. Apart from that the datetime and location fields help the service to perform faster queries in MongoDB. In addition ObjectId is a unique ID for each MongoDB document and is given by an insertion of a document to the MongoDB.

- The collection ACCELI_MISSION contains spatial and temporal information, when an object is detected by the tethered UAV drone. Figure 7 has an example of a document from this collection. It shows each field with its correspondent type of value. Apart from that an example of a document exists in Figure 8.

| | |
|---|---|
| _id | ObjectId |
| source | String |
| missionID | String |
| componentID | Int32 |
| TimeUTC | Date |
| resourceType | String |
| fps | Int32 |
| longitude | Double |
| latitude | Double |
| altitude | Double |
| location | Object |
| type | String |
| coordinates | Array |
| [0] | Double |
| [1] | Double |
| datetime | Date |

**Figure 7.** Fields and values of the ACCELI_MISSION collection


```
{
    "_id" :   ObjectId("6229c0b09375c0fe1d2561e4"),
    "source" : "ACCELI_MISSION",
    "missionID" : "MISSION_01",
    "componentID" : 0,
    "TimeUTC" :  ISODate("2022-03-10T09:11:10.000Z"),
    "resourceType" : "objectDetection",
    "fps" : 30,
    "longitude" : 12.3384704589844,
    "latitude" : 45.4341697692871,
    "altitude" : 15.2399997711182,
    "location" : {
        "type" : "Point",
        "coordinates" : [
            12.3384704589844,
            45.4341697692871
        ]
    },
    "datetime" : ISODate("2022-03-10T09:11:10.000Z")
}
```


**Figure 8.** An example of a document from the ACCELI_MISSION collection

- The collection CERTH_ACT contains the detected abnormal behaviours of passengers (like, running) on the ship with spatial and temporal information. Figure 9 and Figure 10 contains the fields with the type of values and an example of a document of this collection.

| | |
|---|---|
| _id | ObjectId |
| missionID | Int32 |
| componentID | Int32 |
| source | String |
| attachmentURL | String |
| height | Int32 |
| width | Int32 |
| fps | Int32 |
| longitude | Double |
| latitude | Double |
| altitude | Double |
| zoneID | String |
| zoneName | String |
| detectedActivities | Object |
| activityID | Int32 |
| startTime | Date |
| endTime | Date |
| activityName | String |
| confidence | Double |
| location | Object |
| type | String |
| coordinates | Array |
| [0] | Double |
| [1] | Double |
| datetime | Date |

**Figure 9.** Fields and values of the CERTH_ACT collection

```
{
    "_id" : ObjectId("623c1e9bc24d040d22a061d0"),
    "missionID" : 1,
    "componentID" : 2,
    "source" : "IBM_DRONE",
    "attachmentURL" : "2020-11-25-16-11-06.jpg",
    "height" : 450,
    "width" : 800,
    "fps" : 25,
    "longitude" : 39.1243951137686,
    "latitude" : 22.1243951137686,
    "altitude" : 23.1243951137686,
    "zoneID" : "ZONE ID 1",
    "zoneName" : "ZONE NAME 1",
    "detectedActivities" : {
        "activityID" : 3,
        "startTime" : ISODate("2020-01-01T09:00:00.010Z"),
        "endTime" : ISODate("2020-01-01T09:01:00.020Z"),
        "activityName" : "running",
        "confidence" : 0.900143784046173
    },
    "location" : {
        "type" : "Point",
        "coordinates" : [
            39.1243951137686,
            22.1243951137686
        ]
    },
    "datetime" : ISODate("2022-01-13T09:41:05.010Z")
}
```


**Figure 10.** An example of a document from the CERTH_ACT collection

- The collection CERTH_OBJ includes images with spatial and temporal information, when an object is detected in videos from a UAV. The source field may be ACCELI or IBM_DRONES and this field is provided by the CERTH_OBJ. An example of a document and its value exists in Figure 11 and in Figure 12. The service computes the metadata based on the inputs of CERTH_OBJ and inserts the document to the MongoDB. It contains six arrays. Specifically the first three corresponds to the visual, temporal and spatial feature of each image, while the later three to the hash codes of each of the aforementioned modalities (image, time, location). The visual, temporal and spatial feature arrays have length 4096, 203, 3 elements, respectively. The binary arrays have the same size and particularly each consists of 16 bits.

| | |
|---|---|
| _id | ObjectId |
| missionID | String |
| componentID | Int32 |
| source | String |
| height | Int32 |
| width | Int32 |
| fps | String |
| longitude | Double |
| latitude | Double |
| altitude | Double |
| location | Object |
| type | String |
| coordinates | Array |
| [0] | Double |
| [1] | Double |
| datetime | Date |
| zoneID | String |
| zoneName | String |
| sequence | Array |
| [0] | Object |
| frameID | Int32 |
| detection | Array |
| [0] | Object |
| objectID | Int32 |
| class | String |
| confidence | String |
| bbox | Object |
| x_min | String |
| y_min | String |
| x_max | String |
| y_max | String |
| metadata | Object |
| visual_feature | Array |
| temporal_feature | Array |
| spatial_feature | Array |
| visual_bin_vec | Array |
| temporal_bin_vec | Array |
| spatial_bin_vec | Array |
| videoURL | String |
| attachmentURL | String |
| fullpathURL | String |

**Figure 11. .** Fields and values of the CERTH_OBJ collection

```
{
    "_id" :  ObjectId("62b429848ad9561a4a9aca67"),
    "missionID" : "MISSION_01",
    "componentID" : 1,
    "source" : "ACCELI_MISSION",
    "height" : 720,
    "width" : 1280,
    "fps" : "0.040",
    "longitude" : 12.33847,
    "latitude" : 45.43417,
    "altitude" : 15.24,
    "location" : {
        "type" : "Point",
        "coordinates" : [
            12.33847,
            45.43417
        ]
    },
    "datetime" :  ISODate("2022-06-23T11:44:47.288Z"),
    "zoneID" : "1_303",
    "zoneName" : "Garage (Restricted)",
    "sequence" : [
        {
            "frameID" : 3,
            "detection" : [
                {
                    "objectID" : 1,
                    "class" : "person",
                    "confidence" : "0.9202678",
                    "bbox" : {
                        "x_min" : "573",
                        "y_min" : "177",
                        "x_max" : "595",
                        "y_max" : "241"
                    }
                }
            ]
        },
    ],
    "metadata" : {
        "visual_feature" : [
        "temporal_feature" : [
        "spatial_feature" : [
        "visual_bin_vec" : [
        "temporal_bin_vec" : [
        "spatial_bin_vec" : [
    },
    "videoURL" : "2020-11-25-16-11-06.mp4",
    "attachmentURL" : "UAV123_person4_10fps_fr0002.jpg",
    "fullpathURL" : "/home/mpegia/MyDatasets/ISOLA_DATA/CERTH_OBJ/UAV123_person4_10fps_fr0002.jpg"
}
```


**Figure 12.** An example of a document from the CERTH_OBJ collection

- The collection CERTH_OBJ_QUERIES contains the queries from Decision Support for the data of CERTH_OBJ for piracy incident scenario. The service performs queries that combine visual and spatial information and return the most similar images. Figure 13 and Figure 14 contain the fields and the value's types as well as an example of this collection.

| | |
|---|---|
| _id | ObjectId |
| source | String |
| reference | String |
| dataKey | String |
| parameters | Object |
| attachmentURL | String |
| xPos | String |
| yPos | String |
| posType | String |
| fullpathURL | String |
| location | Object |
| type | String |
| coordinates | Array |
| [0] | Double |
| [1] | Double |
| datetime | Date |

**Figure 13.** Fields and values of the CERTH_OBJ_QUERIES collection


```
{
    "_id" : ObjectId("6272177e29b0c403d5265c04"),
    "source" : "CENTRIC_DECISION",
    "reference" : "a42e8f203b",
    "dataKey" : "mmi-query",
    "parameters" : {
        "attachmentURL" : "UAV123_person4_10fps_fr0002.jpg",
        "xPos" : "12.542113",
        "yPos" : "45.543555",
        "posType" : "internal",
        "fullpathURL" : "/home/mpegia/MyDatasets/ISOLA_DATA/CERTH_OBJ_QUERIES/UAV123_person4_10fps_fr0002.jpg"
    },
    "location" : {
        "type" : "Point",
        "coordinates" : [
            12.543555,
            45.542113
        ]
    },
    "datetime" : ISODate("2022-05-04T12:00:00.000Z")
}
```


**Figure 14.** An example of a document from CERTH_OBJ_QUERIES collection

- The collection IDMG_FACE includes the information of each passenger of the ship and is provided by the face detection service. Figure 15 presents the fields and the value of each field for this collection. An example of a document exists in Figure 16.

| | |
|---|---|
| _id | ObjectId |
| source | String |
| cameraID | Int32 |
| ip | String |
| timeUTC | Date |
| eventID | String |
| eventType | String |
| watchlistName | String |
| score | Double |
| modality | String |
| probeImageURL | String |
| candidateImageURL | String |
| contextImageURL | String |
| firstName | String |
| lastName | String |
| memberOF | String |
| text | String |
| datetime | Date |

**Figure 15.** Fields and values of the IDMG_FACE collection


```
{
    "_id" : ObjectId("61dfcb9461ad8ac9d67b3a09")
    "source" : "IDMG_FACE",
    "cameraID" : 1,
    "ip" : "160.53.21.12",
    "timeUTC" : ISODate("2020-01-01T11:00:00.000Z"),
    "eventID" : "gwrhwa",
    "eventType" : "edgarheah",
    "watchlistName" : "grwehw",
    "score" : 42.123,
    "modality" : "grehwe",
    "probeImageURL" : "gwehgw.jpg",
    "candidateImageURL" : "fwegqw.jpg",
    "contextImageURL" : "gweghw.jpg",
    "firstName" : "gewhwal",
    "lastName" : "gwegl",
    "memberOF" : "gwgea",
    "text" : "gwhgewrhte",
    "datetime" : ISODate("2020-01-01T12:00:00.000Z")
}
```


**Figure 16.** An example of a document from the IDMG_FACE collection

- The collection OMST_UUV includes the data from UUV service. Figure 17 presents the fields and the value of each field for this collection. An example of a document exists in Figure 18.

| | |
|---|---|
| _id | ObjectId |
| source | String |
| planID | String |
| startTime | Date |
| endTime | Date |
| datetime | Date |
| attachments | Array |
| [0] | Object |
| sensor | String |
| Content-Type | String |
| atttachmentURL | String |
| framePathURL | String |
| metadata | Array |
| [0] | Object |
| visual_feature | Array |
| fullpathURL | String |

**Figure 17.** Fields and values of the OMST_UUV collection


```
{
    "_id" : ObjectId("62cd56dd5c522206fcb38823"),
    "source" : "OMST_UUV",
    "planID" : "isola-survey-May",
    "startTime" : ISODate("2022-05-24T13:10:07.000Z"),
    "endTime" : ISODate("2022-05-24T13:10:07.230Z"),
    "datetime" : ISODate("2022-05-24T13:10:07.000Z"),
    "attachments" : [
        {
            "sensor" : "Camera",
            "Content-Type" : "video/mp4",
            "atttachmentURL" : "vlc-record-2022-05-24-13h14m58s-GX021572.MP4-.mp4",
            "framePathURL" : "/home/mpegia/MyDatasets/ISOLA_DATA/OMST_UUV/videos/vlc-record-2022-05-24-13h14m58s-GX021572.MP4-.mp4"
        },
        {
            "sensor" : "Camera",
            "Content-Type" : "image/jpg",
            "attachmentURL" : "vlc-record-2022-05-24-13h14m58s-GX021572_0001.jpg",
            "fullpathURL" : "/home/mpegia/MyDatasets/ISOLA_DATA/OMST_UUV/images/vlc-record-2022-05-24-13h14m58s-GX021572_0001.jpg"
        },
    ],
    "metadata" : [
        {
            "visual_feature" : [
            "fullpathURL" : "/home/mpegia/MyDatasets/ISOLA_DATA/OMST_UUV/images/vlc-record-2022-05-24-13h14m58s-GX021572_0001.jpg"
        }
    ]
}
```


**Figure 18.** An example of a document from the OMST_UUV collection

- The collection OMST_UUV_QUERIES contains the queries made by the Decision Support service to CERTH_MULTI requesting data from the UUV service. It has the URL of an image from the bottom of the vessel and the service finds and returns the most visual similar to the query images. Figure 19 presents the fields and values of a document of this collection, while Figure 20 has an example of a document.

| | |
|---|---|
| _id | ObjectId |
| source | String |
| planID | String |
| startTime | Date |
| endTime | Date |
| attachments | Array |
| [0] | Object |
| sensor | String |
| Content-Type | String |
| attachmentURL | String |
| msgType | String |
| datetime | Date |

**Figure 19.** Fields and values of the OMST_UUV_QUERIES collection


```
{
    "_id" : ObjectId("62d16a940b5899760c78c28a"),
    "source" : "OMST_UUV",
    "planID" : "PathPlanner",
    "startTime" : ISODate("2022-05-04T12:00:00.000Z"),
    "endTime" : ISODate("2022-05-04T12:30:00.000Z"),
    "attachments" : [
        {
            "sensor" : "sidescan",
            "Content-Type" : "image/png",
            "attachmentURL" : "/home/mpegia/MyDatasets/ISOLA_DATA/OMST_UUV/images/vlc-record-2022-05-24-13h10m07s-GX011578_0001.jpg"
        }
    ],
    "msgType" : "queryRequest",
    "datetime" : ISODate("2022-05-04T12:00:00.000Z")
}
```


**Figure 20.** An example of a document from the OMST_UUV_QUERIES collection

- The collection SIMAVI_MOBILE contains the social data of each passenger's mobile and the data is provided by the mobile and social media service. CERTH_MULTI and SIMAVI decide specific predefined keywords that correspond to alerts (Figure 21). If any of this alert keywords occur in a Twitter message, then the CERTH_MULTI adds a new field to the document alert with value True. This field emphasises that something suspicious exists in social data and the intelligent reporting should inform about it. The fields and the value of each field for this collection are given in Figure 22. Figure 23 presents an example of a document.


```
{
    "keywords": [
        "piracy", "attack", "fire", "running", "accident", "incident", "stealing", "steal", "robbery",
        "robber", "theft", "thief", "scream", "shout", "screaming", "shouting", "yell", "yelling",
        "suspect", "suspicious", "suspicion", "illegal", "fake"
    ]
}
```


**Figure 21.** An example of a document from the SIMAVI_MOBILE collection

| | |
|---|---|
| _id | ObjectId |
| source | String |
| userID | String |
| username | String |
| userScreenName | String |
| profileImageURL | String |
| profileCreatedAt | Date |
| datetime | Date |
| tweets | Array |
| [0] | Object |
| tweetID | String |
| tweetText | String |
| tweetCreateAt | Date |
| tweetHashTags | String |

**Figure 22.** Fields and values of the SIMAVI_MOBILE collection


```
{
    "_id" : ObjectId("62cd48ec0935df9e92e912f2"),
    "source" : "SIMAVI_MOBILE",
    "userID" : "1502239083225092099",
    "username" : "jeen-yuhs",
    "userScreenName" : "avsdasaint",
    "profileImageURL" : "http://pbs.twimg.com/profile_images/1545275313680269312/FaS7kHH__normal.jpg",
    "profileCreatedAt" : ISODate("2022-03-11T09:05:05.000Z"),
    "datetime" : ISODate("2022-07-12T10:11:55.000Z"),
    "tweets" : [
        {
            "tweetID" : "1546799123877314560",
            "tweetText" : "RT @OrdinaryGamers: Piracy is always morally okay.",
            "tweetCreateAt" :ISODate("2022-07-12T07:10:23.000Z"),
            "tweetHashTags" : ""
        }
    ]
}
```


**Figure 23.** An example of a document from the SIMAVI_MOBILE collection

## 4.2 Publicly available datasets

Given that ISOLA focuses on identifying threats or abnormal activities related to vessels that may be detected by UAVs or UUVs or social media, the experiments were focused on vessel related datasets. Thus, the following three publicly available vessel datasets are used in the experiments, MarDCT (Bloisi2015), SeaDronesSee (Varga2022) and SeaShips (Shao2018). More information for each dataset can be found in Section 4.2.1, Section 4.2.2 and Section 4.2.3, respectively. Similar to the ISOLA datasets, these collections are also stored in the MongoDB and examples of document for each collection are explained in the following subsections.

### 4.2.1 MarDCT dataset

MarDCT is a vessel dataset, which consists of 6743 annotated data with temporal available information. Some images from MarDCT dataset exist in Figure 24. There is a proposed way for splitting the dataset according to (Bloisi2015). Specifically, query/training/validation set correspond to the values 1969/1064/4774. Figure 25 presents the data for each class and Figure 26 the data of each class for query and train set.

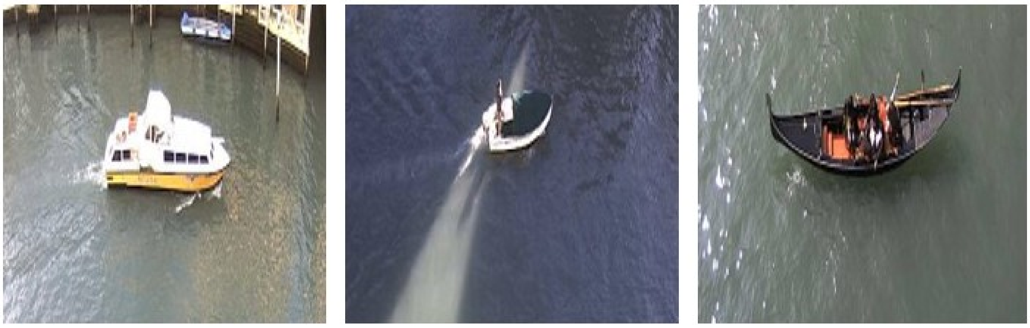

**Figure 24.** Some images from MarDCT dataset

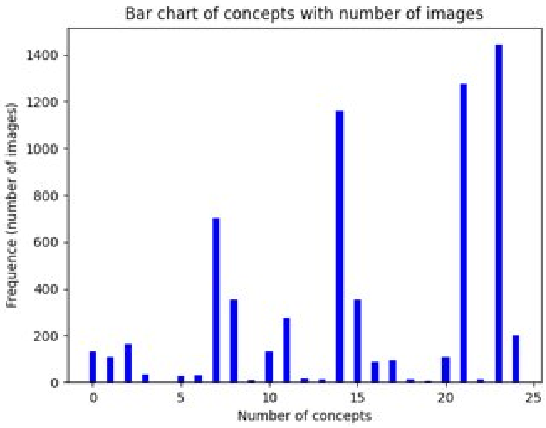


**Figure 25.** Histogram of classes for MarDCT dataset

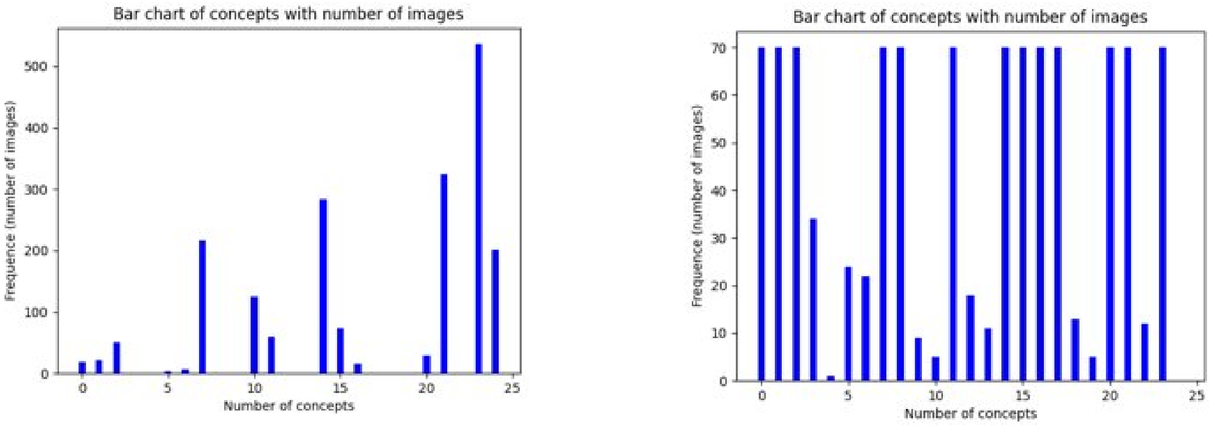


**Figure 26.** Histogram of classes for query (left) and train (right) set for MarDCT dataset

The collection MarDCT corresponds to the dataset MarDCT with associated metadata. Figure 27 presents the fields and values of a document, while Figure 28 presents an example of a document from this collection. The metadata field consists of the computed features from the service.

| | |
|---|---|
| _id | ObjectId |
| datetime | Date |
| datetime_mp | Array |
| [0] | Int32 |
| [1] | Int32 |
| [2] | Int32 |
| [3] | Int32 |
| [4] | Int32 |
| [5] | Int32 |
| category | String |
| category_id | Int32 |
| metadata | Object |
| visual_feature | Array |
| temporal_feature | Array |
| visual_bin_vec | Array |
| temporal_bin_vec | Array |
| attachmentURL | String |
| fullpathURL | String |

**Figure 27.** Fields and values of the MarDCT collection


```
{
    "_id" : ObjectId("62384c66325bc9fcb6eb848b"),
    "datetime" : ISODate("2013-03-05T11:06:11.000Z"),
    "datetime_mp" : [
        2013,
        3,
        5,
        11,
        6,
        11
    ],
    "category" : "Lanciafino10mMarrone",
    "category_id" : 2,
    "metadata" : {
        "visual_feature" : [
        "temporal_feature" : [
        "visual_bin_vec" : [
        "temporal_bin_vec" : [
    },
    "attachmentURL" : "20130305_110611_62553.jpg",
    "fullpathURL" : "/home/mpegia/MyDatasets/ISOLA_DATA/MarDCT/Lanciafino10mMarrone/20130305_110611_62553.jpg"
}
```


**Figure 28. .** An example of a document from the MarDCT collection

### 4.2.2 SeaDronesSee dataset

SeaDronesSee dataset consists of 5630 vessel annotated images with available temporal and spatial information. Images from SeaDronesSee dataset are presented in Figure 29. The dataset is proposed to be split into 1796 query, 2975 training and 859 validation data by the documentation (Varga2022). Figure 30 contains the data for each class and Figure 31 the data of each class for query and train set.

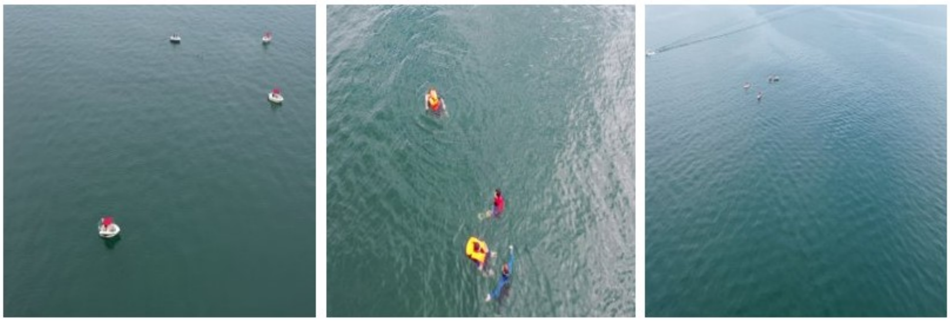

**Figure 29.** Some images from SeaDronesSee dataset

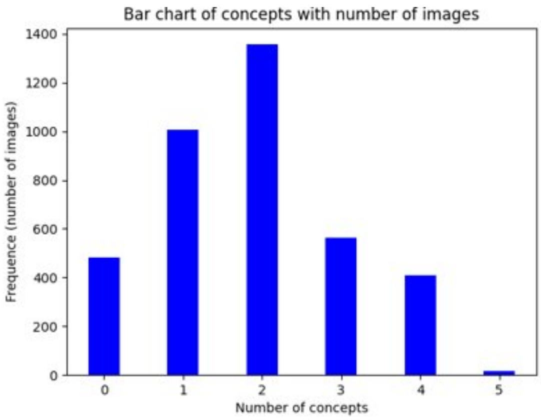


**Figure 30.** Histogram of classes for SeaDronesSee dataset

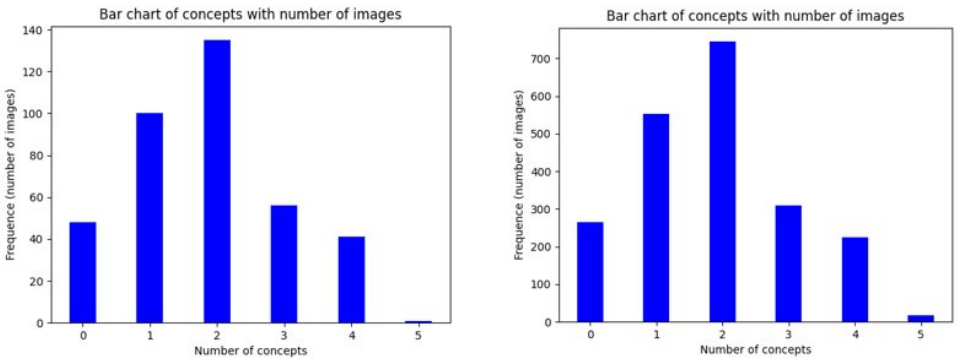


**Figure 31.** Histogram of classes for query (left) and train (right) set for SeaDronesSee dataset

The collection SeaDronesSee is related to the dataset SeaDronesSee with additionally metadata. Figure 32 includes the fields and values of a document, while Figure 33 illustrates an example of a document from this collection.

| | |
|---|---|
| _id | ObjectId |
| datetime | Date |
| datetime_mp | Array |
| [0] | Int32 |
| [1] | Int32 |
| [2] | Int32 |
| [3] | Int32 |
| [4] | Int32 |
| [5] | Int32 |
| location | Object |
| type | String |
| coordinates | Array |
| [0] | Double |
| [1] | Double |
| location_mp | Array |
| [0] | Double |
| [1] | Double |
| [2] | Double |
| category | String |
| category_id | Int32 |
| metadata | Object |
| visual_feature | Array |
| temporal_feature | Array |
| spatial_feature | Array |
| visual_bin_vec | Array |
| temporal_bin_vec | Array |
| spatial_bin_vec | Array |
| attachmentURL | String |
| fullpathURL | String |

**Figure 32.** Fields and values of the SeaDronesSee collection

```
{
    "_id" : ObjectId("62384ea4130cf34650310a35"),
    "datetime" : ISODate("2020-08-27T12:18:36.000Z"),
    "datetime_mp" : [
        2020,
        8,
        27,
        12,
        18,
        36
    ],
    "location" : {
        "type" : "Point",
        "coordinates" : [
            9.269725,
            47.671949
        ]
    },
    "location_mp" : [
        9.269725,
        47.671949,
        8.59958061566596
    ],
    "category" : "None",
    "category_id" : -1,
    "metadata" : {
        "visual_feature" : [
        "temporal_feature" : [
        "spatial_feature" : [
        "visual_bin_vec" : [
        "temporal_bin_vec" : [
        "spatial_bin_vec" : [
    },
    "attachmentURL" : "2115.png",
    "fullpathURL" : "/home/mpegia/MyDatasets/ISOLA_DATA/SeaDronesSee/images/2115.png"
}
```


**Figure 33.** An example of a document from the SeaDronesSee collection

### 4.2.3 SeaShips dataset

The dataset SeaShips includes 7000 annotated images. Specifically it contains five classes (like, cargo ship and passenger ship), each corresponding to a different vessel type. Figure 34 presents some images from the dataset. Figure 35 contains the data for each class and Figure 36 the data of each class for the query and train sets respectively.


**Figure 34.** Some images from SeaShips dataset

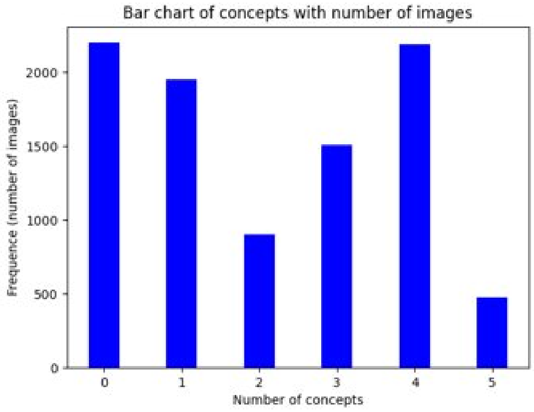


**Figure 35.** Histogram of classes for SeaShips dataset

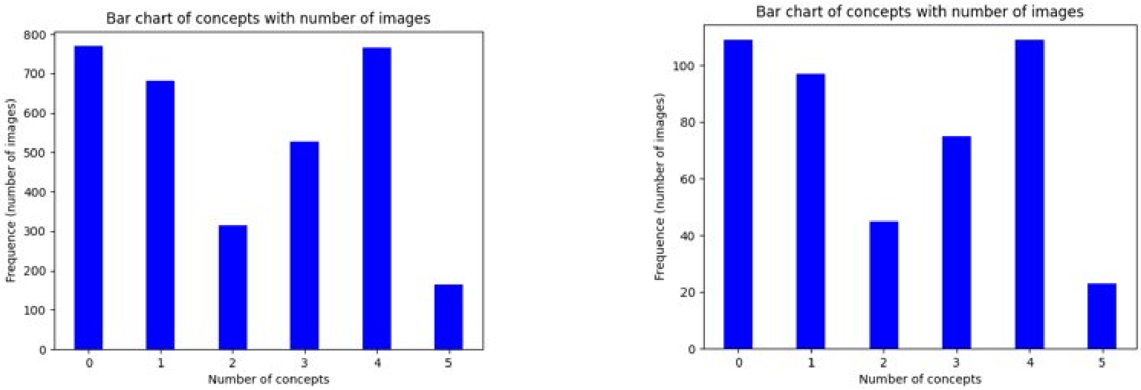


**Figure 36.** Histogram of classes for query (left) and train (right) set for SeaShips dataset

The collection SeaShips corresponds to the dataset SeaShips with associated metadata. Figure 37 presents the fields and values of a document and Figure 38 contains an example of a document from this collection.

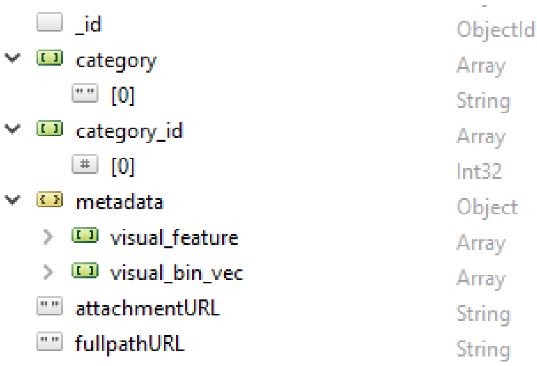


**Figure 37.** Fields and values of the SeaShips collection

```
{
    "_id" : ObjectId("623b039bc7c85de91bbca666"),
    "category" : [
        "ore carrier"
    ],
    "category_id" : [
        0
    ],
    "metadata" : {
        "visual feature" : [
        "visual bin vec" : [
    },
    "attachmentURL" : "000001.jpg",
    "fullpathURL" : "/home/mpegia/MyDatasets/ISOLA_DATA/SeaShips/JPEGImages/000001.jpg"
}
```


**Figure 38.** An example of a document from the SeaShips collection.

# 5 Experiments

This section contains the performance measures, the parameters' values and the experimental results that involves the comparison of the BiasHash approach against other methods and against several datasets. From the literature only two methods were chosen. Specifically, the methods against which the BiasHash is validated is SSAH (Li2018), FCMH (Wang2021). Furthermore the other methods of the same category perform worst as highlighted in the work of Pegia2022. BiasHash belongs to the same supervised hashing category as the other two methods and therefore the comparison was made in the same category for greater impartiality.

## *5.1 Evaluation Metrics*

The Precision (prec) is the fraction of the relevant results in the total number of results.

$$prec = \frac{Number\ of\ relevant\ results}{total\ number\ of\ results}$$

**(11)**

The Precision at k (prec@k) is the fraction of relevant items in the top k recommended results.

$$prec@k = \frac{Number\ of\ relevant\ items\ in\ topk\ results}{k}$$

**(12)**

The average precision at k (AP@k) is the sum of precision at k for different values of k divided by the total number of relevant items in the top k results.

$$AP@k = \sum_{i=1}^{k} \frac{1}{r_i} \frac{(Number\ of\ relevant\ items\ in\ top\ i\ results) \times rel(i)}{i}$$

$$rel(i) = \begin{cases} 1, & if\ i-th\ item\ is\ relevant \\ 0, & otherwise \end{cases}$$

**(13)**

## 5.2 Parameters

BiasHash trained VGG-16 to the three public vessel dataset (MarDCT, SeaDronesSee and SeaShips) and used the best model for visual feature extraction and learnt visual hash functions. Different training sizes are used for each vessel dataset. More information can be found in Section 5.4, Section 5.5 and Section 5.6. Similarly, the hash function for temporal and spatial modality are used from the training set of the public vessel dataset.

## 5.3 ISOLA Dataset

This section includes some experiments of the service of the ISOLA dataset. We give examples from the decision support service, the object detection service and the UUV service. In particular, the queries of decision support service about UUV and UAV are presented in Section 5.3.1. In the case of UAV, the method uses the pretrained VGG-16 for visual feature extraction and hash code computation, because the model is trained to vessel dataset. However, in the case of UUV, there is not an available public dataset about vessels. Therefore, the service computes only the differences in feature level between images and ranks them in ascending order without using the BiasHash method.

### 5.3.1 Experiments

An example from CERTH_OBJ collection is depicted in Figure 39.

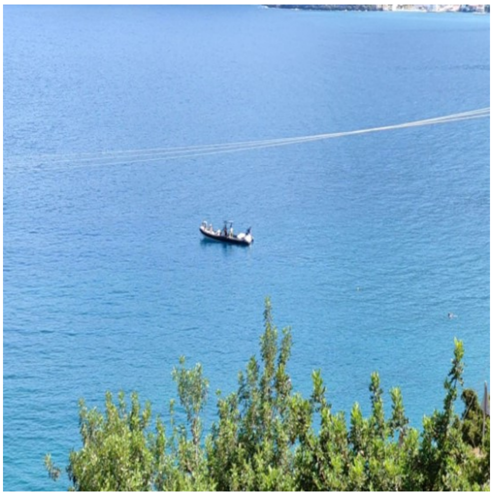

**Figure 39.** Image example from CERTH_OBJ collection

An example of decision support query for visual similarities related to the object detection service data is presented in Figure 40 along with its retrieved results.

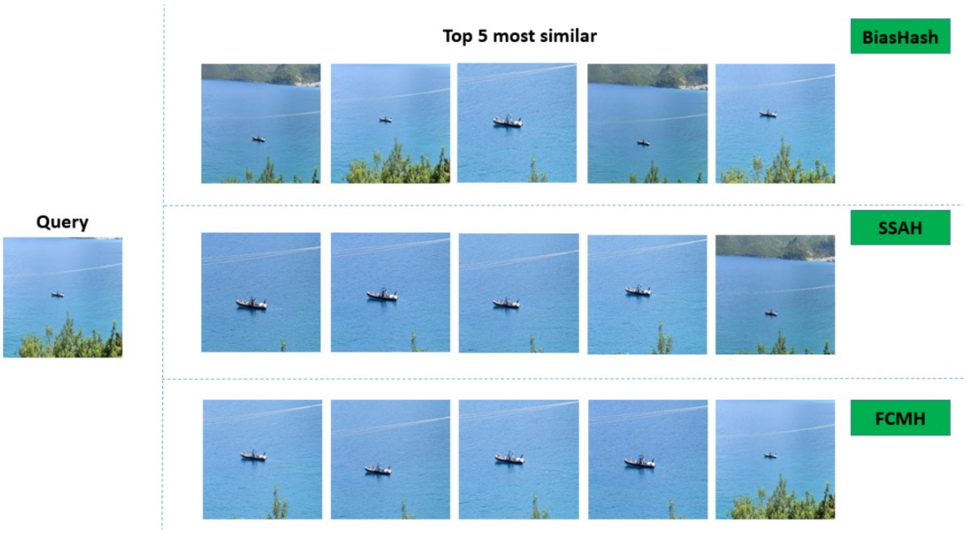


**Figure 40.** Example of visual similarities for a given query from decision support service

The OMST_UUV dataset contains the UUV from UUV service. An example of an image from UUV and an example of a query from Decision Support for finding visual similarities is shown in Figure 41 and Figure 42. Figure 43 presents a scenario of finding the most dissimilar images for a given query.

**Figure 41.** An image from OMST_UUV collection

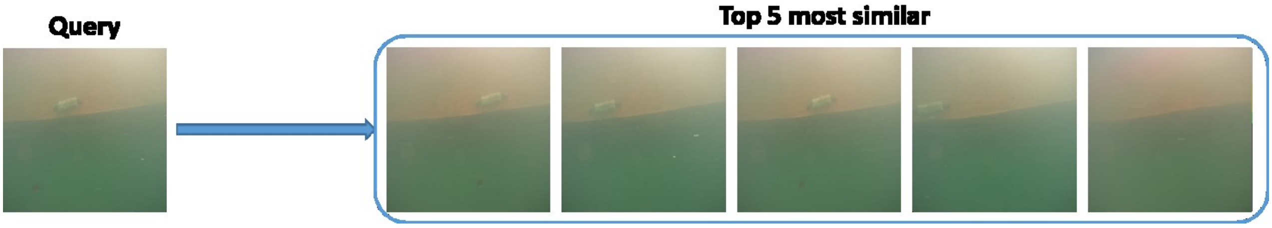


**Figure 42.** An example of visual similarities of a query from decision support service for OMST_UUV data


**Figure 43.** An example of visual dissimilarities of a query from decision support service for OMST_UUV data

### 5.3.2 Conclusions

From the experiments in the previous subsection (Section 5.3.1), the service can detect similarities for UAV and UUV data. The scenario of finding the most dissimilar image from a mission of UUV service could be helpful to limit the number of queries to the service. Apart from that, from these first experiments the BiasHash can perform quite well uni-modal (visual) and multimodal (more than one modalities, like visual and spatial) queries in contrast to the SSAH (Li2018), FCMH (Wang2021) methods.

## *5.4 MarDCT dataset*

This section contains some quality and visual results for the MarDCT dataset. In particular, Figure 44 illustrates some visual results of the retrieved results of BiasHash for different code lengths and modalities. Furthermore experiments comparing the state-of-the-art methods in MarDCT is depicted in Figure 45. Finally, Table 2 and Table 3 present some quality results in terms of AP@k, k = 20, 50, 100, 200, 300 using Equation (13), for BiasHash and for SSAH (Li2018), FCMH (Wang2021) methods.

### 5.4.1 Experiments

Figure 44 illustrates the first ten results of BiasHash for any combination of visual and temporal modality and for hash code lengths 16, 32, 64, 128 bit in MarDCT. BiasHash performs better in visual modality and in the combination of two modalities. Table 2 presents the performance of BiasHash in terms of AP@k for different code lengths and for k = 30, 50, 100, 200, 300. The performance of BiasHash increases as the code lengths increases.

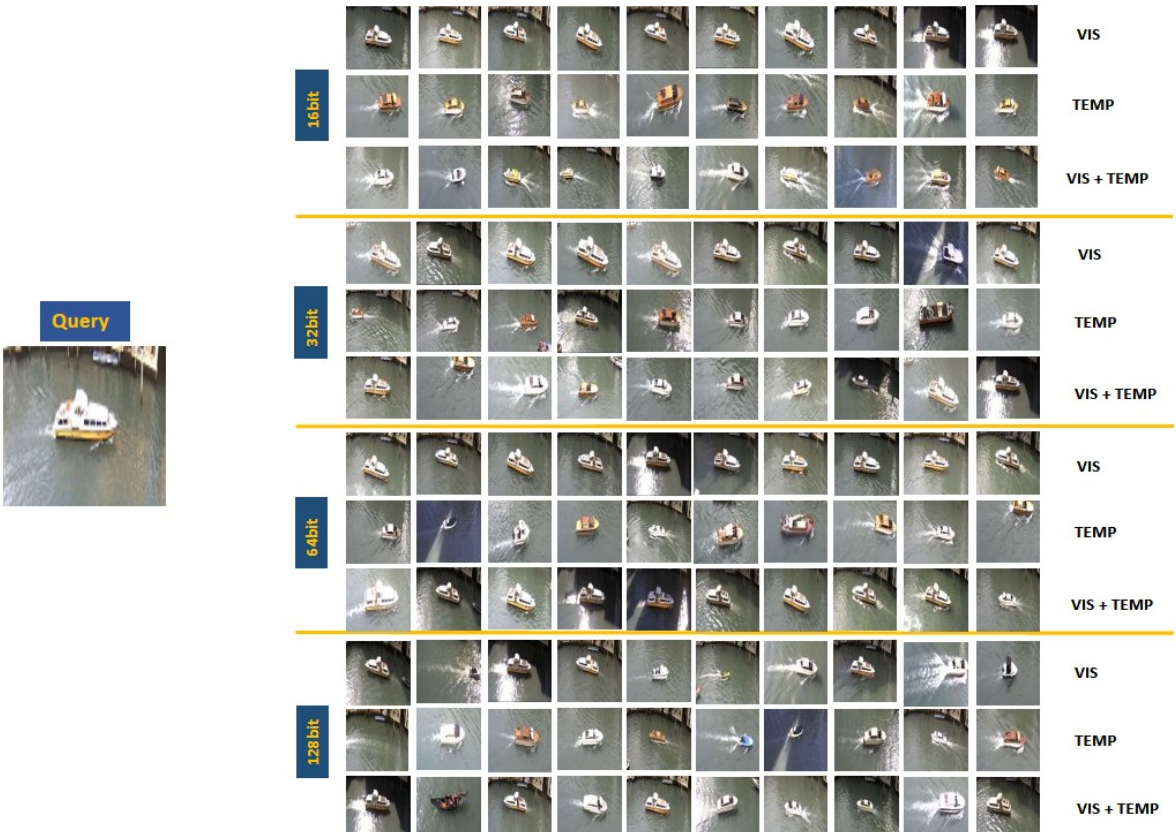


**Figure 44.** The first 10 results from BiasHash method for code lengths 16, 32, 64, 128 bits and for any combination of visual and temporal information on MarDCT dataset

| Modality/k | 30 | 50 | 100 | 200 | 300 |
|---|---|---|---|---|---|
| 16bit | | | | | |
| Visual | 0.68471 | 0.68060 | 0.68202 | 0.67634 | 0.67325 |
| Temporal | 0.26297 | 0.24609 | 0.23063 | 0.21987 | 0.21741 |
| Visual+Temporal | 0.67939 | 0.66869 | 0.64720 | 0.62141 | 0.60141 |
| 32bit | | | | | |
| Visual | 0.71850 | 0.71477 | 0.70297 | 0.69346 | 0.68530 |
| Temporal | 0.22164 | 0.21367 | 0.20183 | 0.19014 | 0.18485 |
| Visual+Temporal | 0.67939 | 0.66869 | 0.64720 | 0.62053 | 0.60141 |
| 64bit | | | | | |
| Modality/k | 30 | 50 | 100 | 200 | 300 |
| | | | | | |

| Modality/k | 30 | 50 | 100 | 200 | 300 |
|---|---|---|---|---|---|
| Visual | 0.76496 | **0.76296** | **0.75517** | **0.75517** | **0.75097** |
| Temporal | 0.23998 | 0.22592 | 0.20717 | 0.19217 | 0.18774 |
| Visual+Temporal | **0.77298** | 0.76243 | 0.74044 | 0.71485 | 0.69572 |
| 128bit | | | | | |
| Visual | 0.72811 | 0.72372 | 0.71664 | 0.70857 | 0.70373 |
| Temporal | 0.24670 | 0.22626 | 0.20237 | 0.18613 | 0.18025 |
| Visual+Temporal | 0.68879 | 0.66979 | 0.63981 | 0.60418 | 0.58080 |

**Table 2.** The AP@k of BiasHash method for code lengths 16, 32, 64, 128 bit and for any combination of visual and temporal modality on MarDCT dataset

Figure 45 presents the top 10 results of each method for different modalities. BiasHash returns the most relevant and better quality results compared to the two state-of-the-art methods. In addition, Table 3 contains the performance of each method in terms of AP@k, for k = 30, 50, 100, 200 and 300.

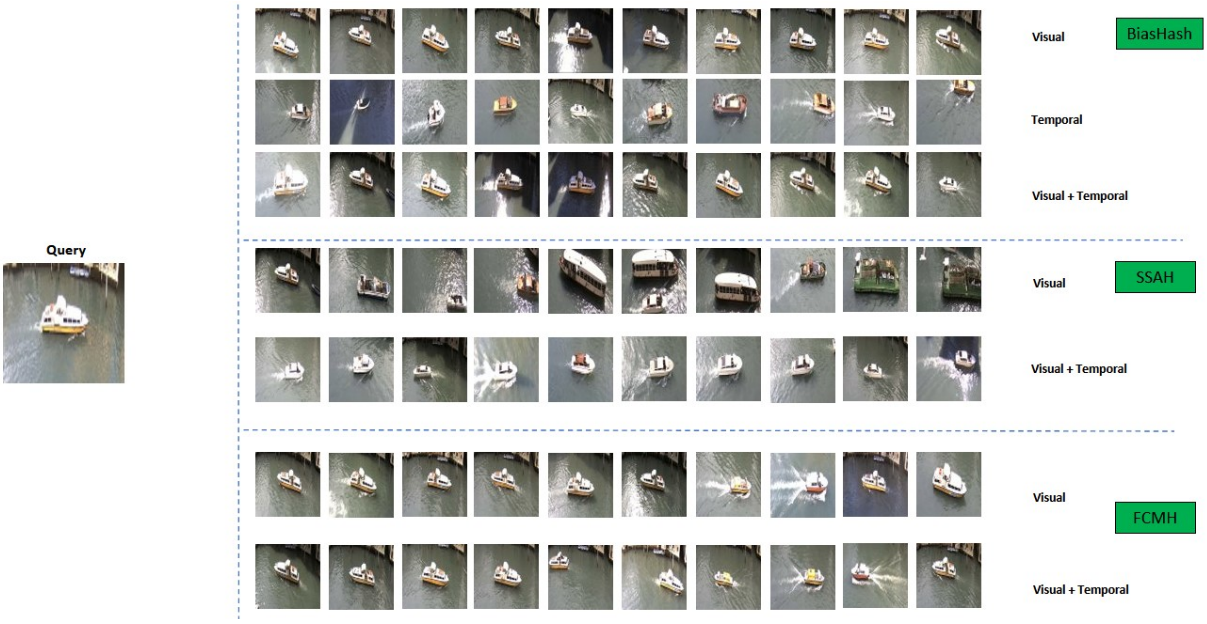


**Figure 45.** The first 10 results from BiasHash, SSAH and FCMH methods for any combination of visual and temporal information on MarDCT dataset for 64 bit

| Method/k | 30 | 50 | 100 | 200 | 300 |
|---|---|---|---|---|---|
| VISUAL | | | | | |
| | | | | | |

| Method/k | 30 | 50 | 100 | 200 | 300 |
|---|---|---|---|---|---|
| SSAH | 0.44771 | 0.44249 | 0.43509 | 0.48787 | 0.41545 |
| FCMH | 0.71700 | 0.70231 | 0.68123 | 0.67001 | 0.66321 |
| BiasHash | 0.76497 | **0.76296** | **0.75517** | **0.75517** | **0.75097** |
| TEMPORAL | | | | | |
| SSAH | - | - | - | - | - |
| FCMH | - | - | - | - | - |
| BiasHash | 0.23998 | 0.22592 | 0.20717 | 0.19217 | 0.18774 |
| VISUAL+TEMPORAL | | | | | |
| SSAH | 0.22314 | 0.23331 | 0.22940 | 0.21694 | 0.21514 |
| FCMH | 0.71180 | 0.70731 | 0.69344 | 0.68012 | 0.67722 |
| BiasHash | **0.77298** | 0.76243 | 0.74044 | 0.71485 | 0.69572 |

**Table 3.** The AP@k of BiasHash method for code lengths 16, 32, 64, 128 bit and for any combination of visual and temporal modality on MarDCT dataset

### 5.4.2 Conclusions

The performance of BiasHash mainly increases as the hash code length increases, reflecting its capability of utilizing longer hash codes to better preserve information. It should be noted that the results for temporal modality are the lowest, because the datetimes of the dataset have a wide range. However, the BiasHash performs better in multimodal queries, which is expected, because if a method uses more information, has a more compact understanding of the input data. Furthermore, BiasHash outperforms the two state-of-the-art methods, namely SSAH and FCMH, and it can also perform multimodal queries in contrast to the other two methods.

## *5.5 SeaDronesSee dataset*

This section includes some quality and visual results of the SeaDronesSee dataset. Specifically, Figure 46, Figure 47, Figure 48, Figure 49 contain some visual results of the retrieved results of BiasHash for different code lengths and modalities. Moreover, experiments with comparison with two state-of-the-art methods, SSAH (Li2018), FCMH (Wang2021), in SeaDronesSee are shown in Figure 50. Finally, Table 4 and Table 5 provide some quality results in terms of AP@k, k = 20, 50, 100, 200, 300, using Equation (13), for BiasHash and for all compared methods.

### 5.5.1 Experiments

Figure 46, Figure 47, Figure 48 and Figure 49 presents the top 10 results of BiasHash in SeaDronesSee for different modalities and for code lengths 16, 32, 64 and 128bit, respectively. Figure 50 contains the visual results of BiasHash and the other two compared methods.

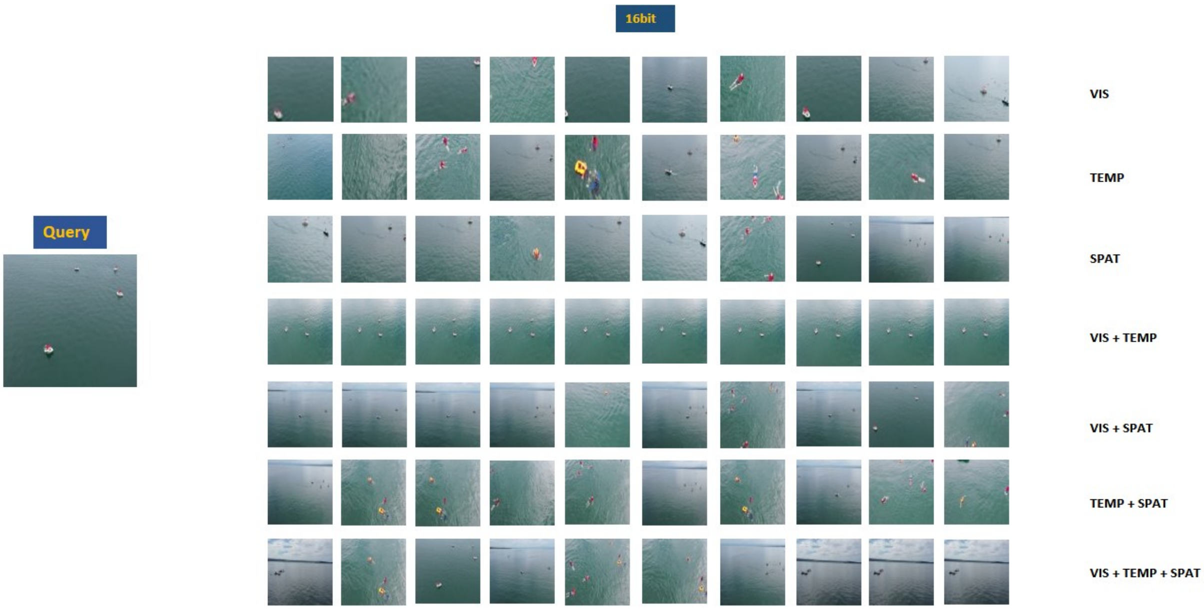


**Figure 46.** The first 10 results from BiasHash method for 16bits and for any combination of visual, temporal and spatial information in SeaDronesSee dataset for 128bit

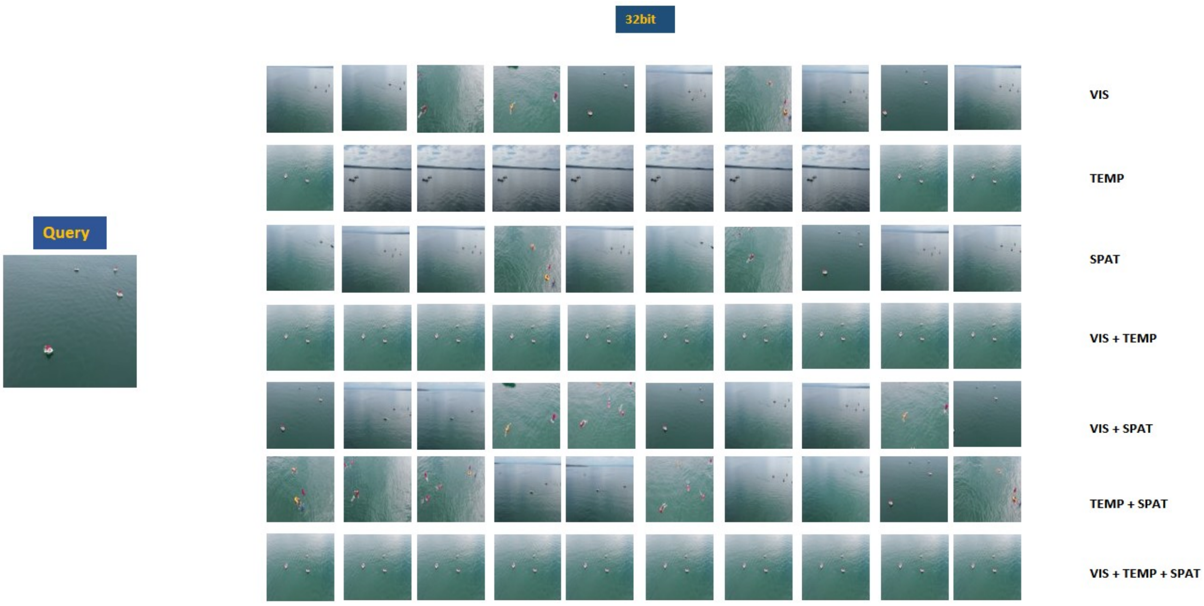


**Figure 47.** The first 10 results from BiasHash method for 32bits and for any combination of visual, temporal and spatial information in SeaDronesSee dataset for 128bit

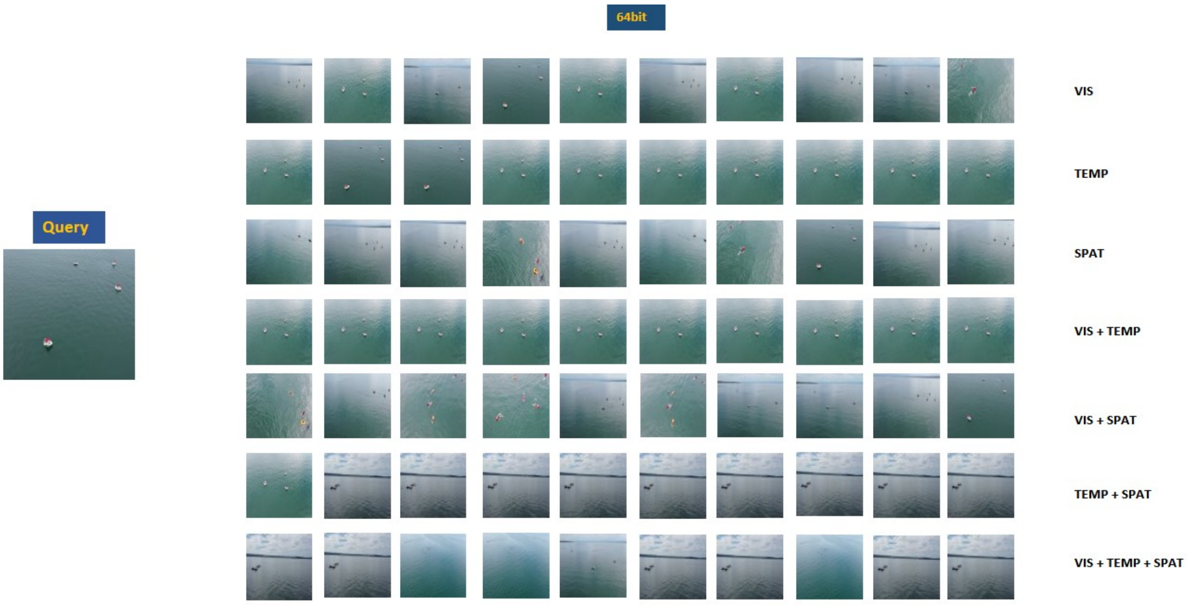


**Figure 48.** The first 10 results from BiasHash method for 64bits and for any combination of visual, temporal and spatial information in SeaDronesSee dataset for 128bit


**Figure 49.** The first 10 results from BiasHash method for 128bits and for any combination of visual, temporal and spatial information in SeaDronesSee dataset for 128bit

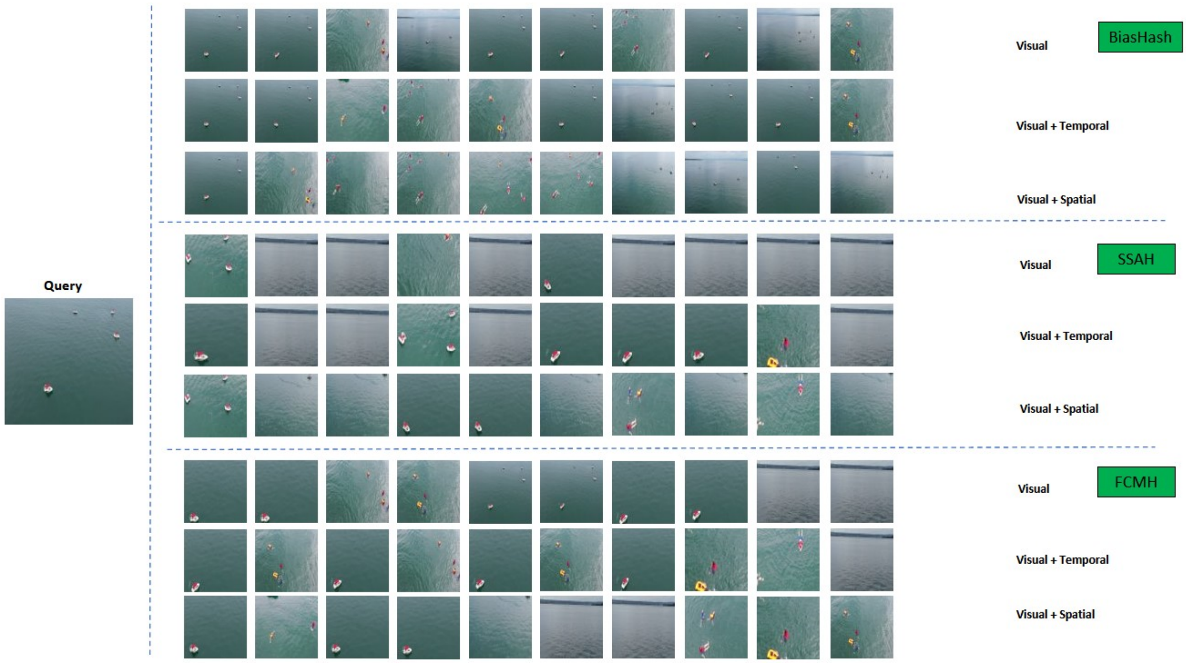


**Figure 50.** The first 10 results from BiasHash, SSAH and FCMH methods for any combination of visual and temporal information on SeaDronesSee dataset for 128bit

Table 4 includes the AP@k of BiasHash for any combination of one or more modalities, for k = 30, 50, 100, 200, 300. Also, Table 5 records the performance of BiasHash and the other two state-of-the-art methods.

| Modality/k | 30 | 50 | 100 | 200 | 300 |
|---|---|---|---|---|---|
| 16bit | | | | | |
| Visual | 0.642-1 | 0.61295 | 0.55992 | 0.58315 | 0.57216 |
| Temporal | 0.65935 | 0.65729 | 0.65904 | 0.66198 | 0.66082 |
| Spatial | 0.77038 | 0.65572 | 0.62976 | 0.61657 | 0.61209 |
| Visual+Temporal | 0.67849 | 0.66679 | 0.64113 | 0.62882 | 0.62151 |
| Visual+Spatial | 0.64049 | 0.62127 | 0.69255 | 0.68269 | 0.68294 |
| Temporal+Spatial | 0.61908 | 0.62163 | 0.62129 | 0.61837 | 0.61837 |
| All modalities | 0.61020 | 0.69695 | 0.69075 | 0.68753 | 0.68216 |
| 32bit | | | | | |
| Modality/k | 30 | 50 | 100 | 200 | 300 |
| Visual | 0.63139 | 0.60922 | 0.68982 | 0.68207 | 0.57739 |
| | | | | | |

| **Modality/k** | **30** | **50** | **100** | **200** | **300** |
|---|---|---|---|---|---|
| Temporal | 0.81372 | 0.80309 | 0.86986 | 0.83075 | 0.70001 |
| Spatial | 0.67038 | 0.65572 | 0.62976 | 0.61657 | 0.61209 |
| Visual+Temporal | 0.64351 | 0.62757 | 0.61631 | 0.61657 | 0.61209 |
| Visual+Spatial | 0.64024 | 0.63654 | 0.61863 | 0.60199 | 0.59869 |
| Temporal+Spatial | 0.57993 | 0.57961 | 0.56072 | 0.57256 | 0.57623 |
| All modalities | 0.73017 | 0.70553 | 0.78863 | 0.77746 | 0.77041 |
| 64bit | | | | | |
| Visual | 0.69373 | 0.66930 | 0.64425 | 0.62312 | 0.61322 |
| Temporal | 0.78130 | 0.74111 | 0.71239 | 0.69649 | 0.68897 |
| Spatial | 0.67038 | 0.65572 | 0.62976 | 0.61657 | 0.51209 |
| Visual+Temporal | 0.68881 | 0.66834 | 0.64753 | 0.62158 | 0.61108 |
| Visual+Spatial | 0.62182 | 0.60906 | 0.59324 | 0.57812 | 0.56439 |
| Temporal+Spatial | 0.63968 | 0.64063 | 0.61905 | 0.61905 | 0.61999 |
| All modalities | 0.71639 | 0.79649 | 0.77192 | 0.75767 | 0.75157 |
| | | | | | |
| **Modality/k** | **30** | **50** | **100** | **200** | **300** |
| 128bit | | | | | |
| Visual | 0.88117 | 0.83167 | 0.77889 | 0.73642 | 0.71339 |
| Temporal | 0.85718 | 0.85494 | 0.83462 | 0.78276 | 0.75135 |
| Spatial | 0.67038 | 0.65572 | 0.62976 | 0.61657 | 0.61209 |
| Visual+Temporal | 0.83616 | **0.88854** | 0.82684 | **0.86351** | **0.82581** |
| Visual+Spatial | 0.85842 | 0.82639 | 0.77752 | 0.73691 | 0.71219 |
| Temporal+Spatial | 084546 | 0.83930 | 0.79000 | 0.72869 | 0.69195 |
| All modalities | **0.89847** | 0.84899 | **0.89574** | 0.84555 | 0.80934 |

**Table 4.** The AP@k of BiasHash method for code lengths 16, 32, 64, 128 bit and for any combination of visual, temporal and spatial modality on SeaDronseSee dataset

| Method/k | 30 | 50 | 100 | 200 | 300 |
|---|---|---|---|---|---|
| VISUAL | | | | | |
| SSAH | 0.64879 | 0.60450 | 0.60467 | 0.63433 | 0.58913 |
| FCMH | 0.79231 | 0.80302 | 0.74111 | 0.72001 | 0.69122 |
| BiasHash | 0.88117 | 0.83167 | 0.77889 | 0.73642 | 0.71339 |
| TEMPORAL | | | | | |
| SSAH | - | - | - | - | - |
| FCMH | - | - | - | - | - |
| BiasHash | 0.85718 | 0.85494 | 0.83462 | 0.78276 | 0.75135 |
| SPATIAL | | | | | |
| SSAH | - | - | - | - | - |
| FCMH | - | - | - | - | - |
| BiasHash | 0.67038 | 0.65572 | 0.62976 | 0.61657 | 0.61209 |
| | | | | | |
| **Method/k** | **30** | **50** | **100** | **200** | **300** |
| VISUAL+TEMPORAL | | | | | |
| SSAH | 0.65786 | 0.60991 | 0.60798 | 0.61419 | 0.55816 |
| FCMH | 0.78261 | 0.77231 | 0.77001 | 0.76121 | 0.74001 |
| BiasHash | 0.83616 | **0.88854** | 0.82684 | **0.86351** | **0.82581** |
| VISUAL+SPATIAL | | | | | |
| SSAH | 0.73900 | 0.68432 | 0.65019 | 0.64195 | 0.60607 |
| FCMH | 0.78261 | 0.77231 | 0.77001 | 0.76121 | 0.74001 |
| BiasHash | 0.85842 | 0.82639 | 0.77752 | 0.73691 | 0.71219 |
| **Method/k** | **30** | **50** | **100** | **200** | **300** |
| TEMPORAL+SPATIAL | | | | | |
| | | | | | |

| Method/k | 30 | 50 | 100 | 200 | 300 |
|---|---|---|---|---|---|
| SSAH | - | - | - | - | - |
| FCMH | - | - | - | - | - |
| BiasHash | 0.84546 | 0.83930 | 0.79000 | 0.72869 | 0.69195 |
| VISUAL+TEMPORAL+SPATIAL | | | | | |
| SSAH | - | - | - | - | - |
| FCMH | - | - | - | - | - |
| BiasHash | **0.89847** | 0.84899 | **0.89547** | 0.84555 | 0.80934 |

**Table 5.** The AP@k of BiasHash method for code lengths 16, 32, 64, 128 bit and for any combination of visual, temporal and spatial modality on SeaDronseSee dataset**.**

### 5.5.2 Conclusions

Similar to the results of previous dataset, the performance of BiasHash increases as the hash code length increases. The results of BiasHash for the temporal modality are better in comparison with the MarDCT dataset, because the datetimes are closer. Moreover, BiasHash outperforms almost in all cases the two state-of-the-art methods, SSAH and FCMH and can perform multimodal queries. Only for the combination of visual-spatial queries FCMH has better results for some topk results.

## *5.6 SeaShips dataset*

This section provides some quality and visual results of the SeaDronesSee dataset. In more details, it includes some visual results of the retrieved results of BiasHash for different code lengths and modalities in Figure 51Figure 51. In addition, experiments with comparison with SSAH (Li2018), FCMH (Wang2021) methods on SeaDronesSee exist in Figure 52. Finally, Table 6 and Table 7 contain some quality results in terms of AP@k, k = 20, 50, 100, 200, 300, using Equation (13), for BiasHash and for all compared methods.

### 5.6.1 Experiments

Figure 51 illustrates the top 10 results for visual modality and for code lengths 16, 32, 64 and 128 bit on SeaShips dataset. Table 6 presents the AP@k of BiasHash, SSAH and FCMH, for k = 30, 50, 100, 200 and 300.

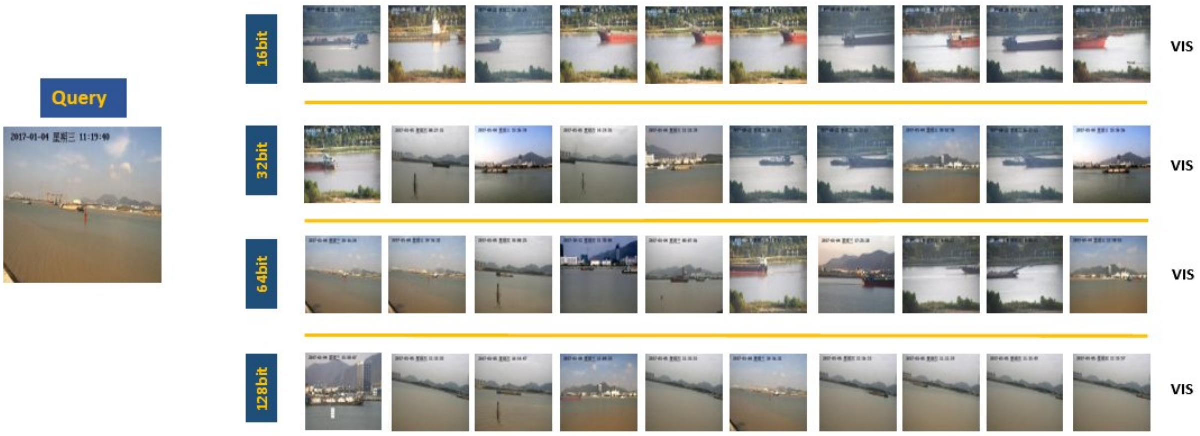


**Figure 51.** The first 10 results from BiasHash method for code lengths 16, 32, 64, 128 bits and for visual information on SeaShips dataset

| Modality/k | 30 | 50 | 100 | 200 | 300 |
|---|---|---|---|---|---|
| 16bit | | | | | |
| Visual | 0.54552 | 0.54825 | 0.51966 | 0.52038 | 0.52411 |
| 32bit | | | | | |
| Visual | 0.59675 | 0.58577 | 0.56744 | 0.55321 | 0.54472 |
| 64bit | | | | | |
| Visual | **0.64929** | **0.63956** | **0.62851** | **0.61626** | **0.61036** |
| 128bit | | | | | |
| Visual | 0.61644 | 0.59437 | 0.55881 | 0.53092 | 0.51642 |

**Table 6.** The AP@k of BiasHash method for code lengths 16, 32, 64, 128 bit and for visual modality on SeaShips dataset

Figure 52 contains the top 10 images for each method on SeaShips dataset for a given query. Also, Table 7 has the AP@k of BiasHash, FCMH and SSAH, for k = 30, 50, 100, 200 and 300.

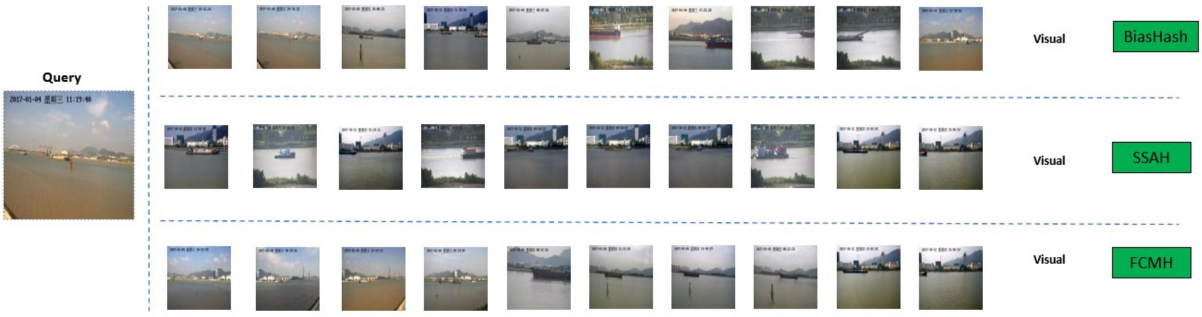


**Figure 52.** The first 10 results from BiasHash, SSAH and FCMH methods for any combination of visual and temporal information on SeaDronesSee dataset for 64bit

| Method/k | 30 | 50 | 100 | 200 | 300 |
|---|---|---|---|---|---|
| VISUAL | | | | | |
| SSAH | 0.44855 | 0.44775 | 0.41889 | 0.38972 | 0.38109 |
| FCMH | 0.60230 | 0.59001 | 0.58822 | 0.58322 | 0.56001 |
| BiasHash | **0.64929** | **0.63956** | **0.62851** | **0.61626** | **0.61036** |

**Table 7.** The AP@k of BiasHash method for code lengths 16, 32, 64, 128 bit and for any combination of visual and temporal modality on SeaShips dataset

### 5.6.2 Conclusions

BiasHash has better performance as the hash code length increases. Furthermore, BiasHash outperforms the two state-of-the-art methods when applied on the SeaShips dataset.

After carefully observing the results of the experiments realized in the available datasets and the comparison with existing methods, we can conclude that in general BiasHash gives the better results and can handle multimodal queries.

# 6 Integration of the Multimodal Indexing service to ISOLA

This section discusses the integration of the Integration Layer and Multimodal Indexing service into the ISOLA system. In general, the system of ISOLA is set in the open source, distributed messaging system Apache Kafka architecture. More specifically, the Event-Driven Architecture is used. The Message Bus application of Kafka provides the communication with events between services without any directly contacting of one service to another. Each part works asynchronously, without being linked to each other. In Kafka messages are sent to and read from places called topics. Each Topic has a name.

Kafka architecture has four actors: Broker, Zookeeper, Producer and Consumer.

- Kafka contains one or more Brokers, which work interdependently. Messages sent to Kafka in brokers are stored on the hard drive and processed.
- Zookeeper is an open-source software that helps Kafka to manage all Brokers.

- Producer writes data to Kafka by sending record in JSON format via it.
- Consumer reads data from Kafka by receiving record in JSON format via it.

The Integration Layer and Multimodal Indexing service has one consumer that hears to TOPIC1, TOPIC10 and TOPIC15 and one producer that hears to TOPIC2. Figure 53**Error! Reference source not found.** illustrates the topics from Kafka the service reads and sends. In particular, the consumer listens to messages from mission drones service, object detection service abnormal behavior service, face detection service, and underwater vehicle service in TOPIC1, from mobile and social media service in TOPIC10 and from decision support service in TOPIC15, while produces messages to ontologies service and decision support in TOPIC2.

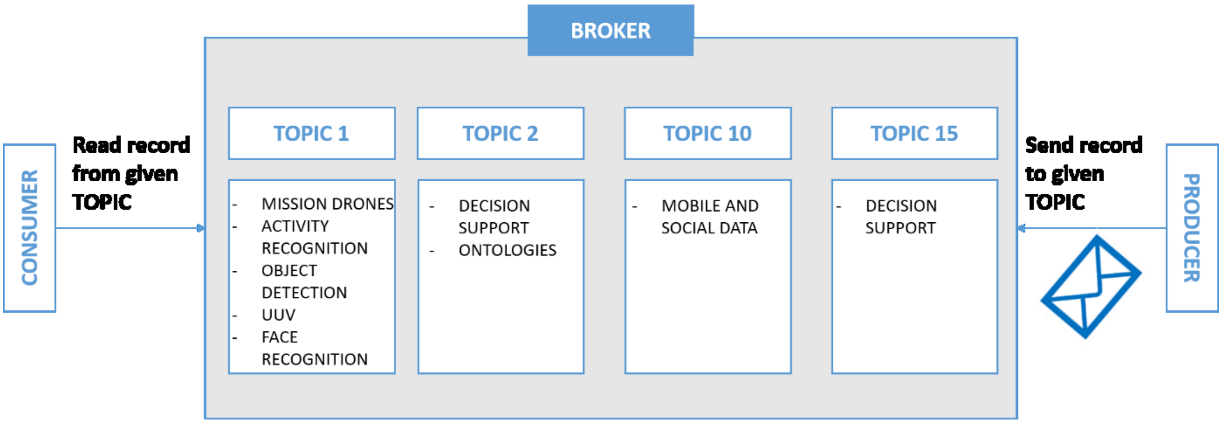


**Figure 53.** Kafka architecture for Integration Layer and Multimodal Indexing service

Figure 54**Error! Reference source not found.** presents an example of a message that the consumer of the service can read. The available TOPIC_NAME values are TOPIC_01, TOPIC_10 and TOPIC_15. The field sender can take the values “ACCELI_MISSION”, “CENTRIC_DECISION”, “CERTH_ACT”, “CERTH_OBJ”, “IDMG_FACE”, “OMST_UUV” and “SIMAVI_MOBILE”. Finally, the SOURCE_NAME can take the values “”ACCELI_MISSION”, “IBM_DRONES”, “IDMG_FACE”, “OMST_UUV” and “SIMAVI_MOBILE” and corresponds to the source from which the information begin in a Kafka flow scenario.


```
{
    "header": {
        "topicName": "TOPIC_NAME",
        "topicMajorVersion": 2,
        "topicMinorVersion": 4,
        "sender": "SENDER_NAME",
        "msgIdentifier": "330485579130999987214468708833 17897954",
        "sentUTC": "2022-07-28T12:00:00.00Z",
        "status": "Actual",
        "recipients": "CERTH_MULTI",
        "references": "33048557913099998721446870883317897952"
    },
    "body": {
        "source": "SOURCE_NAME",
        ...
    }
}
```


**Figure 54.** An example from a Kafka message, which the service can read from message bus

In addition, Figure 55**Error! Reference source not found.** contains an example of a produced message from the service. The field recipients can take values CERTH_ONTOL and CENTRIC_DECISION. Similarly with the consumer messages from Figure 54**Error! Reference source not found.**, the SOURCE_NAME can take the aforementioned values.


```
{
    "header": {
        "topicName": "TOPIC_2",
        "topicMajorVersion": 2,
        "topicMinorVersion": 4,
        "sender": "CERTH_MULTI",
        "msgIdentifier": "33048557913099998721446870883317897954",
        "sentUTC": "2022-07-28T12:00:00.00Z",
        "status": "Actual",
        "recipients": "RECIPIENTS_NAME",
        "references": "33048557913099998721446870883317897952"
    },
    "body": {
        "source": "SOURCE_NAME",
        ...
     }
}
```


**Figure 55.** An example from a Kafka message, which the service can produce to message bus

The service forwards the messages from the mission drones, object detection, abnormal behavior, underwater vehicle services and mobile and social data service to other services belonging in TOPIC_02. In particular, it sends the messages from mobile and social data service if and only if a keyword from a list of the predefined alert keywords exists in the Twitter text. More information for the alert keywords list is given in Section 4.1. Furthermore, the service extracts visual features from data of object detection, abnormal behaviour and underwater vehicle services, temporal features from data of mission drones, object detection, abnormal behaviour, underwater vehicle and mobile and social data services, and spatial features from data of object detection and abnormal behaviour services using the corresponding feature extractor. After that it computes the hash codes of each available modality and saves all the metadata to the MongoDB. Apart from that the service computes visual similarities for UUV and UAV data to queries from decision support service. It returns the most relevant to a given query results in a list.

The data from Kafka messages, which are received from the consumer are stored in a MongoDB. A different collection corresponds to a different message. The service uses eight collections in the MongoDB. The collections are ACCELI_MISSION, CERTH_ACT, CERTH_OBJ, CERTH_OBJ_QUERIES, IDMG_FACE, OMST_UUV, OMST_UUV_QUERIES and SIMAVI_MOBILE. More information of each collection is given in Section 4.1. Figure 56**Error! Reference source not found.** illustrates the framework of the overall service architecture. It consists of two parts, one is the Kafka architecture and the other one is the database architecture with additional procedures on each part. The Kafka part contains the read and write messages via Message Bus, while the database inserts and retrieves data from the MongoDB. The service is dockerised and is connected via Kafka with other services. Docker deploys a new application container and reduces the execution overhead of each service.

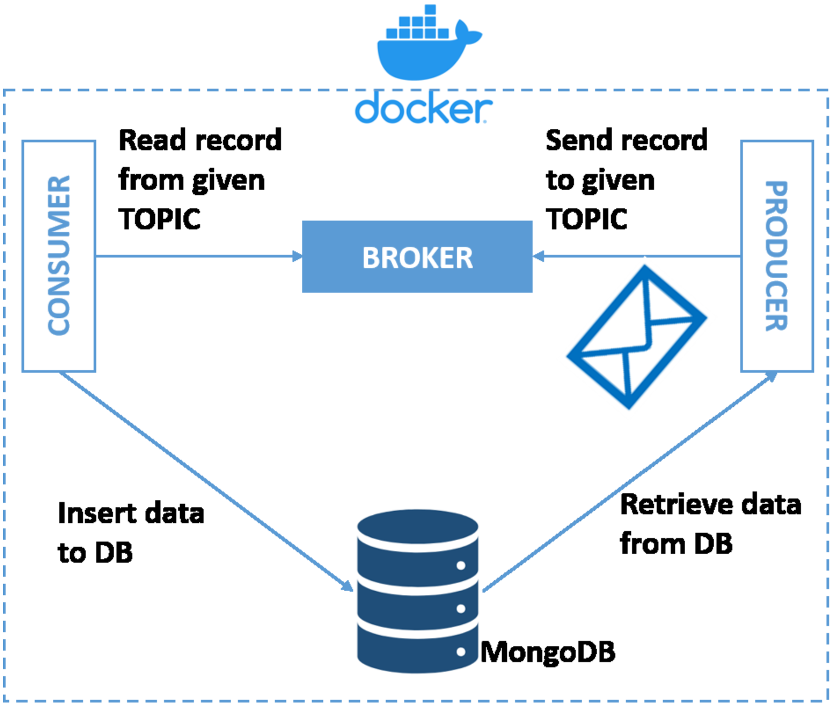


**Figure 56.** Example with the overall architecture of the service

# 7 Conclusions

In this deliverable a description of the BiasHash framework and the Integration Layer and Multimodal Indexing service has been included. Then, a concise presentation of the state-of-the-art technologies on multimodal retrieval using hashing is presented and the presentation of the proposed framework is presented in detail. Furthermore, several public datasets and ISOLA datasets are presented and experiments are realized on these datasets in order to conclude on the efficiency of the method. In summary, the results so far are quite satisfactory. Apart from that, a problem was the unavailability of public UUV data. Therefore, a method is used that computes the differences of UUV data in feature level.

Finally, the framework proposed is transformed into a service that is integrated into ISOLA system and the basic procedures related to introduction of data into the service through Kafka and the storing of interim data into a MongoDB are explained.

As next steps no additional experiments on ISOLA dataset will be performed. However more effort on the dockerisation of the service to be uninterrupted as well as the integration of the service to the ISOLA project will be allocated.

# Acknowledgements

This work was supported by the EU's Horizon 2020 research and innovation programme under grant agreement H2020-883302 ISOLA.

# References

Bai, C., Zeng, C., Ma, Q., Zhang, J., & Chen, S. (2020). Deep adversarial discrete hashing for cross-modal retrieval. ICMR '20: Proceedings of the 2020 International Conference on Multimedia Retrieval, pp. 525–531. https://doi.org/10.1145/3372278.3390711

Bloisi, D. D., Iocchi, L., Pennisi, A., Tombolini, L. (2015). ARGOS-Venice Boat Classification. 12th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), pp.1-6, 2015. doi: https://doi.org/10.1109/AVSS.2015.7301727

Cao, Z., Sun, Z., Long, M., Wang, J., & Yu P.S. (2018). Deep Priority Hashing. ACM Multimedia Conference (MM), pp. 1653-1661. https://doi.org/10.1145/3240508.32405434

Chen, Y., & Lu, X. (2020). Deep discrete hashing with pairwise correlation learning. Neurocomputing (Elsevier), 385, pp. 111-121. https://doi.org/10.1016/j.neucom.2019.12.078

Cui, H., Zhu, L., Li, J., Cheng, Z., & Zhang, Z. (2021). Two-pronged Strategy: Lightweight Augmented Graph Network Hashing for Scalable Image Retrieval. Proceedings of the 29th ACM International Conference on Multimedia (ACM MM). https://doi.org/10.1145/3474085.3475605

Fu, H., Li, Y., Zhang, H., Liu, J., & Yao, T. (2020). Rank-embedded hashing for large-scale image retrieval. Proceedings of the 2020 International Conference on Multimedia Retrieval, pp. 563-570. https://doi.org/10.1145/3372278.3390716

Gu, W., Gu, X., Gu, J., Li, B., Xiong, Z., & Wang, W. (2019). Adversary Guided Asymmetric Hashing for Cross-Modal Retrieval. Proceedings of the 2019 on International Conference on Multimedia Retrieval, pp. 159-167. https://doi.org/10.1145/3323873.3325045

Indyk, P., Motwani, R., Raghavan, P., & Vempala, S. S. (1997). Locality-Preserving Hashing in Multidimensional Spaces. In Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997, Frank Thomson Leighton and Peter W. Shor (Eds.). ACM, pp. 618-625. https://doi.org/10.1145/258533.258656

Maaten, L., & Hinton, G. (2008). Visualizing data using t-sne. Journal of Machine Learning Research, 9(85), pp. 2579-2605.

Jiang, Q.-Y., & Li, W.-J. (2018). Discrete Latent Factor Model for Cross-Modal Hashing. IEEE Transactions on Image Processing 28(7), pp. 3490-3501. https://doi.org/10.1109/TIP.2019.2897944

Li, C., Deng, C., Li, N., Liu, W., Gao, X., & Tao, D. (2018). Self-Supervised Adversarial Hashing Networks for Cross-Modal Retrieval. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4242-4251. https://doi.org/10.1109/CVPR.2018.00446

Lin, G., Shen, C., Shi, Q., Hengel, A., & Suter, D. (2014). Fast supervised hashing with decision trees for high-dimensional data. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1963–1970. https://doi.org/10.1109/CVPR.2014.253

Lin, Z., Ding, G., Hu, M., & Wang, J. (2015). Semantics-Preserving Hashing for Cross-View Retrieval. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3864-3872. https://doi.org/10.1109/CVPR.2015.7299011

Lin, M., Ji, R., Liu, H., & Wu, Y. (2018). Supervised Online Hashing via Hadamard Codebook Learning. Proceedings of the 26th ACM international conference on Multimedia, pp. 1635-1643. https://doi.org/10.1145/3240508.3240519

Liu, X., Hu, Z., Ling, H., & Cheung, Y. (2019). MTFH: A matrix tri-factorization hashing framework for efficient cross-modal retrieval. IEEE Transactions on Pattern Analysis and Machine Intelligence 43(3), pp. 964-981. https://doi.org/10.1109/TPAMI.2019.2940446

Mandal, D., Chaudhury, K. N., & Biswas, S. (2018). Generalizes Semantic Preserving Hashing for N-Label Cross-Modal Retrieval. IEEE Transactions on Image Processing (28)1, pp. 102-112. https://doi.org/10.1109/TIP.2018.2863040

Papadimitriou, C.H. (1981). On the complexity of integer programming. J. ACM, 28(4).

https://doi.org/10.1145/322276.322287

Pegia, M., Moumtzidou, A., Gialampoukidis, I., Johnsson, B.T., Vrochidis, S., & Kompatsiaris, I. (2022). BiasHash: A Bayesian Hashing Framework for Image Retrieval. IEEE 14th Image, Video, and Multidimensional Signal Processing Workshop (IVMSP). https://doi.org/10.1109/IVMSP54334.2022.9816233

Pedregosa, F., & et al. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, pp. 2825-2830. https://hal.inria.fr/hal-00650905v2

Tipping, M.E. (2001). Sparse bayesian learning and the relevance vector machine. Journal of Machine Learning Research, 1(3), pp. 211–244. https://doi.org/10.1162/15324430152748236

Ruder, S. (2017) An overview of gradient descent optimization algorithms. https://arxiv.org/abs/1609.04747

Shao, Z., Wu, W., Wang, Z., Du, W., Li, C.. (2018). SeaShips: A Large-Scale Precisely-Annotated Dataset for Ship Detection. IEEE Transactions on Multimedia. https://doi.org/10.1109/TMM.2018.2865686

Varga, L. A., Kiefer, B., Messmer, M., Zell, A. (2022). Seadronessee: A maritime benchmark for detecting humans in open water. Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 2260-2270. https://doi.org/10.48550/arXiv.2105.01922

Wang, Y., Chen, Z.-D., Luo, X., Li, R., & Xu, X.-S. (2021). Fast Cross-Modal Hashing With Global and Local Similarity Embedding. IEEE Transactions on Cybernetics (IEEE), pp. 1-14. https://doi.org/10.1109/TCYB.2021.3059886

Xie, Y., Liu, Y., Wang, Y., Gao, L., Wang, P., & Zhou, K. (2020). Label-Attended Hashing for Multi-Label Image Retrieval. Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI), pp. 955–962. https://doi.org/10.24963/ijcai.2020/133%205

Yuan, L., Wang, Zhang, X., Tay, F. EH, Jie, Z., Liu, W., & Feng, J. (2020). Central Similarity Quantization for Efficient Image and Video Retrieval. In Proceedings of of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3083-3092. https://doi.org/10.1109/CVPR42600.2020.00315

Zhang, D., Wang, J., Cai, D., & Lu, J. (2010). Self-taught hashing for fast similarity search. In Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 18-25. https://doi.org/10.1145/1835449.1835455

Zhao, H., She, X., Wang, S., & Ma, K. (2021). Fast Discrete Matrix Factorization Hashing for Large-Scale Cross-Modal Retrieval. International Conference on Multimedia Modelling. Springer, Cham, pp. 24-36. https://doi.org/10.1109/TCYB.2015.2392052

Zhen, Y., Gao, Y., Yeung, D.Y., Zha, H., & Li, X. (2016). Spectral Multimodal Hashing and Its Application to Multimedia Retrieval. IEEE Transactions on Cybernetics, 46(1), pp. 27-38. https://doi.org/10.1109/TCYB.2015.2392052

Zhou, J., Ding, G., & Guo. Y. (2014). Latent semantic sparse hashing for cross-modal similarity search. Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval, pp. 415-424. https://doi.org/10.1145/2600428.2609610

Zhu, X., Huang, Z., Shen, H. T., & Zhao, X. (2013). Linear cross-modal hashing for efficient multimedia search. In Proceedings of the 21st ACM International Conference on Multimedia, pp. 143-152. https://doi.org/10.1145/2502081.2502107