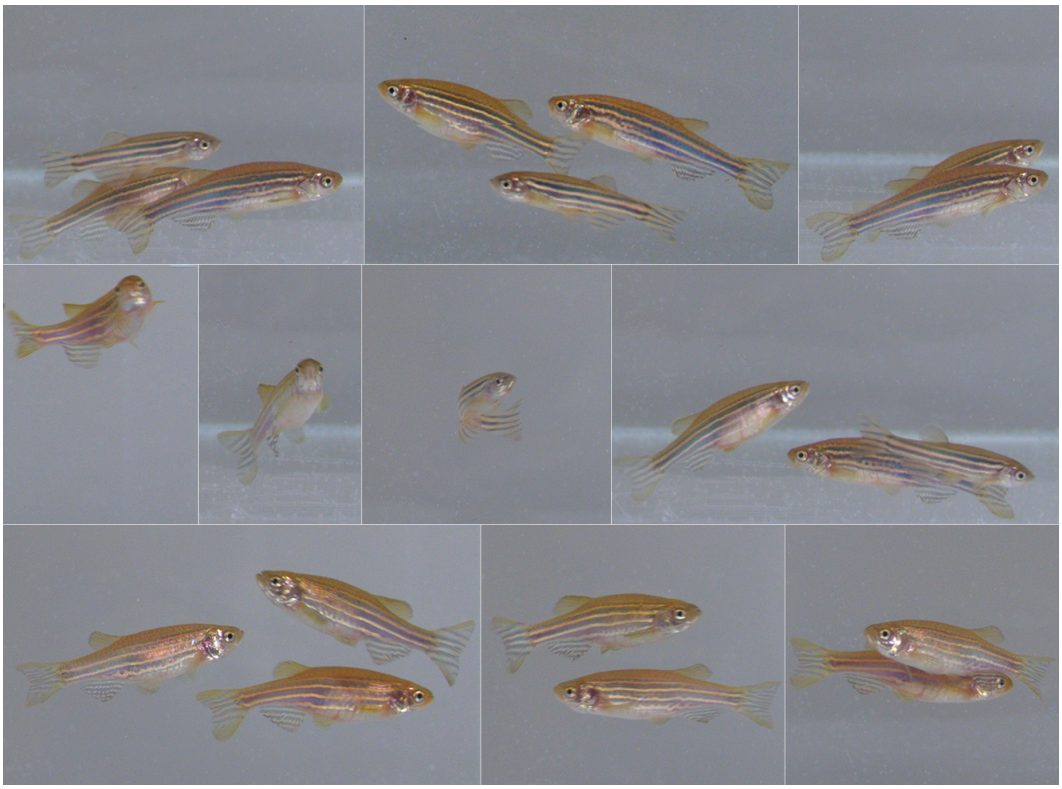
Re-Identification of Zebrafish Using Metric Learning

A Master Thesis

Anastasija Karpova & Joakim Bruslund Haurum

Vision, Graphics & Interactive Systems



AALBORG UNIVERSITY
DENMARK

June 7, 2018

This report is set with the document markup language LaTeX, originally develop by Leslie Lamport and based on Donald Knuth's TeX. The body text is set with *Computer Modern* pt 11, designed by Donald Knuth. The document was written with the use of `www.overleaf.com`, an online collaborative writing and publishing system for LaTeX. For implementation and evaluation Python 3.6 was used.

Department of Electronic Systems
VGIS, 10th semester
Fredrik Bajers Vej 7B
DK-9220 Aalborg

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Re-Identification of Zebrafish Using Metric Learning

Theme:
Master Thesis

Project Period:
P10: 1st February 2018 - 7th June 2018

Project Group:
18gr1043

Participant(s):
Anastasija Karpova
Joakim Bruslund Haurum

Supervisor:
Thomas B. Moeslund

Co-Supervisor:
Stefan H. Bengtson

Page Numbers: 81

Appendices: 19

Date of Completion:
June 7th, 2018

Abstract:

This report describes the research conducted during the master thesis for the Vision, Graphics and Interactive Systems master programme.

In this work, we explore a part of animal tracking: re-identification of zebrafish (*Danio rerio*), in particular. The literature demonstrates the importance of the zebrafish for drug development and pushing the human understanding of organisms further. The state-of-the-art re-identification methods of zebrafish employ solely a top-view and grayscale images and require a lot of data for the best performance.

This research rather aims to study how well the side-view and color images can perform in the same task while keeping data processing to a minimum. Inspired by the person re-identification field, two feature descriptors, each encoding both color and texture information, and five metric learning algorithms were tested. The contribution of the color and texture components of the feature descriptors were also investigated.

The experiments were conducted on two and three fish, separately. The results show that for the same fish, a mean average precision (mAP) of 100% can be obtained, and when testing on previously unseen fish, a mAP of 89.40% is achieved, at best. When training on small sequences, with a length up to only 50 frames, the methods can sustain very few (0-2%) identity switches. This approach clearly shows potential and encourages the further research on the topic.

Contents

Preface

This report is submitted as the thesis project at the Vision, Graphics and Interactive Systems (VGIS) master's program at Aalborg University (AAU). The focus of the project has been to investigate and test whether it is possible to re-identify zebrafish, using a single color camera view, in a controlled environment.

Citations follow the APA-style, meaning they are alphabetized and are either of the type *(Name of source/Author's surname, year)* or *Name of source/Author's surname, (year).* Links to websites in the bibliography are of the following type: Author. (Year). Title. URL. Date the website was last visited. If the year of citation could not be determined, the abbreviation *nd* is used for "not defined".

The report is intended for people who have an interest in computer vision and re-identification topics, as well as acting as documentation for future development on the system. The group assumes that the reader has a basic understanding of computer vision, machine learning, and programming, meaning that the specifics of the implemented code, as well as commonly used methods within the mentioned fields will not be covered. However, if specific methods have a central role in the program, they will be described.

We would like to give our thanks to our supervisors, Thomas B. Moeslund and Stefan H. Bengtson, for continuous guidance and assistance during the project. Additionally, we would like to thank Malte Pedersen, for providing assistance with handling the fish and the initial steps of the project. We would also like to give our thanks to the Visual Analysis of People lab at AAU, for letting us use their equipment and fish.

Aalborg University, 7th June 2018

Anastasija Karpova
akarpo13@student.aau.dk

Joakim Bruslund Haurum
jhauru13@student.aau.dk

1 | Introduction

In biology, some experiments cannot be conducted on human subjects, either due to it being unethical or infeasible. In these cases, *model organisms*, which are organisms that carry some resemblance to humans, are instead used. Classic examples of model organisms are fruit flies (*Drosophila melanogaster*) and mice (*Mus musculus*).

However, when studying the development of vertebrae organisms, the zebrafish (*Danio rerio*) has been found quite useful, due to their transparent body during early development, which allows the researchers to observe the test subjects in a non-obtrusive manner (Hill et al., 2005). It has also been found that 71.4% of human genes have a corresponding gene in zebrafish and that up to 82% of these corresponding genes are associated with human genes known to be related to diseases (Howe et al., 2013).

Zebrafish are also used to analyze group behaviors, such as the effect of different drugs (Green et al., 2012) or complex brain disorders (Kalueff et al., 2014) since the fish are highly social animals. This is reflected by the zebrafish often swimming in groups, called *shoals* or *schools*. The difference in shoals and schools lies in the degree of independence for fish movements: in shoals, fish swim independently but close to each other, and in schools fish movements are shared, by the direction and speed. Groups often show characteristics of both shoaling and schooling. Whether the fish swim in shoals or schools depend on the group size and the environment, with the fish tending to swim in schools when found in a new environment and within a smaller group (Miller and Gerlai, 2012).

To examine the differences in the external effects on the zebrafish group behavior, it is necessary to track an individual fish over time. This is typically done through a tracking system, as it is infeasible to manually annotate all the video data needed. However, since the fish swim in groups, they often occlude each other, causing the trajectories to break into tracklets, which need to be associated. The fish also swim in a quite impulsive and unpredictable way when stressed or anxious (for example by being in a new environment) (Egan et al., 2009), which makes it difficult to set up a motion model for the fish, and makes the tracking even harder.

Bengtson and Pedersen (2017) developed a 3D tracking solution utilizing stereo vision, where the fish are filmed from a top and side views, from which they can reconstruct 3D tracklets of several fish at the same time. They do however not look into associating those tracklets. In general, just a few papers solve association task based on visual characteristics, and when they do, only the top view and the head of the zebrafish are used.

In this project, we have therefore investigated whether it is possible to associate zebrafish by utilizing color images captured from a side view. The side view images are interesting to look at, as the side of zebrafish consists of a set of dark blue stripes, with gold and silver stripes in between for males and females, respectively (Wixon, 2000), from which the fish gets its name. These stripes extend onto the tail and the lower fin of the fish. It is unknown whether the pattern is identical on both sides of the fish, but it varies from fish to fish (see *Figure: 1.1*). It is hypothesized that these stripes can be used to re-identify the fish.

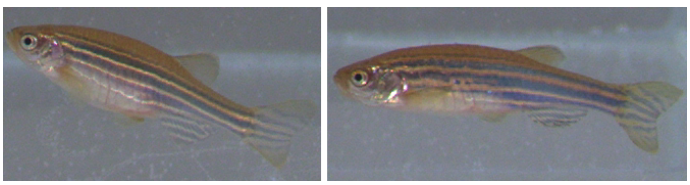


Figure 1.1: Illustration of two zebrafish with different stripe patterns.

In the scope of this project, we define properties and limitations listed below, that are partially inspired by Bengtson and Pedersen (2017):

1. **Focus only on re-identification**: The system will not deal with detecting the fish, but instead only associate a set of supplied detections.
2. **Single aquarium**: The system will only be tested on a single aquarium.
3. **Laboratory environment**: The system will be tested in a lab environment, with controlled lighting and no extra obstacles introduced.
4. **No interference**: The system must not interfere with the fish in any way.
5. **Force 2D movement**: The fish are constricted to move only in 2 dimensions, as to keep them approximately the same size in the image throughout the video.

2 | Literature Survey

This chapter describes the recent literature, which is relevant to the problem of re-identification. The described works are divided into two major parts: zebrafish tracking and person re-identification. Each section is followed by a conclusion.

2.1 Zebrafish Tracking

The field of tracking zebrafish is quite a limited area, that has only in recent years started getting more attention and traction. We focus on traditional 3D tracking methods (as MacRì et al. (2017) found that analyzing zebrafish behavior from 2D data results in erroneous conclusions); and 2D approaches based on convolutional neural networks (CNN), as in recent years, deep learning achieved state-of-the-art results in many computer vision areas.

It would be also interesting to look into tracking and re-identification of other model organisms, such as mice and fruit flies. However, in the literature found, the authors either mark the animals, in order to distinguish between them (Ohayon et al., 2013) or use the silhouette of an animal, rather than its visual characteristics, such as color or texture (Ardekani et al., 2013; Cheng et al., 2015). The first finding is troublesome to apply on zebrafish, and it also contradicts with one of the limitations for this projects, defined in *Chapter 1*. The second finding defeats the whole underlying motivation for this research, as we want to look into using texture and color of zebrafish for re-identification task. Therefore, these papers are not included in the literature survey.

2.1.1 Traditional tracking Approaches

Pérez-Escudero et al. (2014) developed a tracking system, *idTracker*, which reliably tracks small animals (fruit flies, mice, and zebrafish) under strict laboratory settings from a top view. The authors point out that previous methods have problems assigning identities correctly when multiple subjects cross paths. They extract a virtual "fingerprint" for each subject, which can solve the identity assignment problem reliably over time (even for videos taken days apart). The system requires a homogeneous lighting and a uniformed background, yet contrasting to the tracked animals.

Qian and Chen (2017) proposed a 3D tracking algorithm based on a multi-camera setup, utilizing a "feature point"-based approach, to match the fish from different camera views, for both occluded and non-occluded fish.

Method

Pérez-Escudero et al. (2014) segment the video frames, resulting in a set of fragments (BLOBs in the consecutive frames). These fragments are assumed to be the same subject if two BLOBs are overlapping only with each other between the frames. If a BLOB does not fit anywhere, a new fragment is created.

The authors calculate translation and rotation invariant *intensity maps* and *contrast maps* for each BLOB, by plotting a 2D histogram of the distances between two pixels in the BLOB and the sum of the intensities (i.e. intensity map) or the absolute difference (i.e. contrast map). This is done for all pixel pairs in the BLOB. This process is illustrated in *Figure: 2.1*. All BLOBs in the video are then compared to the intensity and contrast maps of the single-subject BLOBs, by calculating the summed distance between the maps.

The system builds a set of reference images for each subject: by finding frame sequences where the amount

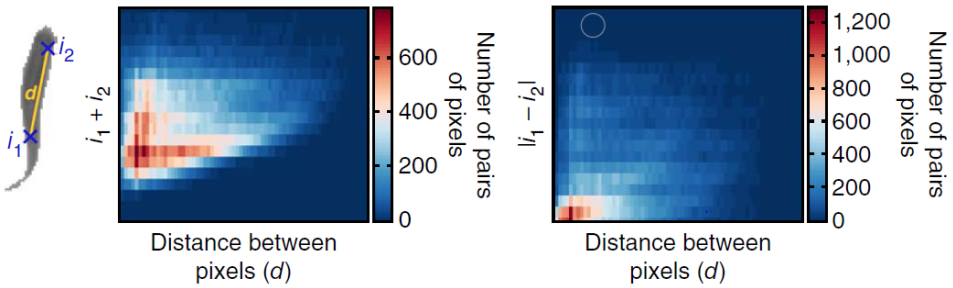


Figure 2.1: Illustration of how the intensity and contrast maps are created for an extracted fish. The intensity values at two points are compared and plotted in a 2D histogram based on the distance between the two points. Source: (Pérez-Escudero et al., 2014, Fig. 2d-f)

of one-subject fragments is equal to the number of animals. Those sequences are set to be more than 15 frames. To associate the fragments, the authors compare them as pairs, where the collection of fragments with most frames is used as reference images and the other set of fragments have test images from each fragment compared to the reference images. The intensity and contrast distances between the test image and reference images determine whether the test image is assigned to an identity.

The trajectories are built by joining up the center-of-mass of the BLOBs, using only the one-subject fragments, which results in a set of tracklets. To connect the tracklets, an iterative approach is applied based on a set of heuristics built upon the max speed of the trajectories.

Qian and Chen (2017) propose a multi-camera setup for multi-target 3D zebrafish tracking. They use three cameras, each parallel to a side of the tank and perpendicular to each other. This way even if a fish is occluded in one view, the two other views can most likely be used to determine the 3D position. The proposed algorithm consists of two main steps: object detection and object tracking.

The object detection is done by extracting the skeleton of the BLOBS, together with the center and end points of it. Based on these points, two feature models describe the detected objects. For the top view, a *Double Feature Point Model* (DFPM) is used, which describes a fish by its center point and head endpoint. The head endpoint is found in the end of the skeleton with the largest average width (as from above the fish narrows near the tail). For the side view a *Three Feature Point Model* (TFPM) is used, which describes the fish by its center point and both endpoints, since it is not as easy to find the direction of the fish, based on its appearance only.

For tracking, the authors focus on the top view, as it is more discriminative (i.e. fewer occlusions and a more stable swimming pattern), and only use the side view as supplemental information. If an object is not occluded (i.e. has no more than 2 endpoints), then it is assigned to an object in a previous frame, with the assumption that the direction and position will not have changed drastically. This is done by assigning a cost value to every possible assignment, and then finding the solution resulting in the most associations. Compared to the top view, where the DFPM encodes both position and direction, in a side view the authors calculate the direction in TFPM, by the relative position change between the two objects. This simplifies it into DFPM. Through the use of stereo matching and epipolar geometry, the authors decide whether an object in the top view matches any of the objects in the side views.

If an object is occluded, a set of tracklets is created and connected into trajectories: by matching the objects before and after the occlusions, with the use of the multi-view information. The linking is done only for the top view, as the 3D position of the tracklets can be calculated by applying stereo matching with the side views.

Results

For idTracker evaluation, Pérez-Escudero et al. (2014) report a 99.7% success of correct identities in the trajectories. The authors found that idTracker can re-identify the subjects after they left the camera view or were occluded, i.e it is applicable for examining group behaviors. The system is correctly re-identifying the animals even when several days had passed between experiments, beating both state-of-the-art methods, as well as human operators. For a robust performance, the authors suggest to use a video of at least 5 minutes, but preferably 30 minutes or longer.

Qian and Chen (2017) evaluate their algorithm on two datasets they collected themselves. Datasets are filmed at 2048×2040 pixels at 90 FPS, and consists of 1000 consecutive frames, each manually labeled. The datasets contain 5 and 10 fish. The input parameters are determined by using 300 training samples.

The algorithm is evaluated by:

1. Precision - correctly tracked objects out of all tracked objects
2. Recall - correctly tracked object out of all ground-truth objects
3. Mostly Tracked Trajectories (MT) - correctly tracked objects for more than 80% of total length
4. Mostly Lost Trajectories (ML) - correctly tracked objects for less than 20% of total length
5. Fragments - interrupted ground-truth trajectory
6. ID Switches - change of identity of the tracked objects

The algorithm is compared with the proposed algorithm using just two camera views and with idTracker by Pérez-Escudero et al. (2014), where they apply the algorithm on the top and on the side views, and fuse the obtained trajectories.

The authors find all three algorithms obtain a similar high Precision rate, but that idTracker scores lower on the Recall rate by up to 10 percentage points. Moreover, idTracker scores poorly on the MT and ML measures, while also having a large number of identity switches and trajectory fragments. Comparatively, the proposed algorithm obtains a perfect score on the MT and ML measures, and only a few identity switches and fragments. The two-view version performs slightly worse with more fragments and ID switches. The authors argue that using three views will help correctly determine the trajectories when the fish are occluded in the two other views.

In defense of idTracker, the authors explain that the appearance of the fish changes greatly when observed in 3D (rather than in 2D). These changes require a more complex tracking model, for maintaining an accurate performance: idTracker needs at least 5 minutes of data to build a proper “fingerprint” for each fish, which was not available in the test and could lead to a worse performance. The authors suspect the idTracker would perform better if it had been given more training data, to pick up the subtleties of fish appearance in a 3D environment.

As the authors suggest, while the algorithm obtains good results, it is not capable of learning from the video data, which leads to accumulating errors over longer durations. Lastly, since the 3D multi-target problem is so complex, they are currently only capable of tracking a few fish, for a short duration of time.

2.1.2 Convolutional Neural Network Approaches

Wang et al. (2017) propose a CNN-based 2D tracking method based on detection and identification of fish head patterns, in every frame. To assist with the association step, they combine the CNN results with motion characteristics of the fish.

Xu and Cheng (2017) present a 2D tracking model of zebrafish based on CNN. Even though fish identities might be difficult to distinguish by the human eye, the authors believe the CNN can pick up the right features and correctly identify the fish. The method is based on the shape and the position of the fish head region, which authors refer as fish "identification photo".

Methods

To deal with illumination changes, Wang et al. (2017) propose an online CNN approach, where the sample pool is adjusted accordingly. They argue that CNN can pick up the subtle differences between two fish, and hence, perform well on the identification task. The complete model is divided into two parts: detection and tracking; and the final output is a combination of the calculated motion state and CNN identifications.

In the detection step, the authors find each fish, by using a scale-space *Determinant of Hessian* (DoH) head detector, proposed by Qian et al. (2014). These images are filtered out based on the area, scale, and intensity. The result is the images of fish head aligned to the same position and orientation.This way the image data is more clean for training a CNN. To verify that the images are actual images of fish heads, the authors use a *Histogram of Oriented Gradients* (HOG) (Dalal and Triggs, 2005) - based *Support Vector Machine* (SVM) classifier, where features are extracted from positive and negative patches (i.e. fish head versus anything else).

The tracking step takes the head images as inputs for the CNN. The head images are weighted using a mask based on the mean of all head images. The authors used a rather simple CNN structure, to predict the fish identity of the head sample: the input patch of shape 48×48, three convolutional layers, three subsampling layers, and a softmax layer. The authors argue that the association part cannot be dependent only on the CNN outcome, as it will lead to tracking errors; hence, they combine the CNN identification results with motion characteristics of the fish. Due to the small displacement of the fish in high frame rates, they treat it as uniformly distributed data.

In case of an identification or no detection due to occlusions, an SVM classifier verifies if the patch predicted by the motion characteristics is a real fish head. If it is not, the trajectory is interrupted and re-linked afterwards, as a post-processing step.

The training data is a sensitive part of the CNN training, which defines how fast and accurate the network will learn. The authors use a semi-automatic technique for data annotation: they detect and track fish by using Qian et al. (2014)'s algorithm, and manually correct it afterwards. Also, they train on the same video iteratively, which boosts their network performance.

Xu and Cheng (2017) suggest that from the top view a fish head can be represented as a BLOB, since the pixel intensities are darker than in other parts of the fish, and the head shape is nearly elliptical. They use the scale-space DoH algorithm by Qian et al. (2014), together with the scale and intensity thresholding, to obtain the BLOB of the fish head. As a preprocessing step, the authors binarize and rotate the head patch (see *Figure: 2.2*). The training images are augmented by random shifting in horizontal and vertical directions.

To create the training dataset, the authors define that: 1) each trajectory segment should be corresponding to only one identity and 2) the number of identities is priorly known. If the start point of the candidate

Figure 2.2: Pre-processing steps of thresholding and adjusting the angle, to create the dataset by Xu and Cheng (2017). Source: (Xu and Cheng, 2017, Fig. 4)

trajectories is within the threshold radius of the segment end point and do appear in the following 200 frames, then the authors choose a segment with the smallest difference of the overlapped head regions from two frames. The algorithm is repeated up until the number of trajectories correspond to the amount of the identities.

The model architecture consists of four convolutional layers with the filter size of 3, two fully-connected layers and a softmax output. The network is trained from scratch for 30 epochs. To extend the dataset, they review new candidate segments and their IDs: (if a candidate has only a single ID, the latter is removed from other segments with multiple IDs). After this iterative process is done, if the number of segments correspond to the number of fish, the segments are added to the training dataset.

When the final CNN is trained, each trajectory segment get assigned an ID. To link these trajectories, first, they define a base trajectory (a segment with a smallest number of frames of the specific ID), find the end point of it, and find all candidates within the next 2000 frames with the same ID and within a specified radius of the end point. The segment that has the smallest frame number is connected to the base, and becomes the base itself. The process is iterative, until no more segments available and all IDs are processed.

Results

Wang et al. (2017) evaluate the model on five videos of zebrafish, with 10 - 49 fish in a video, and various degree of illumination change and noise (i.e. stones on the bottom of the tank). The camera and the light source are both placed above the tank. The authors make sure that each fish has at least 300 head image patches for CNN training. Wang et al. (2017) only slightly explain identification evaluation, as it is highly connected to the detection part of the model. They suggest that the identification is not altered by illumination, nor noise; however, it is strongly affected by occlusions (as detections do too). The combined detection and identification results show that the highest Miss Ratio (percent of fish undetected in all frames) is 1.19%; the highest Error Ratio (percent of wrongly detected fish in all frames) is 1.04%; and the lowest Identification Rate (on average, per dataset) is 74.4%. The tracking part was compared to other state-of-the-art methods of tracking fish, namely iDTracker (Pérez-Escudero et al., 2014) and the algorithm proposed by Qian et al. (2014). The proposed method outperformed the other two, when the number of fish increased, hence prove to deal better with more occlusions.

Xu and Cheng (2017) employ five datasets, which varies in fish density (from 5 to 25 zebrafish). In total, they use the same evaluation metrics as Qian and Chen (2017), as well as the F1-measure (the harmonic mean of precision and recall), to compare the tracking system performance with the annotated data. They define a correct tracking to be if x and y coordinates are no further than 70 pixels from the ground truth. The lowest precision they obtained is 98.3% and the highest is 99.9%. However, the frequency of ID switches varies by the datasets. The lowest amount of ID switches they get 6 switches, and 48 switches at the highest.

Summary

From the investigated literature on tracking of zebrafish, it was found that in all cases the top camera view was used for identifying the fish and linking the tracklets. The fish were in most cases discriminated using their head in some way, and the video data was recorded in monochrome.

In all cases the side of the fish have not been primarily used, ignoring the potential useful color features and texture structure of the fish body. While the fish can appear similar to the human eye, at closer inspection there are clear differences in their patterns and color. We are therefore interested in looking into the area of person re-identification, which deals with the problem of associating people, that can have similar appearances, over time and different camera views, for inspiration.

2.2 Person Re-Identification

Person re-identification (Re-ID) is a large and growing field, with a lot of new research being published regularly. The field consists of two subfields: image-based methods and video-based methods. In both subfields, the area of metric learning is used extensively. We divide the literature survey into two parts: an image-based person Re-ID and video-based person Re-ID.

2.2.1 Image-Based Person Re-Identification

When investigating the image-based person re-identification approaches, we split the survey into two method sections: feature representation and metric-learning, as to make it easier to get an overview.

Feature Representation

Xiao et al. (2016) propose an algorithm where several person re-identification datasets, with underlying distributions, are used to train a single network, in a multi-domain approach. The network is called JSTL, as it deals with the *joint single-task learning* problem. This way they hope to learn distinct generic feature representations (even when there is no large dataset available for the specific problem), which can then be fine-tuned for each dataset.

The proposed method utilizes a CNN consisting of several InceptionV3 modules proposed by Szegedy et al. (2016), which classifies samples from all the datasets at the same time. This is done using a single softmax layer, with a fully-connected layer before it. They train a feature extractor where the output is similar for images of the same identity and otherwise different. The network first trained until convergence on the datasets. Dropout (Srivastava et al., 2014) with $p = 0.5$ is applied to the fully-connected layer. After the JSTL network has been trained, Xiao et al. (2016) apply a novel algorithm called *Domain Guided Dropout* (DGD) in order to determine which neurons in the fully-connected layer actually contribute to each dataset, and which ones are useless, based on their *impact score*.

From the impact scores, a binary mask, $\mathbf{m}$, is generated per dataset to mask out irrelevant nodes. The authors propose two ways of generating the mask: a deterministic and stochastic approach. The deterministic approach sets $\mathbf{m}_i$ to 1, if the impact score s_i is above 0, and otherwise 0. The stochastic approach draws each value from a Bernoulli distribution.

The JSTL network is then again trained, this time with the deterministic DGD mask applied, resulting in a JSTL+DGD network. Finally, a dataset specific version of the JSTL+DGD is fine-tuned per dataset by applying stochastic DGD, which results in the network performing better per dataset, but sacrificing the ability to generalize to different datasets. These networks are denoted FT-JSTL+DGD.

Wu et al. (2016) propose a CNN-based feature extractor, where the final feature representation, is regularized by an ensemble of hand-crafted features. The idea is that hand-crafted color and texture features encode information which is important for the cross-view person re-identification task, and therefore can be used to guide the CNN features, to create an even better feature representation.

The proposed network is called *Fusion Feature Net* (FFN). The authors use the *CaffeNet* architecture (Donahue, 2014) (which is an adaption of the AlexNet architecture (Krizhevsky et al., 2012)) connected to a 4096-dimension fully-connected layer, resulting in what is called the CNN-features, and adapt the *Ensemble of Localized Features* (ELF) from Gray and Tao (2008), to include more kinds of features. The authors adapt the ELF descriptor by splitting the input image into 16 horizontal patches, and for each patch calculate color features in six color spaces, as well as with Gabor (Fogel and Sagi, 1989), Schmid (Schmid, 2001) and *Local Binary Pattern* (LBP) (Ojala et al., 1994, 2002) texture descriptors.

Each of the feature channels are represented as a 16-dimensional histogram, which are L_1 normalized, and concatenated per patch. The patch histograms are then concatenated to give a 9216-dimensional feature vector for the input image[1]. This feature representation is denoted ELF16. The CNN-feature and ELF16 are then each connected to separate 4096-dimensional fully-connected layers, and both of these layers are connected to a single 4096-dimensional fully-connected layer, which gives the final feature output.

Metric Learning

Köstinger et al. (2012) propose a supervised Mahalanobis metric-based approach, which deals with some of the scalability issues encountered when working with large datasets. One method to deal with large datasets is to utilize an iterative approach, which can be computationally expensive if working on dataset of several thousand data points. Köstinger et al. (2012) instead propose a non-iterative and parameter free approach, based on an equivalence constraint i.e. on whether the compared samples are from the same class or not. The approach is named KISSME after the *Keep It Simple and Straightforward* principle.

The approach is based on statistical point of view, where determining if $\mathbf{x}_i$ and $\mathbf{x}_j$ are dissimilar, is investigated in the pairwise difference space given by $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$ with label y_{ij} which is 0 for dissimilar points and 1 for similar points. Since the pairwise difference is symmetric, this space has a mean value of 0.

They find that the Mahalanobis matrix $\mathbf{M}$ is obtained as *Equation: (2.1)*, if the pairwise difference space is assumed to be Gaussian.

$$\mathbf{M} = \mathbf{\Sigma}_{+}^{-1} - \mathbf{\Sigma}_{-}^{-1} \tag{2.1}$$

Where $\mathbf{\Sigma}$ is the covariance matrix, with a subscript + for similar points, and subscript - for dissimilar points.

Since this approach does not require any parameter fitting and is non-iterative, it scales well to large datasets. It is however still limited by the feature dimensions, and the utilized features are therefore first dimensionality reduced through *Principal Component Analysis* (PCA).

Liao et al. (2015) propose a new algorithm consisting of a new feature representation as well as an extension of the KISSME algorithm. They observe that KISSME is in its foundation just a variation of *Quadratic Discriminant Analysis* (QDA). They develop an algorithm called *Cross-View Quadratic Discriminant Analysis* (XQDA), where they utilize cross-view data (data from two viewpoints) to learn a subspace projection, $\mathbf{W}$, as well as a distance function in this subspace. The method can also be used for single-view data. The optimal subspace projection $\mathbf{W}$ is then found by maximizing *Equation: (2.2)*.

[1] The paper states the feature vector is 8064-dimensional. We cannot get this to match with the amount of filters described. It does however match the ELF variation used by Chen et al. (2015), which is cited as an inspiration.

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{\Sigma}_- \mathbf{w}}{\mathbf{w}^T \mathbf{\Sigma}_+ \mathbf{w}} \tag{2.2}$$

Where $\mathbf{w}$ is the projection direction and column vector of $\mathbf{W}$.

In addition to the XQDA algorithm, Liao et al. (2015) also propose a new feature representation called *Local Maximal Occurrence* (LOMO). LOMO is a patch-based feature extractor, which creates a histogram for the joint-HSV values and the *Scale Invariant Local Ternary Pattern* (SILTP) (Liao et al., 2010) features. The histograms for each row of patches are concatenated to create the final feature representation.

Yang et al. (2016) propose another extension of the KISSME algorithm, called *Large Scale Similarity Learning* (LSSL). LSSL is based on the observation that for similar points, the norm of the difference, $\mathbf{e} = \mathbf{x}_i - \mathbf{x}_j$, is small, while the norm of the commonness, $\mathbf{m} = \mathbf{x}_i + \mathbf{x}_j$, is high (and inverse for dissimilar points). This is under the assumption the points have been L_2 normalized. The authors therefore propose to use both the difference and commonness to create a more discriminative similarity measure, by subtracting the dissimilarity measure (difference) from the similarity measure (commonness), as shown in *Equation: (5.12)*.

$$s_{LSSL}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{m}^T \mathbf{A} \mathbf{m} - \lambda \mathbf{e}^T \mathbf{B} \mathbf{e} \tag{2.3}$$

Where $\mathbf{A}$ and $\mathbf{B}$ are the similarity and dissimilarity matrices, respectively, and λ balances them. The authors also found that these matrices can be calculated purely from the similar points, making the computation of the matrices faster and simpler.

Zhang et al. (2016) propose learning a *Discriminative Null Space* (DNS) to deal with the *small sample size* problem, which occurs when there are much fewer data samples than feature dimensions, where all samples from each identity are collapsed onto a single point per identity. This is done by using the *Null Foley-Sammon Transformation* (NFST) (Guo et al., 2006), which is an extension of LDA. The NFST algorithm finds a projection $\mathbf{W}$, where the column vectors $\mathbf{w}$ fulfill the following inequalities for the within-class scatter $\mathbf{S}_w$, and the between-class scatter $\mathbf{S}_b$: $\mathbf{w}^T \mathbf{S}_w \mathbf{w} = 0$, and $\mathbf{w}^T \mathbf{S}_b \mathbf{w} > 0$. This will according to *Equation: (2.2)* result in the best possible separation of the c identities.

The proposed method can be extended to work with kernel functions, in order to deal with nonlinearities in the input space, as well as used for semi-supervised learning using an iterative update and assignment scheme.

Ahmed et al. (2015) propose a unique siamese CNN structure, see *Figure: 2.3*, where two input images are first sent through two convolutional and max-pooling layers, with shared weights. They then utilize a set of unique layers which can be interpreted as part of the systems loss function.

The unique layers consists of a *cross-input neighborhood difference* layer, to get a rough understanding of the relationship of features between the two input images, by comparing just the local neighborhoods. These relations are summarized in a *patch summary* layer, which generates a high level representation of the neighborhood differences. Lastly, an *across patch* layer, which learn the spatial relationships between neighborhood differences. Finally a global representation is generated by flattening and concatenating the across-patch feature maps, connecting it to a fully-connected layer, and finally connect it to a 2-node softmax layer.

For training, Ahmed et al. (2015) try to circumvent the imbalanced dataset problems which comes with it being easier to generate negative pairs than positive pairs. This is done by randomly choosing only twice as many negative pairs compared to positive pairs. The network is then trained until it converges. All negative pairs are then classified with the trained network, and the worst negative pairs are used to fine-tune the fully-connected layer of the network (this time only using as many negative as positive pairs). This process is called *hard negative mining*.

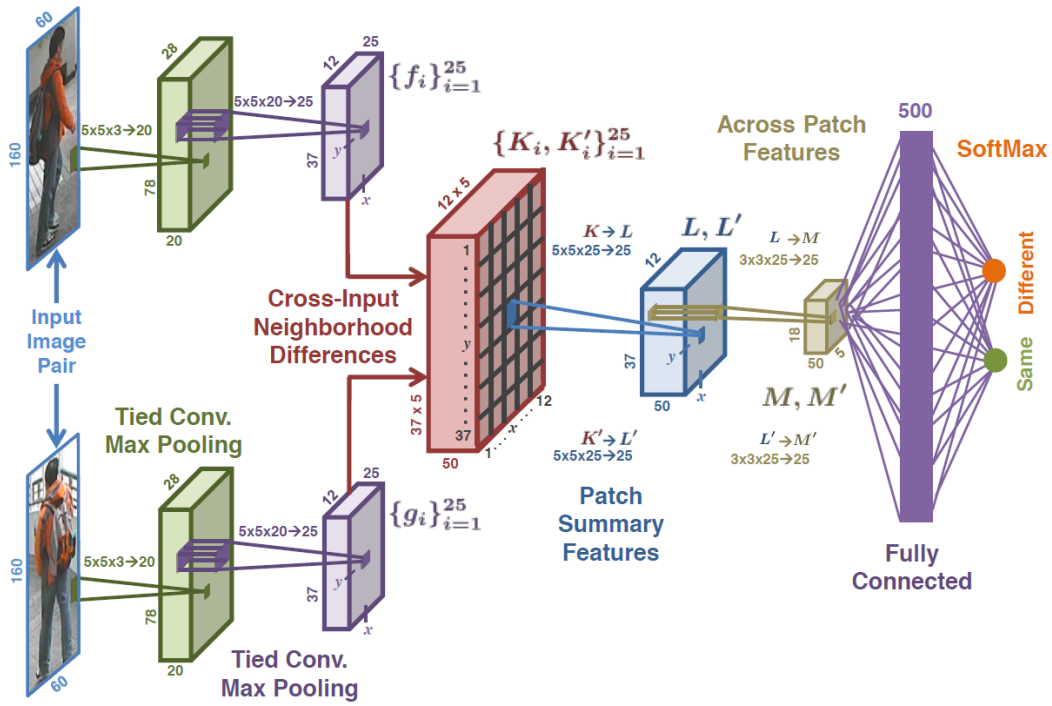


Figure 2.3: Illustration of the person re-identification network proposed by Ahmed et al. (2015). Source: (Ahmed et al., 2015, Fig. 2)

Zhu et al. (2017) propose a new deep learning approach which tries to approximate the discriminative capabilities of the Mahalanobis metric, through observing that the L_2 metric and cosine similarity are just simplifications of the Mahalanobis metric. The proposed approach utilizes a siamese network, with a novel hybrid similarity layer shown in *Equation: (2.4)*.

$$d_H(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{W}_d^T |\mathbf{x}_i - \mathbf{x}_j| + \mathbf{W}_m^T (\mathbf{x}_i \odot \mathbf{x}_j) \tag{2.4}$$

Where $\odot$ is the Hadamard product, which performs an element-wise multiplication of the input vectors, and the $d \times 1$ matrices $\mathbf{W}_d$ and $\mathbf{W}_m$ are the trainable weights for projecting from a d-dimensional vector to a scalar value. This approach only uses $2d$ parameters and tries to approximate the Mahalanobis metric, which would have required d^2 parameters. Zhu et al. (2017) utilizes a logistic loss function, to optimize their network, which transforms the Re-ID problem into a classification problem.

The siamese network used is a small CNN, which consists of three convolutional (Conv), batch normalization (Ioffe and Szegedy, 2015) and max pooling layers, applied in succession, one at a time. The ReLU activation is applied after the batch normalization layers. The layers start with 32 features maps, and doubles for each iteration. After the last max pooling layer an average pooling layer with a 1×6 kernel and stride of 1 is applied, resulting in a 16×1 output per feature map. This horizontal averaging is performed to make the network more robust to different viewpoints. The 128 16-dimensional feature maps from the average pooling layer are concatenated, resulting in a 2048 dimensional vector per branch in the network, and supplied to the hybrid similarity layer described in *Equation: (2.4)*, to obtain a similarity value for the input pair.

Bak and Carr (2017b) developed an patch-based one-shot metric learning (OSML) approach, which requires significantly fewer training examples compared to other supervised methods. This is done by utilizing a Mahalanobis-based metric, where the learned matrix $\mathbf{M}$ is split into two independent components, texture

Figure 2.4: The utilized ColorChecker chart for one given camera, rotated 90 degrees clockwise. Source: (Bak and Carr, 2017b, Fig. 4)

and color, by constructing $\mathbf{M}$ as shown in *Equation: (2.5)*.

$$\mathbf{M} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{G} \end{bmatrix} \tag{2.5}$$

Where $\mathbf{I}$ is the identity matrix, which corresponds to calculating the Euclidean distance between the texture features, and $\mathbf{G}$ is a color metric matrix which is determined from a one-shot approach.

The texture features are obtained by training a CNN on images from multiple datasets as proposed by Xiao et al. (2016), but only using intensity images, as it is found that the network then generalizes better and does not have to be fine-tuned per dataset. The color features are generated by computing a set of color space histograms, per patch, and applying PCA.

The color metric matrix, $\mathbf{G}$, is a matrix learned based on the principles of the KISSME approach, where the covariance matrices of the positive and negative pairwise differences, $\mathbf{\Sigma}_+$ and $\mathbf{\Sigma}_-$, are computed.

The $\mathbf{\Sigma}_-$ matrix is computed from randomly sampled color features from each camera view, as it is determined that the chance of picking features from the same identity at a frequent rate is negligible. The $\mathbf{\Sigma}_+$ requires supervision to ensure that only color features of the same identity is used. Bak and Carr (2017b) models the color features from a camera by using explicit terms for the foreground, σ, and background, ϵ, instead of assuming that the metric will learn to differentiate the background from the foreground.

The positive pairwise covariance matrix thereby becomes $\mathbf{\Sigma}_+ = \mathbf{\Sigma}_{\sigma,+} + \mathbf{\Sigma}_{\epsilon,+}$. $\mathbf{\Sigma}_{\epsilon,+}$ is computed by sampling background images from a camera pair. The background samples are gathered by sampling from the patches with background elements, which do not intersect the foreground element. $\mathbf{\Sigma}_{\sigma,+}$ is computed through the use of a *ColorChecker* calibration chart, see *Figure: 2.4*, which encodes the necessary information about the color distribution of the camera pair.

The model is improved further by splitting the input image into n patches, making the color metric matrix $\mathbf{G}$ patch-dependent, so that the spatial variances and deformation of the patches can be modeled. Furthermore, a detector dependent parameter $\alpha^{(n)}$ is a learned parameter per patch, which describes the influence of the background for the patch. As the values is detector dependent, it only has to be determined using a single dataset.

To handle different poses, which cause patch position in a fixed patch grid to not correspond to the same features, the patches in one bounding box are allowed to be mapped to different patches in the second bounding box. This is done by building a $n \times n$ cost matrix, consisting of a similarity value between the two patches, as long as they are within the same neighborhood of each other. Otherwise the value is set to ∞. The construction of the cost matrix is shown in *Figure: 2.5*. The linear assignment problem is solved by applying the Hungarian algorithm (Kuhn, 1955).

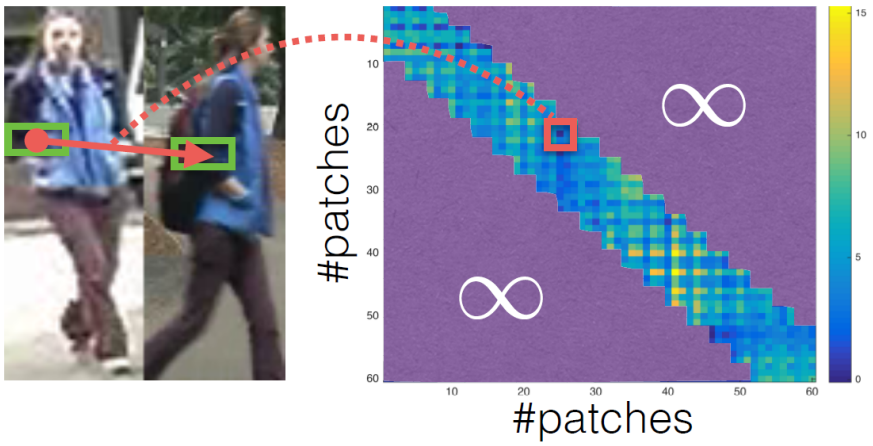


Figure 2.5: Illustration of the cost matrix for associating patches within some neighborhood. Source: (Bak and Carr, 2017b, Fig. 3)

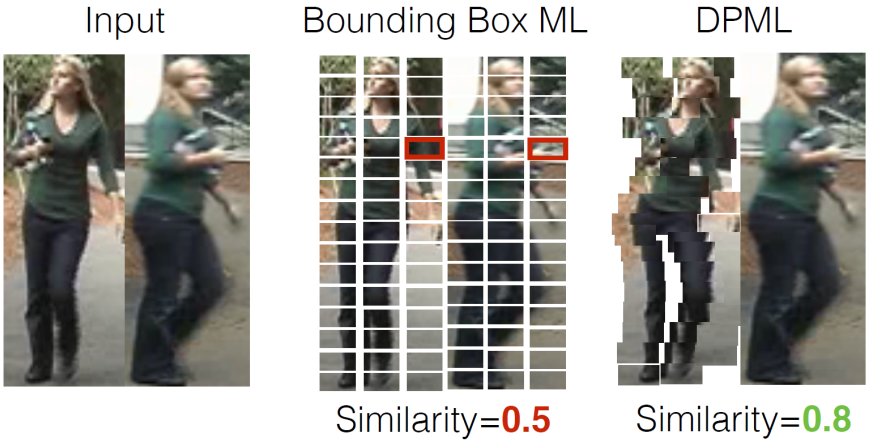


Figure 2.6: Comparison of the PML and DPML algorithms. PML is denoted as *Bounding Box ML*. Source: (Bak and Carr, 2017a, Fig. 1)

For the color features the image is scaled to 128×48, from which overlapping patches of size 12×24 with a stride of 6×12 are constructed, resulting in 60 patches. For the CNN texture extractor the images are scaled to 160×64 to fit the JSTL architecture.

Bak and Carr (2017a) propose a deep deformable patch based approach for the person Re-ID problem. The work is in part similar to that of Bak and Carr (2017b), though it presents and evaluates several approaches. The authors propose that instead of learning a global metric, then a metric should be applied per patch, where dissimilarity scores are then integrated through some function. Each patch need not have a unique metric, but may share with neighboring patches. Additionally, by sharing metrics the amount of training data is effectively increased. A simple approach to this, which the authors call the *Rigid model* or PML, would be to simply sum the dissimilarity score over all patches, where the patch dissimilarity is computed by comparing the patches in the two images, that are in the same position in the "patch grid" (See middle column of *Figure: 2.6*).

It can however be beneficial to allow the patches to "move", meaning that the compared patches does not necessarily have the same position in the patch grid, resulting in a deformable approach. The authors use an unsupervised approach, which was also used in Bak and Carr (2017b), where a $n \times n$ cost matrix

is constructed. The final dissimilarity value is then the sum of the dissimilarity scores of the associated patches. This model is called *Unsupervised Deformable Model* and denoted as HPML by the authors.

The authors also propose a *Supervised Deformable Model* denoted DPML by the authors, where the model translates the patches in a continuous fashion, which approximates non-affine warps, as shown *Figure: 2.6*. This method uses a set of *spring constraints* α_n, which determine how much a patch is allowed to move, as well as a set of per patch weights, w_n, stored in the vector $\mathbf{w}$.

The weights $\mathbf{w}$ and spring constraints α_n are determined through a two-step process, where first the optimal α_n is found through an exhaustive grid search when $\mathbf{w} = \mathbf{1}$ by maximizing the Rank-1 recognition rate, and then $\mathbf{w}$ is determined by using Structural SVMs (Joachims et al., 2009). The number of unique spring constraints are limited, and in the paper the authors use just two constraints, meaning the constraint are shared across patches.

Lastly, the authors propose the use of a CNN feature extractor, which has been trained on patches instead of entire images. The CNN used is a JSTL-based model, where the last feature layer is 256-dimensional. The input is constructed by splitting each image into 8 non-overlapping patches, and assign each patch position a unique ID. This way the CNN has to learn both the person ID and the patch position, which results in a powerful feature representation.

Results

The described image-based methods are tested and trained on a wide variety of datasets. However, it is only for the VIPeR dataset that all methods are evaluated on, and therefore the discussion of the methods will focus on this dataset.

The specifications of the VIPeR dataset is shown in *Table: 2.1*. All of the datasets are evaluated by using a *probe* and a *gallery*, where the *probe* image is compared to all the *gallery* images. The methods are evaluated by how accurate they are when the correct gallery image has to be in the top $n\%$ of the ranked gallery images. If there is only one image of each ID in the gallery and only one probe image, then it is considered a *single-shot* approach, and otherwise it is called *multi-shot* approach. Single-shot methods are evaluated through a *Cumulative Matching Characteristic* (CMC) curve, which reports the accuracy when the correct gallery image is within the top-n ranked images, denoted Rank-n. For multi-shot approaches the *mean Average Precision* (mAP) is also reported, as it more accurately reflects how the correct gallery images are ranked.

VIPeR is evaluated using a single-shot approach, therefore only the CMC score for top-n ranks is reported. The results of the methods when evaluated on the VIPeR dataset are shown in *Table: 2.2*.

Table 2.1: VIPeR dataset description

Dataset	#Images	#ID	ID split	Label method
VIPeR (Gray et al., 2007)	1264	632	316/0/316	Manual

The KISSME algorithm by Köstinger et al. (2012) is only evaluated on the VIPeR dataset. The dataset does not have a predetermined training and test dataset, so the authors split the dataset in half, and report their results as an average over 100 independent runs. Each image is split into a set of patches and described by a LAB and HSV histogram, with 24 bins per channel, and a LBP histogram. The histograms are concatenated and reduced to a 34-dimensional vector through PCA.

Table 2.2: Ranking accuracy results for the VIPeR dataset. The results are listed in chronological order.

Dataset	Method	Rank-1	Rank-5	Rank-10	Rank-20
VIPeR	KISSME	19.60%	-	62.20%	-
	XQDA+LOMO	40.00%	-	80.51%	91.08%
	(Ahmed et al., 2015)	34.81%	-	-	-
	DNS	51.17%	82.09%	90.51%	95.92%
	DNS (semi)	41.01%	69.81%	81.61%	91.04%
	LSSL	47.78%	77.90%	87.60%	94.20%
	FFN+LOMO	51.06%	81.01%	91.39%	96.90%
	FT-JSTL+DGD	38.60%	-	-	-
	DHSL	44.87%	-	86.01%	93.70%
	DHSL (100 IDs)	30.79%	-	72.78%	83.92%
	OSML	34.30%	-	-	-
	DPML-CNN	51.70%	-	-	-
	HPML-CNN	48.20%	-	-	-
	PML-CNN	47.00%	-	-	-

The XQDA algorithm by Liao et al. (2015) used the LOMO feature descriptor, and the results are averaged over 10 runs.

Ahmed et al. (2015) first pre-trained their model on two larger datasets, CUHK01 (Li et al., 2013) and CUHK03 (Li et al., 2014b), and fine-tuned the network on the VIPeR training data.

Zhang et al. (2016) tested their DNS approach, while using the LOMO feature descriptor as well as a 5138 dimensional feature descriptor based on the concatenated RGB, LAB, HSV (without the Value component), HOG and LBP histograms. The results were averaged over 10 runs. They found that when fusing the scores from the two feature representation, their algorithm performs better.

Yang et al. (2016) only tested the LSSL algorithm on the VIPeR dataset. They represent the images by splitting the image into 10 horizontal stripes, and for each calculating the *Salient Color Names Based Color Descriptor* (SCNCD) (Yang et al., 2014) and fusing it with color histograms and color names (van de Weijer et al., 2009), for the following color spaces: RGB, *rgb*, $l_1l_2l_3$, HSV, YCbCr and LAB. PCA is applied to reduce the dimensionality of the features to 75.

The FT-JSTL+DGD networks from Xiao et al. (2016) are trained on the Shinpuhkan (Kawanishi et al., 2014), VIPeR (Gray et al., 2007), CUHK01 (Li et al., 2013), CUHK03 (Li et al., 2014b), i-LIDS (Zheng et al., 2009), 3DPeS (Baltieri et al., 2011) and PRID2011 (Hirzer et al., 2011) datasets. For the proposed model the ranking is computed by calculating the Euclidean distance between the feature vector obtained from the layer before the softmax layer, for each of the images.

Wu et al. (2016) fine-tuned their FFN on the Market-1501 Zheng et al. (2015) dataset, as the network is pre-trained on the ImageNet dataset (Russakovsky et al., 2015). Hard negative mining proposed by Ahmed et al. (2015) is also used to train the network. The proposed method only focuses on the feature extractor which can be used with any metric comparison method. The authors use the *Mirror Kernel Marginal Fisher Analysis* (M-KMFA) algorithm by Chen et al. (2015). The dataset is split into equally-sized training and testing datasets, and the training dataset is used to learn the M-KMFA algorithm, on the extracted features. They find that they get better results when combining the features with the LOMO feature representation by Liao et al. (2015), resulting in a 31056 dimensional feature vector, indicating that the FFN and LOMO feature representations are complimentary.

The DHSL algorithm by Zhu et al. (2017) split the dataset in equal halves, for training and testing, and reported results over 10 runs. For training, the data is augmented by randomly rotating the image by ± 3 degrees.

The one-shot metric learning approach from Bak and Carr (2017b) tests on four established datasets: VIPeR, i-LIDS, CUHK01, and PRID2011, as well as their own dataset CCH. For training, all four dataset are used as well as the CUHK03 dataset. The CCH dataset is primarily used to compare the proposed method with other color calibration methods. An image of the ColorCheck Chart is captured by each camera, and used to calculate color metric matrix $\mathbf{G}$. The proposed algorithm performs competitively with the DNS method. It is also competitive, while not beating, with the supervised approaches.

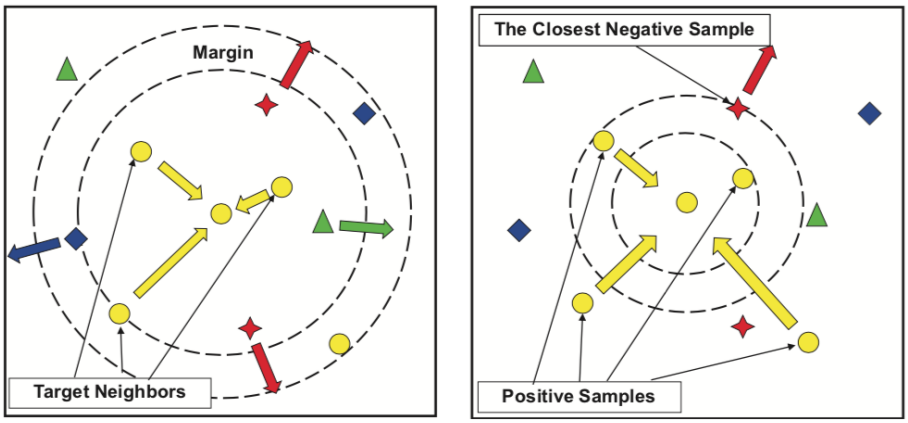
Bak and Carr (2017a) conduct a thorough analysis of their proposed algorithms. They find that the deep CNN features result in a 8 - 29% increase over hand-crafted features, and that their patch approach performs better than using a single metric for the entire image.They also investigate the performance of the KISSME, XQDA and LSSL algorithms. The results show that using LSSL consistently performs the best, and is therefore used for the results in *Table: 2.2*. The HPML and DPML models have the disadvantageous of having to solve either the Hungarian algorithm (HPML) or determine the spring constraints and weights (DPML). The Hungarian cost matrix will however most likely be sparse and can be evaluated quickly, resulting in the VIPeR evaluation to take 330 seconds, with the CNN feature extraction taking the most time, while for DPML it takes 30 minutes. The PML, while performing worse than HPML and DPML, is much faster and still performs competitively, when compared to other methods.

2.2.2 Video-Based Person Re-Identification

In person re-identification, if a sequence of frames is available, a video-based method can be utilized. For this type of approach, the research is not as extensive as for the image-based approaches, and does not vary in methods as much. The common practice is to use a *Recurrent Neural Network* (RNN) for series of inputs, based on their spatiotemporal features, combined with feature-extractors in a form of CNN or hand crafted features (McLaughlin et al., 2016; Sadeghian et al., 2017). Because RNNs often require a large amount of data, another approach that uses metric learning is surveyed.

You et al. (2016) propose a *top-push distance learning* (TDL) method for video-based re-identification, as a way to optimize the learning of the most discriminative features of people, and learn the Mahalanobis matrix $\mathbf{M}$. Additionally, You et al. (2016) argue that besides the appearance features that are usually extracted in image-based approaches, the temporal changes may give more finer details on people walking patterns, their clothes moving, etc. Furthermore, the sequences provide an abundance of appearance cues too, which minimizes occlusion and background impacts, since more data samples exist. This way video-based methods can be beneficial for small datasets.

The main idea is to use a top-push constraint that was first introduced by Li et al. (2014a), and extended by the authors from a linear ranking function to a second-order function, by investigating the latent subspace of the features. The TDL algorithm iteratively looks at the two closest between-class points, and minimizes an



(a) LMNN (b) TDL

Figure 2.7: Comparison of LMNN and Top-Push Distance Learning Model. Source: (You et al., 2016, Fig. 3)

Table 2.3: Dataset overview

Dataset	Avg. #Images per sequence	#ID	ID split	Label method
iLIDS-VID (Wang et al., 2014)	73	300	150/0/150	Manual
PRID2011 (Hirzer et al., 2011)	100	200	100/0/100	Manual

objective function which pull all same-class points of one of the point closer, while pushing the other point away, where after the matrix $\mathbf{M}$ is updated. This is repeated until a convergence criteria is reached. The authors look into feature representations of people: they use HOG3D (Klaser et al., 2008) for spatiotemporal features; and color histograms and LBP for appearance features. They apply average pooling on the latter set of features, on all video frames.

The authors see the spatiotemporal and appearance features, as complimentary, which can help with challenges present in a video, i.e. occlusions and background differences. Specifically, the frames are resized to 128×48 pixels, and divided into patches of 8×16 with a 50% overlap in both directions. From the resulting 155 patches the color histograms of the HSV and LAB color spaces, and LBP descriptor were calculated. The image input was constructed by concatenating the average of the color histograms, LBP and HOG3D features.

The work by You et al. (2016) is comparable to Large Margin Nearest Neighbor classification (LMNN) (Weinberger et al., 2006) and LDA methods. LMNN is an extension of k-nearest neighbor algorithm, where a Mahalanobis distance metric is learned. The goal is to have the k-nearest neighbors be the same class, and separate them from data points of other classes by a large margin. However, the difference between LMNN and the proposed method lies in how strict the penalty of close impostors is (see Figure 2.7), as TDL only pushes the closest imposer value away, while pulling all true values closer. This constraint aims to heavily penalize the impostors that are the closest to the positive samples.

Results

You et al. (2016) found that they improved the state-of-the-art results, starting from Rank-1, at the time of publication, when testing on the datasets described in *Table: 2.3*. First, You et al. (2016) test the model with video-based approaches, where they yield 56.74% in Rank-1 and 56.33% in Rank-1, for the PRID2011 and

the iLIDS-VID datasets, respectively. You et al. (2016) also look investigate an image-based matching, where the authors randomly select 5 images per person and represented without any spatiotemporal information, they get significantly worse results, which shows that combining motion and appearance features together yield the best performance of the model.

2.3 Summary

Based on the conducted literature review, it is found that there has been little investigation of how to re-identify zebrafish reliably while tracking it, and that the work that has been conducted has focused on a top view with monochrome images. Meanwhile, a lot of work has been put into the field of person re-identification, for both image- and video-based methods, focusing on re-identifying people over time through their color features and, for video, their movement pattern. Specifically, the field of metric learning is widely used for person re-identification. It is therefore interesting to look into the use of metric learning combined with color side view images of the fish, as these have not been investigated before, in order to re-identify the fish.

3 | Problem Statement

The analysis from *Section 2.1.1* and *Section 2.1.2* point out successful methods for tracking and identifying zebrafish, by using traditional and deep learning methods. Even though the results are satisfactory, some drawbacks still persist, and need to be worked on further. One of them is an amount of data required. CNNs, as being leading state-of-the-art in many computer vision problems, expect very large amounts of annotated data, to work effectively. Moreover, even traditional methods, like idTracker (Pérez-Escudero et al., 2014), need long video sequences, to achieve the best performance (i.e. minimum 5 minutes of video, as suggested by the authors). This adds complexity to the system, both storage- and computation-wise.

To the best of our knowledge, all of the literature within the zebrafish tracking area focuses on top-view of the fish, using only its head to distinguish two or more fish identities. This research will focus on gathering data from a side-view, and examining whether the visually distinct patterns can potentially be used for re-identifying a number of fish.

Finally, *Section 2.2* concludes that metric learning can effectively separate pool of features of n classes, even for smaller datasets. This is desirable for the current research project, since obtaining and annotating lots of data is not feasible within the allocated time period.

Above considered, the research question is formulated as follows:

Through computer vision and machine learning, how can a set of zebrafish be reliably re-identified based on a single camera view from the side?

4 | Dataset

This chapter presents the utilized datasets, their collection process and analysis, as well as the camera, lens and lighting considerations made.

For the dataset collection, a $32 \times 32 \times 32$ cm large tank was used and in order to limit the size of the setup it was decided that the camera setup should be within a meter from the fish tank.

4.1 Imaging System Considerations

In the following section, we determine the requirements for the imaging system and the camera and lens choice.

4.1.1 Camera and Lens Considerations

In the problem statement defined in *Chapter 3*, it is stated that it is necessary to use a single camera, filming the fish tank from the side. Furthermore, it is decided that when filming the tank, the width of the tank should fill the entire video frame, in order to not "waste" pixels on uninteresting elements near the fish tank. Also, the camera should record in color, as opposed to the approaches described in *Section 2.1*. This is so that the potentially useful colorful appearance of the zebrafish is not lost.

(Rowena et al., 2007, p. 15) find that the zebrafish rarely grow longer than 4 cm. The height of the fish is not stated, but for simplicity it is assumed the fish is 1 cm tall. Based on this the amount of pixels occupied by the fish, as well as an analysis of motion blur can be produced.

Camera Analysis

If the entire side of the tank, of size 32×32 cm, or 1024 cm^2 is contained exactly within the video frame, then a fish of size 1×4 cm, or 4 cm^2 takes up $\frac{4}{1024} = 0.0039\%$ of the pixels. That is, if the frame is 1000×1000 pixels (1 megapixel), then the fish will occupy $0.0039 \times 1000 \times 1000 = 3900$ pixels. When comparing to the size of images used in the VIPeR person re-identification dataset ($128 \times 48 = 6144$ pixels), this is deemed too low, and that at least twice the number of pixels should be available.

Bengtson and Pedersen (2017) found that the approximate maximum speed of the zebrafish is 70-90 $\frac{cm}{s}$, with an approximate average speed of 9-13 $\frac{cm}{s}$ in a stressful environment. Then, with a shutter speed of $\frac{1}{30}$ seconds, which is the maximum shutter exposure which can be used together with the common 30 Hz frame rate, the fish will at maximum move $90\frac{cm}{s}\frac{1}{30}s = 3.00cm$ per sensor exposure and on average $13\frac{cm}{s}\frac{1}{30}s = 0.433cm$ per sensor exposure. This is too much, as the fish could potentially move approximately 10% of the length of the tank, in just a single exposure, resulting in a large amount of blur. This can be mitigated by using a shutter speed at least twice as fast, and by using a global shutter. Furthermore, Bengtson and Pedersen (2017) suggest recording with a high frame rate, as to make the displacement smaller between frames, and therefore potentially have fewer errors. Therefore, the frame rate should be at least 60 Hz. This requirement is introduced so that the dataset can also be used for tracking, even though it is not specifically needed for re-identification.

Ideally, the chosen camera has a C-mount, due to availability of the compatible lenses in the *Visual Analysis of People* (VAP) lab at AAU.

Lens Analysis

A wide array of lenses can be used. In order to circumvent potential problems with distortion, it is desirable to choose a lens with a longer focal length, resulting in a more narrow *field-of-view* (FoV). This means that the camera has to be placed further away from the tank, but also that less of the sides of the tank will be visible through the front side of the tank, as described by Audira et al. (2018). A longer focal length will however also result in a smaller *depth-of-field* (DoF). Furthermore, a prime lens should be used, instead of a zoom lens, as prime lenses in general allow for larger apertures. As the shutter speed will be quite high, a large aperture is expected, in order to allow enough light to hit the camera sensor. A large aperture will however also result in a lower DoF. A large focal length lens may therefore not be preferable.

Lastly, the lens should ideally have the same sensor size, or larger, than that of the chosen camera. If the lenses sensor format is larger, then it will result in a more narrow FoV than advertised. The lens should also be rated to match the megapixel of the camera, as the lens might otherwise focus the incoming light onto a spot which spans across several pixels, resulting in a blurred image.

Camera and Lens Requirements

The following requirements for the camera and lens can be made based on the previous considerations.

Camera system

1. The entire width of fish tank must be in the field of view of the camera.
2. The camera and lens setup must at maximum be 1 meter from the fish tank.

Camera

3. A color camera must be used (*Chapter 3*).
4. A single camera must be used (*Chapter 3*).
5. A global shutter must be used (*Section 4.1.1*).
6. The image quality of the output must be at least 2 megapixels (*Section 4.1.1*).
7. The minimum frame rate must be at least 60 Hz (*Section 4.1.1*).
8. The maximum shutter speed must be 1/60 seconds (*Section 4.1.1*).
9. The camera should ideally have a C-mount (*Section 4.1.1*)

Lens

10. The lens must be a prime lens (*Section 4.1.1*).
11. The lens must be rated for the megapixels of the camera (*Section 4.1.1*)
12. The sensor format of the lens must not be smaller than that of the camera (*Section 4.1.1*)

4.1.2 Camera and Lens Selection

Based on the requirements set in *Section 4.1.1*, several cameras were considered. The initial set of candidate cameras were:

- Point Grey Grasshopper 3 (GS3-U3-41C6C-C) [1]
- IO Industries Victorem (51B163-CX) [2]
- JAI GO-5000C-USB [3]
- IDS UI-3070CP Rev. 2 [4]

All four cameras meet the set requirements. However, several of them have some drawbacks which need to be accounted for. After e-mail and phone correspondences with the manufactures the list narrowed down. Point Grey informed us that the Grasshopper3 camera would not be ready for delivery before July. IO Industries stated that the Victorem camera would need an additional frame grabber and digital video recorder system, which would push the total price up towards 100000 Danish kroner. JAI told that in order to film with the camera, we would have to pay for a proprietary camera recording software or develop it ourselves, which they warned would be quite time consuming.

Figure 4.1: IDS UI-3070CP Rev. 2 Source: IDS (nd)

This meant that the only viable camera left within the budget and the available time was the IDS UI-3070CP, which was ordered after IDS confirmed that their proprietary software would be capable of recording videos. A single IDS UI-3070CP Rev. 2 camera, shown in *Figure: 4.1*, costing 714 Euro, was ordered. The ordered camera has the following specifications (IDS, nd):

- **Resolution**: 2056 × 1542 (3.1 Megapixel)
- **Max FPS**: 134
- **Exposure time**: 0.018 ms - 999 ms
- **Lens Mount**: C-Mount
- **Sensor Format**: 1/1.8”
- **Shutter Type**: Global Shutter
- **Connection Type**: USB 3.0

[1] https://www.ptgrey.com/support/downloads/10146
[2] http://ioindustries.com/Victorem%20CoaXPress%20Rev2.1_Web.pdf
[3] http://www.jai.com/en/products/go-5000-usb
[4] http://en.ids-imaging.com/store/products/cameras/ui-3070cp-rev-2.html

Table 4.1: Field of View of the three investigated lenses for 1” and 1/1.8” camera sensor format. All values are in degrees

Lens	LM8HC		LM16HC		LM25HC	
Camera Sensor Format	1”	1/1.8”	1”	1/1.8”	1”	1/1.8”
Vertical FOV	63.0	36.9	33.6	19.0	22.0	12.4
Horizontal FOV	79.4	48.5	44.3	25.3	29.3	16.5
Diagonal FOV	92.4	59.5	54.3	31.5	36.6	20.6

Table 4.2: Distance between camera setup and fishtank for the different lenses. All values are in centimeters

	LM8HC	LM16HC	LM25HC
a (cm)	35.52	71.29	110.35

After selecting the camera, the lens sensor format could be decided on to be at minimum 1/1.8”.

Three prime lenses available from the VAP lab were considered, all of which have a sensor format 1”, are rated up to 4 megapixels resolution, have an f-stop range of 1.4 - 16, and have a C-mount. The three lenses are all from KOWA: LM8HC (KOWA, ndc), LM16HC (KOWA, nda) and LM25HC (KOWA, ndb), which have a respective focal length of 8 mm, 16 mm and 25 mm.

In order to ensure that both requirement 1 and 2 are fulfilled, we calculated how far away the camera should be to have the entire fish tank width within a view. Since the camera’s sensor format is smaller than the sensor format of the lens, the lens will have a lower effective FOV. According to the lens data sheets, the FOV for a 1” and 1/1.8” camera sensor format is shown in *Table: 4.1*.

Using the horizontal FOV and the law of sines (see *Equation: (4.1)*), while assuming the camera setup is perpendicular to the fish tank, the distance between the camera and fish tank is found as per *Equation: (4.1)*.

$$\frac{a}{\sin A} = \frac{b}{\sin B} = \frac{c}{\sin C} \tag{4.1}$$

Where C is the angle between fish tank and camera, B is the horizontal FOV of the camera setup, b is half the width of the fish tank, and a is the distance between the camera setup and the fish tank. It is known that C is 90 degrees, as the camera setup and fish tank are perpendicular, B is half the horizontal FOV of the camera setup and b is equal to 16 cm for the used fish tank. a can therefore be found by restructuring *Equation: (4.1)* into *Equation: (4.2)*.

$$a = \frac{b}{\sin B} \sin A \tag{4.2}$$

While A is not explicitly given, it can easily be calculated as $A = 180 - C - B$.

The distances found by applying *Equation: (4.2)* on the horizontal FOV values in *Table: 4.1*, are shown in *Table: 4.2*. Based on the calculated distances, the KOWA LM16HC lens was chosen, as it fulfills both requirement 1 and 2 from *Section 4.1.1*, while also avoiding a wide angle lens, which is favorable as described in *Section 4.1.1*.

4.2 Setup and Data Acquisition

It was decided to initially perform the re-identification task with data acquired in a laboratory setting, as to see whether it is possible in the close to ideal settings. Therefore, the data acquisition setup was designed so that we had complete control over the light, and no occluding elements in a fish tank were present.

The fish tank used is a $32 \times 32 \times 32$ cm clear glass tank, with a clear acrylic plate inserted, as to limit the movement of the fish along the depth of the tank. This way the potential problems with different scales of the fish, due to different depth distances, are significantly limited. Furthermore, by restricting the amount of space the fish can move in the depth of the tank, we reduce the amount of time the fish can swim at an angle to the camera, and instead constrain it to swim perpendicular to the camera. This was done by placing the divider plate so that the fish had 3.5 cm to swim in.

In order to control the lighting of the test, we recorded in a windowless room, while using two Diva-Lite 401-230 lamps (Kino Flo, 2017a), which utilize florescent tubes. The lamps consist of 4 pairs of tubes and run at an output frequency of 40 kHz, resulting in a flickerless recording. It is possible to only use the two center tube pairs, resulting in a light output equal to 1 f-stop lower while keeping the same color temperature (Kino Flo, 2017b, p. 2). The exact color temperature of the light is not known, and was not measured as the white balance was adjusted accordingly. We used the 2 tube pair configuration, as it resulted in fewer reflections on the fish. The lights were placed at an angle from each side of the tank, 89 cm away from the tank.

Furthermore, the fish tank was placed inside a "photography tent", which is a box with a single open side, made of white nylon. This results in a diffused light on the fish, allowing us to adjust the light to reduce reflections further, while keeping the fish well lit.

The camera was placed 70.5 cm from the tank, where the width of the tank fills the frame, and set at such a height that the bottom of the tank was at the bottom of the frame. The tank was then filled with water, until it reached 10.5 high, so the top of the water was approximately in the middle of the frame, resulting in fewer reflection in the water, and since fish have tendency to jump in the stressful situations.

The setup is shown in *Figure: 4.2*. The angle of the lamps were not noted down.

We recorded videos with 2 and 3 fish in the tank at a time. This way it would be possible to first of all test whether it is possible to differentiate two fish, and if so, increase complexity. Furthermore, a training and testing dataset were recorded for both 2 and 3 fish. The training and testing datasets use the same amount of, but different, fish.

The data was recorded using the *IDS Camera Manager* software, which directly supports video recording with the chosen camera, and saves it as lossless AVI video. The color balance was adjusted, and the exposure set to 9.175 milliseconds. The exposure was chosen by letting the software try and automatically select a value, while keeping the image well lit, which we then adjusted if it was too high. During the recording session, we found that the software does not allow recording at the maximum frame rate at full resolution. Instead, it records at a variable frame rate, but still with the set exposure time. Therefore, we set the software to record 1000 frames per video, resulting in different frame rate per video. However, the software did not record exactly 1000 frames, but potentially more, resulting in videos containing different amount of frames. This means that the requirement 7 from *Section 4.1.1* is not fulfilled, and that the dataset is not suitable for tracking. An overview of the 4 datasets are shown in *Table: 4.3*.

Before the videos were recorded, the fish were given at minimum 15 minutes for rest, as to calm them down. This helped them to be less stressed or anxious about the new environment, and thereby also move in a less erratic manner.

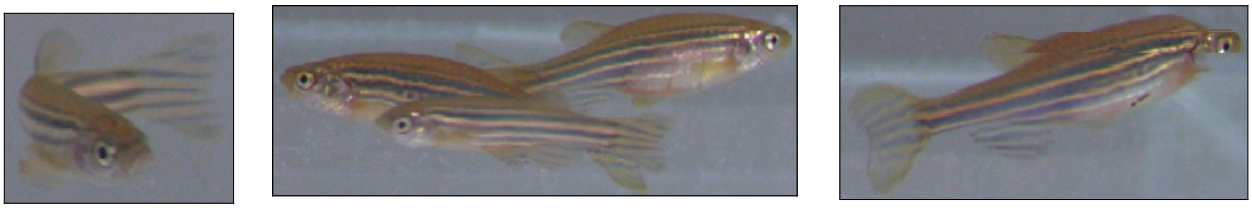
Figure 4.2: Data acquisition setup

Table 4.3: Fish dataset overview

	2-FISH-TRN	2-FISH-TST	3-FISH-TRN	3-FISH-TST
# Images	1084	1138	1086	1138
Frame rate (1/s)	11.45	10.93	11.20	10.84
Video length (s)	94	104	96	105

4.3 Annotations

Since the project does not focus on the detection part, it is necessary to annotate the area which the fish occupy. This is done utilizing the *AAU VAP Bounding Box Annotator* program by Møgelmose and Bahnsen (2018). A fish were only annotated when it swam parallel to the camera, and was not occluded or excessively blurred from a sudden acceleration. Furthermore, the recording software would at times cause "glitches", where the frame was only partially updated. IDS confirmed it to be a software bug through mail correspondence. Examples of each scenario which were purposefully not annotated are shown in *Figure: 4.3*. While annotating, each fish got a specific ID and the orientation of the fish was noted (i.e. left or right), together with starting frames of the fish rotation or occlusion.



(a) Turning **(b)** Occluding **(c)** Glitch

Figure 4.3: Scenarios where the zebrafish are not annotated

While all annotations were made so that the annotator believed it fitted as closely to fish as possible, we added a 5 pixels border to the annotations afterwards. This is so that if any part of the fish are slightly outside the bounding box, they are still included, while also not allowing too much of the background to be in the annotation bounding box.

4.4 Pre-processing

When the fish swim around in the tank, they often swim at an angle. In order to test how large an effect this angle will have on the re-identification, a version of all the datasets was created, where the fish were rotated so they are parallel with the x-axis. An example of this is shown in *Figure: 4.4*.

This was done in two steps:

1. Background subtraction and BLOB extraction
2. Determining angle and rotating bounding box

Lastly, a version of the datasets where the bounding boxes are flipped around the vertical axis was also created, to investigate the potential impact of the differing orientations.

4.4.1 Background Subtraction and BLOB Extraction

To rotate the fish, it is first of all necessary to determine which parts inside the bounding box is potentially the fish. This is done through a basic background subtraction process, where we subtract a background image from the image we are investigating, see *Equation: (4.3)*.

$$\mathbf{I}_{diff} = |\mathbf{I} - \mathbf{I}_{bg}| \tag{4.3}$$

Where $\mathbf{I}$ is the input image, $\mathbf{I}_{bg}$ is the background image calculated as a median image over 100 randomly selected images from the dataset, and $\mathbf{I}_{diff}$ is the resulting image.

(a) Original bounding box **(b)** Bounding box after rotation

Figure 4.4: Effect of pre-processing the bounding boxes

(a) Original binary mask **(b)** Binary mask after rotation

Figure 4.5: Binary masks of the bounding boxes in *Figure: 4.4*. Dark blue is equal to a value of 0, and yellow a value of 1.

The reason a median image of the dataset is used for $\mathbf{I}_{bg}$, is due finding that the background images which were captured during the data acquisition, did not work for all datasets. This is due to small changes in the frame when changing the fish, which were not noticed during the data acquisition.

$\mathbf{I}_{diff}$ is then averaged across all channels, and binarized using a thresholding operation. Through empirical testing we found that a threshold value of 3 performed well, resulting in low amount of noise BLOBs and a majority of the actual fish area. Not all of the fish is correctly binarized, since some of the fins of the zebrafish are semi-transparent. It is however not problematic if these are left out, since the general direction is still obvious from the remaining BLOB. This can be seen in *Figure: 4.5*, where the binary image of the bounding boxes in *Figure: 4.4* is shown.

Remaining noise BLOBs and holes in BLOBs are then removed, by applying morphological opening with a 7×7 circular structuring element. The largest BLOB inside the bounding box is then extracted and assumed to be the BLOB of the fish.

4.4.2 Determining Angle and Rotating Bounding Box

When the largest BLOB is found, a 2×2 covariance matrix, $\mathbf{C}$, of the pixel coordinates is determined. The eigenvalues and eigenvectors of $\mathbf{C}$ are then found. The eigenvector, $\mathbf{v}$, with the largest eigenvalue, corresponds to the direction where the BLOB varies the most, which will be a vector along the length of the fish. The angle between this eigenvector and the x-axis will therefore describe the angle of the fish, θ, which can be calculated through *Equation: (4.4)*.

$$\theta = \tan^{-1}(\frac{\mathbf{v}_y}{\mathbf{v}_x}) \tag{4.4}$$

$\mathbf{I}$ is rotated by $-\theta$ degrees around the center of the BLOB. The new rotated bounding box is then extracted by finding the closest fitting bounding box to the rotated BLOB.

4.5 Analysis

In order to extract features from the fish, it is nearly always necessary to standardize the image size to some predefined size. Therefore, the mean height and width bounding boxes in the datasets were calculated, as well as the median bounding box. The total amount of orientations was also investigated, both per dataset and per fish in the datasets. The results of this analysis are shown in *Table: 4.4* and *Table: 4.5*.

Table: 4.4 shows that when the data is rotated as per *Section 4.4*, the mean height and its standard deviation lowers significantly, as expected. The mean width and its standard deviation does not change significantly between the original and rotated datasets. This corresponds to our observations that the fish in general have a similar height, while the width can vary significantly due to its movements.

Since the bounding box dimensions have a quite high standard deviation, especially the width, it was decided to resize the images to the median bounding box size, to avoid the effect of outliers. This way the images are transformed as little as possible, when resizing. The 2-FISH-TST and 3-FISH-TST datasets were resized to the median bounding box size of 2-FISH-TRN and 3-FISH-TRN, respectively, in order to make the datasets compatible.

Due to the occasional glitches caused by the recording software, the annotations do not result in complete tracklets for the fish, between occlusions and rotations. The low frame rate also means that the tracklets are in general shorter. This is reflected in *Table: 4.6*, where the amount of occlusion and rotation events are counted, as well as the mean tracklet length before each occlusion event. The tracklet length is calculated as the amount of frames between the onset of the investigated occlusion and the onset of the closest previous rotation or occlusion event.

Table 4.4: Overview over the dataset annotations, including the number of annotations facing left and right, as well as the mean and median bounding box height and width in pixels. For the mean bounding box dimensions, the standard deviations are also listed. L and R describe the amount of annotations facing left and right, respectively, and W and H are shorthand for width and height.

Dataset		Total	L/R	Mean H	Mean W	Median H	Median W
2-FISH-TRN	Org.	1460	712/748	148.9 ± 32.33	321 ± 17.21	149	324
	Rot.			102 ± 6.86	331.0 ± 15.15	101	334
2-FISH-TST	Org.	1373	693/680	134.8 ± 34.01	309.3 ± 28.54	127	312
	Rot.			99.3 ± 7.54	319.3 ± 26.80	98	318
3-FISH-TRN	Org.	1660	970/690	130.2 ± 35.99	299.1 ± 40.41	131	298
	Rot.			91.0 ± 10.07	306 ± 41.82	93	310.5
3-FISH-TST	Org.	1880	874/1006	126.8 ± 22.30	328.1 ± 20.66	123	334
	Rot.			104.7 ± 8.40	331.8 ± 19.12	103	337

Table 4.5: Overview of the orientation and amount of rotation events of each fish in the datasets. The ID stated is per dataset, meaning that e.g. the ID 1 does not correspond to the same fish across datasets

Dataset	Fish ID	Total	L/R	# Rotations
2-FISH-TRN	1	748	386/362	17
	2	712	362/350	21
2-FISH-TST	1	687	379/308	26
	2	686	314/372	18
3-FISH-TRN	1	631	360/271	7
	2	539	329/210	36
	3	590	281/209	16
3-FISH-TST	1	617	313/304	29
	2	684	302/382	31
	3	579	259/320	24

Table 4.6: Overview of the amount of occlusion events per dataset. The mean tracklet length before each occlusion event is also found, together with the standard deviation.

Dataset	# Occlusions	Tracklet length before occlusion (frames)
2-FISH-TRN	15	15 ± 11.18
2-FISH-TST	20	10.75 ± 7.10
3-FISH-TRN	34	10.03 ± 6.50
3-FISH-TST	53	7.72 ± 4.60

5 | Methods

The following chapter describes the methods used for the project in theoretical details. First, the general concept of metric learning is explained, followed by descriptions of the investigated metric learning methods. The tested feature descriptors are explained too. Selection of both metric learning and feature descriptors is inspired by person re-identification field, reviewed in *Chapter 2*.

5.1 Metric Learning Methods

In this section, we give an introductory knowledge on the metric learning concept, and describe the selected methods: Keep It Simple and Straightforward (Köstinger et al., 2012), Cross-View Quadratic Discriminative Analysis (Liao et al., 2015), Large Scale Similarity Learning (Yang et al., 2016) and Discriminative Null Space (Zhang et al., 2016).

5.1.1 Metric Learning Primer

A central problem for classification and ranking problems is to determine whether two inputs (e.g. two faces or two pedestrian images) are similar or not. This can be done through distance metric or similarity measure. A distance metric is a pairwise real-valued function that obeys the following conditions (Bellet et al., 2015, p. 7):

1. Non-negative: $f(\mathbf{x}, \mathbf{y}) \geq 0$
2. Symmetric: $f(\mathbf{x}, \mathbf{y}) = f(\mathbf{y}, \mathbf{x})$
3. Triangle inequality: $f(\mathbf{x}, \mathbf{z}) \leq f(\mathbf{x}, \mathbf{y}) + f(\mathbf{y}, \mathbf{z})$
4. Identity of indiscernibles: $f(\mathbf{x}, \mathbf{y}) = 0$ iff. $\mathbf{x} = \mathbf{y}$

In contrary, a similarity measure is not strictly defined and can range from negative to positive values. Moreover, a high value would associate with a high similarity, while a low value would indicate dissimilar inputs.

Often systems focus on obtaining good feature descriptions of the data and comparing the data with some predefined metric. However, this approach leads the feature representation and the deciding metric to be disjoint and not optimized together. This happens when the system, for example, uses an L_p-norm, *Equation: (5.1)*, or the cosine similarity, *Equation: (5.2)*, to decide how similar the inputs are.

$$L_p(\mathbf{x}, \mathbf{y}) = ||\mathbf{x} - \mathbf{y}||_p = \left(\sum_{i=1}^{d} |x_i - y_i|^p \right)^{\frac{1}{p}} \tag{5.1}$$

$$\cos Sim(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}^T \mathbf{y}}{||\mathbf{x}||_2 \ ||\mathbf{y}||_2} \tag{5.2}$$

The cosine similarity can be converted into a "distance" by stating $\cos Dist(\mathbf{x}, \mathbf{y}) = 1 - \cos Sim(\mathbf{x}, \mathbf{y})$, which will be in the range $[0; 2]$. The cosine distance is however not a distance metric as the triangle inequality is not fulfilled in all cases (Korenius et al., 2007).

It is therefore interesting to look into the topic of learned metrics, which allow the data to affect the metric.

Mahalanobis Metric

One of the fundamental learned metrics is the Mahalanobis metric, (Bellet et al., 2015, p. 9), which can be used to determine whether a point $\mathbf{p}$ belongs to a distribution D with a mean $\boldsymbol{\mu}$ and a covariance matrix $\boldsymbol{\Sigma}$. This method measures the similarity between two d-dimensional vectors, $\mathbf{x}$ and $\mathbf{y}$, from the same distribution with covariance matrix $\boldsymbol{\Sigma}$, see *Equation: (5.3)*.

$$d_M(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \mathbf{y})} \tag{5.3}$$

$\mathbf{M}$ is often used as a shorthand for $\boldsymbol{\Sigma}^{-1}$ and will be used from now on. If $\mathbf{M}$ is the identity matrix, it is reduced to the Euclidean distance (i.e. L_2 norm), and if it is a diagonal matrix, then it is seen as the normalized Euclidean distance.

Furthermore, the inverse covariance matrix $\mathbf{M}$ is a real-valued positive semi-definite (PSD) matrix, in which all of its eigenvalues have a value of 0 or above. This also implies that the fourth metric requirement is not necessarily fulfilled (e.g. a zero-valued matrix is also a PSD matrix and it maps two distinctly different inputs to 0). The Mahalanobis metric is, therefore, a *pseudometric*, but this is often disregarded.

The $\mathbf{M}$ matrix can be decomposed as $\mathbf{M} = \mathbf{L}^T\mathbf{L}$, (Bellet et al., 2015, p. 10). $\mathbf{L}$ has the dimensions $k \times d$, where k is the rank of the matrix $\mathbf{M}$. If k is smaller than d, then $\mathbf{L}$ represents a linear projection into a lower dimensional space. *Equation: (5.3)* can be rewritten, as shown in *Equation: (5.4)*. While by the exact definition of the Mahalanobis metric, $\mathbf{M}$ is the covariance matrix, it is at times relaxed to $\mathbf{M}$ being some real-valued PSD matrix:

$$\begin{aligned} d_M(\mathbf{x}, \mathbf{y}) &= \sqrt{(\mathbf{x} - \mathbf{y})^T \mathbf{L}^T \mathbf{L} (\mathbf{x} - \mathbf{y})} \\ &= \sqrt{(\mathbf{Lx} - \mathbf{Ly})^T (\mathbf{Lx} - \mathbf{Ly})} \\ &= ||\mathbf{Lx} - \mathbf{Ly}||_2 \end{aligned} \tag{5.4}$$

From *Equation: (5.4)* it is clear that the Mahalanobis metric is simply the Euclidean distance after a linear transformation, described by $\mathbf{L}$, of the feature space.

There are however disadvantages with the Mahalanobis metric, which corrupt its real-life applications. $\mathbf{M}$ is an unknown matrix which has to be computed. This can be computationally expensive (if the feature dimensions are large) and time-consuming (if the dataset is large), as it needs to estimate d^2 parameters. One can instead estimate $\mathbf{L}$, which potentially requires fewer parameters, but this can end up being a non-convex optimization problem. Besides, since it is a linear operation, it cannot necessarily capture the potential nonlinear complexity of the data.

The research interest lies in finding alternative approaches which help alleviate the disadvantages of the Mahalanobis metric, and find other ways to define the similarity between data points.

5.1.2 Keep It Simple and Straightforward Metric

The KISSME algorithm by Köstinger et al. (2012) is based on statistical point of view, where determining if $\mathbf{x}_i$ and $\mathbf{x}_j$ are dissimilar, is seen as a likelihood ratio test, see *Equation: (5.5)*. This test is stated as by comparing the null and alternative hypothesis (H_0 and H_1, respectively) in the *pairwise difference space*. This space is given by $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$ with label y_{ij} which is 0 for dissimilar points and 1 for similar points. Since the pairwise difference is symmetric, this space has a mean value of 0.

$$\delta(\mathbf{x}_{ij}) = \log \frac{p(\mathbf{x}_{ij}|H_0)}{p(\mathbf{x}_{ij}|H_1)} = \log \frac{f(\mathbf{x}_{ij}|\theta_0)}{f(\mathbf{x}_{ij}|\theta_1)} \tag{5.5}$$

Where f is some function parametrized by θ.

If the hypothesis space is assumed to be Gaussian, then f is a multinomial Gaussian distribution, parametrized by $\theta_0 = (\mathbf{0}, \mathbf{\Sigma}_-)$ and $\theta_1 = (\mathbf{0}, \mathbf{\Sigma}_+)$, with subscript + and - denoting similar and dissimilar pairs. Köstinger et al. (2012) find that maximizing the likelihood, is the same as minimizing the Mahalanobis distance.

Köstinger et al. (2012) further find that the matrix $\mathbf{M}$ can be calculated as *Equation: (5.6)*.

$$\mathbf{M} = \mathbf{\Sigma}_+^{-1} - \mathbf{\Sigma}_-^{-1} \tag{5.6}$$

$\mathbf{M}$ is then projected onto the cone of PSD matrices. This is done by applying eigen decomposition, see *Equation: (5.7)*, to get the eigenvectors, $\mathbf{Q}$, and eigenvalues, $\boldsymbol{\lambda}$, of $\mathbf{M}$, setting all the negative eigenvalues to 0, and reconstructing $\mathbf{M}$, again through *Equation: (5.7)*.

$$\mathbf{M} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{-1} \tag{5.7}$$

Where $\mathbf{\Lambda}$ is a diagonal matrix, with the eigenvalues $\boldsymbol{\lambda}$ on the diagonal. As the original Mahalanobis matrix $\mathbf{M}$ is symmetric and real, $\mathbf{Q}$ will be orthogonal and the transpose can be used instead of the inverse.

Theoretically, all dissimilar points should be used, but since the number of dissimilar points will often be quite large and greatly outnumber the amount of similar points, $\mathbf{\Sigma}_-$ is usually calculated using only as many dissimilar points as similar points, which are chosen at random.

As mentioned in *Chapter 2*, this approach does not require any parameter fitting and is non-iterative, meaning it scales well to large datasets, but requires the features to be dimensionality reduced using PCA in order to be efficient.

5.1.3 Cross-View Quadratic Discriminative Analysis

The XQDA algorithm proposed by Liao et al. (2015) is based on the observation that KISSME is in its foundation a variation of *Quadratic Discriminant Analysis* (QDA). They propose an extended variation called XQDA, which utilize cross-view data (data from two viewpoints, denoted $\mathbf{X}$ and $\mathbf{Z}$) to learn a subspace projection $\mathbf{W}$, as well as a distance function in this subspace. The method can also be used for a single-view data. This is done by estimating the matrix $\mathbf{M}$, based on $\mathbf{W}$, as shown in *Equation: (5.8)*:

$$\mathbf{M}(\mathbf{W}) = \mathbf{W}(\mathbf{\Sigma}_+^{'-1} - \mathbf{\Sigma}_-^{'-1})\mathbf{W}^T \tag{5.8}$$

Where $\mathbf{\Sigma}_-^{'} = \mathbf{W}^T\mathbf{\Sigma}_-\mathbf{W}$ and $\mathbf{\Sigma}_+^{'} = \mathbf{W}^T\mathbf{\Sigma}_+\mathbf{W}$.

The optimal subspace projection $\mathbf{W}$ is found by maximizing the *Generalized Rayleigh Quotient* (GRQ), shown in *Equation: (5.9)*, which will lead to a large variance between dissimilar points, and small variance for similar points.

$$J(\mathbf{w}) = \frac{\mathbf{w}^T\mathbf{\Sigma}_-\mathbf{w}}{\mathbf{w}^T\mathbf{\Sigma}_+\mathbf{w}} \tag{5.9}$$

Where $\mathbf{w}$ is the projection direction, and a column vector of $\mathbf{W}$.

When maximizing *Equation: (5.9)*, the equation can be reformulated into *Equation: (5.10)*, as along as $\mathbf{\Sigma}_+$ is non-singular.

$$\begin{gathered} \arg\max_{\mathbf{w}} J(\mathbf{w}) = \arg\max_{\mathbf{w}} \mathbf{w}^T\mathbf{\Sigma}_-\mathbf{w} \\ s.t.\ \mathbf{w}^T\mathbf{\Sigma}_+\mathbf{w} = 1 \end{gathered} \tag{5.10}$$

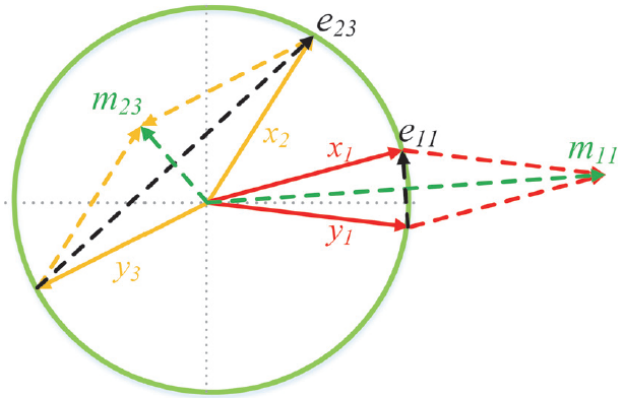


Figure 5.1: Illustration of two similar vectors $(\mathbf{x}_1, \mathbf{y}_1)$ and two dissimilar vectors $(\mathbf{x}_2, \mathbf{y}_3)$, in a 2D Euclidean space. The circle is the unit circle, and it is easy to observe the differences in the commonness and difference vectors, $\mathbf{m}$ and $\mathbf{e}$. Source: (Yang et al., 2016, Fig. 1)

Equation: (5.10) can be solved by finding the eigenvalues and eigenvectors of $\mathbf{\Sigma}_+^{-1}\mathbf{\Sigma}_-$, as it is a case of the standard eigenvector problem (Liao et al., 2015), see *Equation: (5.11)*. To find $\mathbf{W}$, the eigenvectors corresponding to the r largest eigenvalues are chosen. r is chosen so that all eigenvalues above 1 are used and at minimum the eigenvector with the largest eigenvalue.

$$\mathbf{\Sigma}_-\mathbf{w} = \lambda\mathbf{\Sigma}_+\mathbf{w} \;\Rightarrow\; \mathbf{\Sigma}_+^{-1}\mathbf{\Sigma}_-\mathbf{w} = \lambda\mathbf{w} \tag{5.11}$$

The data matrices $\mathbf{X}$ and $\mathbf{Z}$ are of shape $d \times n$ and $d \times m$, respectively, consisting of n and m d-dimensional feature vectors. When d is larger than m and n combined, it is faster to compute $\mathbf{\Sigma}_+$ and $\mathbf{\Sigma}_-$ in the QR decomposed feature space. This is done by applying QR decomposition on the stacked data matrices. This results in the orthonormal matrix $\mathbf{Q}$ of shape $d \times (n+m)$ and the upper triangular matrix $\mathbf{R}$ of shape $(n+m) \times (n+m)$. The data matrices $\mathbf{X}$ and $\mathbf{Z}$ can then be redefined by selecting the first n and last m columns of $\mathbf{R}$, respectively. This results in $\mathbf{X}$ and $\mathbf{Z}$ being of shape $(n+m) \times n$ and $(n+m) \times m$. If the QR decomposition is utilized, the subspace projection is adapted accordingly, by defining $\mathbf{W}$ as the matrix multiplication of $\mathbf{Q}$ and the selected eigenvectors.

During implementation, the practical computation proposed by Liao et al. (2015) is considered. Furthermore, in order to avoid $\mathbf{\Sigma}_+$ being singular, a small regularization value is added to the diagonal elements before inverting. Liao et al. (2015) found that a value of 0.001 worked well when the features were normalized to unit length. Lastly, since computing the inverse of large matrices can be slow and potentially numerically inaccurate, the linear system $\mathbf{\Sigma}_+\mathbf{A} = \mathbf{\Sigma}_-$ is solved, and the eigenvalues of the matrix $\mathbf{A}$ are found.

5.1.4 Large Scale Similarity Learning

LSSL, the similarity based extension of the KISSME algorithm, proposed by Yang et al. (2016), utilizes the fact that for similar feature points the norm of the difference, $\mathbf{e} = \mathbf{x}_i - \mathbf{x}_j$, is small, while the norm of the commonness, $\mathbf{m} = \mathbf{x}_i + \mathbf{x}_j$, is high (and inverse for dissimilar points), under the assumption the points have been L_2 normalized, as shown in *Figure: 5.1*. The authors therefore propose to use both the difference and commonness to create a more discriminative similarity measure, by subtracting the dissimilarity measure (difference) from the similarity measure (commonness), as shown in *Equation: (5.12)*.

$$s_{LSSL}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{m}^T\mathbf{A}\mathbf{m} - \lambda\mathbf{e}^T\mathbf{B}\mathbf{e} \tag{5.12}$$

Where $\mathbf{A}$ and $\mathbf{B}$ parametrize the similarity and dissimilarity measures, respectively, and λ balances them. Both of these matrices are *not* constrained to be PSD, as the authors state that this is not necessary for

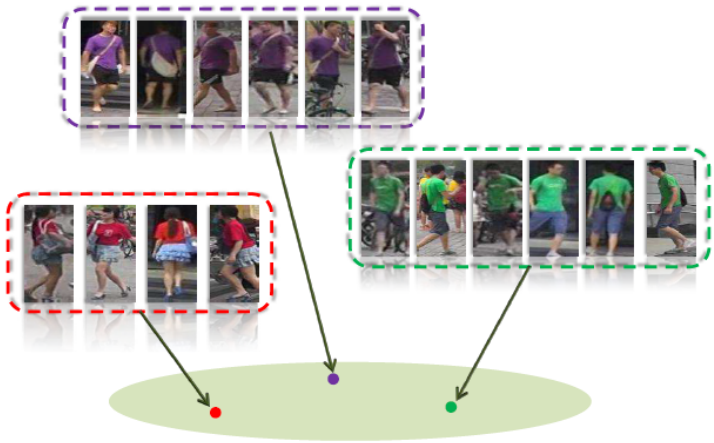


Figure 5.2: Illustration of the discriminative null space proposed by Zhang et al. (2016). Source: (Zhang et al., 2016, Fig. 1)

large datasets. The two matrices are defined by *Equation: (5.13)*. *Equation: (5.12)* and *Equation: (5.13)* can also be reformulated to explicitly include the bilinear similarity and the Mahalanobis metric, but this is left out for brevity.

$$\begin{aligned} \mathbf{A} &= \mathbf{\Sigma}^{-1} - \mathbf{\Sigma}_{\mathbf{m},+}^{-1} \\ \mathbf{B} &= \mathbf{\Sigma}_{\mathbf{e},+}^{-1} - \mathbf{\Sigma}^{-1} \end{aligned} \tag{5.13}$$

Where $\mathbf{\Sigma}_{\mathbf{m},+}$, $\mathbf{\Sigma}_{\mathbf{e},+}$, and $\mathbf{\Sigma}$ are the covariance matrices of the similar difference and commonness, and the combined dissimilar difference/commonness covariance matrix. The authors found that $\mathbf{\Sigma}$ can be calculated purely from the similar points. This means that for the LSSL algorithm, and any KISSME-based algorithm, it is only necessary to use the similar labeled points, which will also be computationally faster. This also means that the KISSME Mahalanobis matrix $\mathbf{M}$ can be computed as $\mathbf{\Sigma}^{-1} - \mathbf{\Sigma}_{\mathbf{e},+}^{-1}$. This variation of the KISSME algorithm is called the *improved KISSME* (iKISSME) algorithm. Like KISSME, LSSL also first applies PCA to the input data, as to reduce the amount of parameters to compute.

5.1.5 Discriminative Null Space

The DNS algorithm proposed Zhang et al. (2016) deals with the *small sample size* problem, which occurs when there are much fewer data samples than feature dimensions, i.e. $n \ll d$, where n is the number of samples and d is the dimensionality of the samples. DNS collapse all samples from each identity onto a single point per identity, using the NFST algorithm by Guo et al. (2006), as illustrated in *Figure: 5.2*. The NFST algorithm finds a projection $\mathbf{W}$, where the column vectors $\mathbf{w}$ fulfill the inequalities in *Equation: (5.14)*, for the within-class scatter matrix $\mathbf{S}_w$, and the between-class scatter matrix $\mathbf{S}_b$.

$$\begin{aligned} \mathbf{w}^T \mathbf{S}_w \mathbf{w} &= 0 \\ \mathbf{w}^T \mathbf{S}_b \mathbf{w} &> 0 \end{aligned} \tag{5.14}$$

This will according to the GRQ, *Equation: (5.9)*, result in the best possible separation of the c different identities.

The authors find that in order for $\mathbf{W}$ to fulfill both of the inequalities, the vectors have to lie in both the null space of $\mathbf{S}_w$, and the row space of the total scatter matrix $\mathbf{S}_t = \mathbf{S}_b + \mathbf{S}_w$. $\mathbf{S}_t$ and $\mathbf{S}_w$ are calculated as shown in *Equation: (5.15)* and *Equation: (5.16)*.

$$\mathbf{S}_t = \frac{1}{n}\mathbf{X}_t\mathbf{X}_t^T \tag{5.15}$$

$$\mathbf{S}_w = \frac{1}{n}\mathbf{X}_w\mathbf{X}_w^T \tag{5.16}$$

Where n is the number of samples, $\mathbf{X}_t$ is the zero-meaned version of the $d \times n$ input data matrix $\mathbf{X}$, and $\mathbf{X}_w$ is the per-class zero-meaned version of $\mathbf{X}$. According to Guo et al. (2006), the sample vectors are almost always linearly independent, as long as $n \ll d$, which is therefore assumed onwards.

Zhang et al. (2016) show that the vectors $\mathbf{w}$ that lie in the row space of $\mathbf{S}_t$ can be determined by first finding the orthonormal basis, $\mathbf{U}$, of $\mathbf{X}_t$. This is done by applying Gram-Schmidt Orthonormalization and representing each of the vectors $\mathbf{w}$, as shown in *Equation: (5.17)*:

$$\mathbf{w} = \mathbf{U}\boldsymbol{\beta} \tag{5.17}$$

Where $\mathbf{U}$ is the found set of orthonormal vectors, and $\boldsymbol{\beta}$ is a vector containing a weight for each vector in $\mathbf{U}$. There will be $n-1$ orthonormal vectors, as the rank of $\mathbf{S}_t$ and $\mathbf{X}_t$ is $n-1$.

The found vectors should also lie in the null space of $\mathbf{S}_w$. This can be done by substituting *Equation: (5.17)* into the inequitably for $\mathbf{S}_w$ in *Equation: (5.14)*, see *Equation: (5.18)*. This equation can be seen as equivalent to a case of the standard eigenproblem, shown in *Equation: (5.19)*, which can be solved.

$$\boldsymbol{\beta}^T\mathbf{U}^T\mathbf{S}_w\mathbf{U}\boldsymbol{\beta} = 0 \tag{5.18}$$

$$(\mathbf{U}^T\mathbf{S}_w\mathbf{U})\boldsymbol{\beta} = 0 \tag{5.19}$$

This gives $c-1$ solutions for $\boldsymbol{\beta}$. Thereby a $c-1$ dimensional projection matrix $\mathbf{W}$ is found by calculating the orthonormal vector $\mathbf{U}\boldsymbol{\beta}$, for each $\boldsymbol{\beta}$. The projected points are then evaluated with the L_2 distance.

Kernel DNS

Zhang et al. (2016) extends the DNS algorithm to work with kernel functions, to deal with nonlinearities in the input space. They utilize some kernel function k, which takes a pair of vectors, and implicitly calculates the inner product of the vectors in some high dimensional space. When done for the entire data matrix $\mathbf{X}$, this results in the symmetric kernel matrix $\mathbf{K}$ of size $n \times n$.

First, the kernalized version of $\mathbf{S}_w$ and $\mathbf{S}_t$, denoted $\mathbf{K}_w$ and $\mathbf{K}_t$, can be calculated as shown in *Equation: (5.20)*:

$$\begin{aligned}\mathbf{K}_t &= \mathbf{K}(\mathbf{I} - \mathbf{1}_n)(\mathbf{I} - \mathbf{1}_n)^T\mathbf{K} \\ \mathbf{K}_w &= \mathbf{K}(\mathbf{I} - \mathbf{L})(\mathbf{I} - \mathbf{L})^T\mathbf{K}\end{aligned} \tag{5.20}$$

Where $\mathbf{I}$ is the identity matrix, $\mathbf{1}_n$ is a $n \times n$ matrix with all elements set to $\frac{1}{n}$, and $\mathbf{L}$ is a block diagonal matrix, with a block per class of size n_c, which is the number of samples of class c, and and all block elements have a value of $\frac{1}{n_c}$. g The orthonormal basis of $\mathbf{K}_t$, denoted $\mathbf{U}_k$, is found through kernel PCA (Schölkopf et al., 1997). First $\mathbf{K}$ is centered, so that the mean of all rows, columns and all values in the matrix equal to 0, see *Equation: (5.21)*.

$$\tilde{\mathbf{K}} = (\mathbf{I} - \mathbf{1}_n)\mathbf{K}(\mathbf{I} - \mathbf{1}_n) \tag{5.21}$$

The eigenvalues and eigenvectors of $\mathbf{K}_t$ are then found by computing the eigenvalue decomposition of $\tilde{\mathbf{K}}$, see *Equation: (5.22)*, which contains $n-1$ non-zero eigenvalues.

$$\mathbf{K}_t = \mathbf{V}\mathbf{E}\mathbf{V}^T \tag{5.22}$$

Where $\mathbf{E}$ is a diagonal matrix containing the eigenvalues, and $\mathbf{V}$ are the eigenvectors.

The eigenvectors has to be scaled by the inverse square root of their respective eigenvalue, see *Equation: (5.23)*, so that the corresponding principal directions in the kernel space (which are not explicitly calculated) have unit length (Schölkopf et al., 1997).

$$\mathbf{V}_t = \mathbf{V}\mathbf{E}^{-\frac{1}{2}} \tag{5.23}$$

In order to apply the kernel PCA operation on test data, it is necessary to center the input kernel matrix. This is included in the formulation of $\mathbf{U}_k$. Therefore, the final representation of $\mathbf{U}_k$ is as in *Equation: (5.24)*.

$$\mathbf{U}_k = ((\mathbf{I} - \mathbf{1}_n)\mathbf{V}_t)^T \tag{5.24}$$

The kernel version of *Equation: (5.19)*, can then be computed, by calculating $\mathbf{K}_w$, see *Equation: (5.20)*, and substituting $\mathbf{K}_w$ from *Equation: (5.20)* and *Equation: (5.24)* into *Equation: (5.19)*. In order to make it easier to read, the substituted *Equation: (5.19)* is represented through *Equation: (5.25)* and *Equation: (5.26)*.

$$\mathbf{H} = ((\mathbf{I} - \mathbf{1}_n)\mathbf{V}_t)^T\mathbf{K}(\mathbf{I} - \mathbf{L}) \tag{5.25}$$

$$\mathbf{H}\mathbf{H}^T\boldsymbol{\beta} = 0 \tag{5.26}$$

Equation: (5.26) is then solved, resulting in $c - 1$ eigenvectors $\boldsymbol{\beta}$, and the final projection $\mathbf{W}$ can then be found by substituting into *Equation: (5.17)*, as shown in *Equation: (5.27)*.

$$\mathbf{W} = \mathbf{U}_k\boldsymbol{\beta} = ((\mathbf{I} - \mathbf{1}_n)\mathbf{V}_t)^T\boldsymbol{\beta} \tag{5.27}$$

The choice of kernel affects the output of the algorithm when using the linear kernel, see *Equation: (5.28)*, the kernelized DNS algorithm gives the same results as the original algorithm. Zhang et al. (2016) use the *Radial Basis Function* (RBF), see *Equation: (5.29)*, when utilizing the LOMO features. For the RBF kernel the adjustable parameter σ is calculated as the mean L_2 distance between the training features. A downside for the kernelized approach is that the training data has to be kept during testing, in order to compute the kernel matrix for the test data.

$$k_{Linear}(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T\mathbf{y} \tag{5.28}$$

$$k_{RBF}(\mathbf{x}, \mathbf{y}) = \exp\Big(-\frac{||\mathbf{x} - \mathbf{y}||_2^2}{2\sigma^2}\Big) \tag{5.29}$$

5.2 Feature Descriptors

In this section, an adaptations of the *Ensemble of Localized Features* (ELF) (Gray and Tao, 2008) and the *Local Maximal Occurrence* (LOMO) (Liao et al., 2015) feature descriptors are described, which were selected due to their popularity in the person re-identification field. Moreover, these feature descriptors are employing both textures and colors of the input data, whose influence is interesting to compare in the zebrafish data.

5.2.1 Ensemble of Localized Features

Gray and Tao (2008) propose to combine simple color and texture features for object representation, which they call ELF. First, they convert an image to three color spaces: RGB, YCbCr and HSV, resulting in a 8 channel color descriptive part (they only use one luminance channel, either Y or V). The authors also propose to use two texture descriptors: Gabor filters and Schmid filters, applied on the luminance channel, while using 8 and 13 parameter settings, respectively.

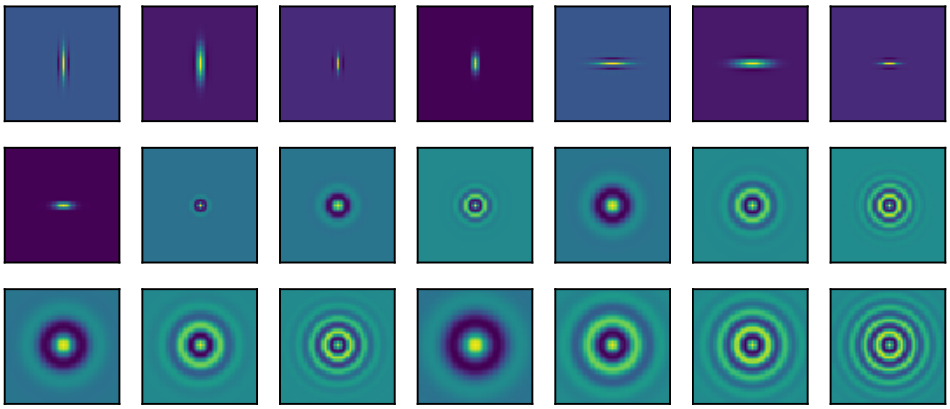


Figure 5.3: Illustration of the Gabor (the first eight images) and Schmid (the rest of the images) filters used in the ELF descriptor. The Gabor and Schmid filters are not on the same scale, but in general dark blue signifies negative values, while yellow signifies positive values.

Gabor filters (Fogel and Sagi, 1989) are rotationally sensitive filters, which in 2D corresponds to a Gaussian filter oriented and repeated according to some sinusoidal wave. The Gabor filter is a complex filter, but for ELF only the real part, see *Equation: (5.30)*, is used in the construction of the filter. The 8 parameter settings by Gray and Tao (2008) were used, and are shown in *Figure: 5.3*. These parameter settings create 2 pairs of 4 filters which are similarly oriented, but with different wavelength and width of the Gaussian filter.

$$g(x, y, \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \cos\left(2\pi \frac{x'}{\lambda} + \psi\right) \tag{5.30}$$

Where λ is the wavelength of the sinusoidal wave, θ is the orientation, ψ is phase offset, σ the standard deviation of the Gaussian filter and γ is the "aspect ratio" of the Gaussian filter i.e. how elliptical the filter is. x' and y' are calculated as shown in *Equation: (5.31)*.

$$\begin{aligned} x' &= x \cos\theta + y \sin\theta \\ y' &= -x \sin\theta + y \cos\theta \end{aligned} \tag{5.31}$$

Schmid filters (Schmid, 2001) are rotationally invariant filters, which mimics the Gabor filters. Due to the rotational invariance the filters can be used to find similar patterns in objects that have either been rotated or deformed. This allows the filters to be used in cases where the descriptor should be invariant to pose and viewpoint (Gray and Tao, 2008). The filters are calculated using *Equation: (5.32)*. The 13 parameter settings used by Gray and Tao (2008) were utilized, and the corresponding filters are shown in *Figure: 5.3*.

$$F(x, y, \tau, \sigma) = F_0(\tau, \sigma) + \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \cos\left(\pi\tau \frac{\sqrt{x^2 + y^2}}{\sigma}\right) \tag{5.32}$$

Where τ is the number of cycles of the Gaussian function, σ is the standard deviation of the Gaussian function, and $F_0(\tau, \sigma)$ is a normalization constant added to ensure the filter has a 0 DC component, i.e. the

filter response is not offset and therefore invariant to the scaling of the input intensities. F_0 is practically applied by subtracting the mean filter coefficient value from the filter.

Inspired by Wu et al. (2016), we additionally extract Locally Binary Pattern (LBP) (Ojala et al., 1994, 2002) features. *Figure: 5.4* visualizes the process of applying LBP filter onto an image. The center pixel is compared to P neighbor points which lie on a circle with a radius of R. Each value above the central pixel results in 1 and each value below or equal to the center pixel results in 0. The pixels are investigated in a clockwise manner starting at the pixel directly to the right of the center pixel. The values are concatenated creating a binary string, which is converted to a decimal representation afterwards. We chose to utilize a radius R of 1 and 8 neighbor points P.

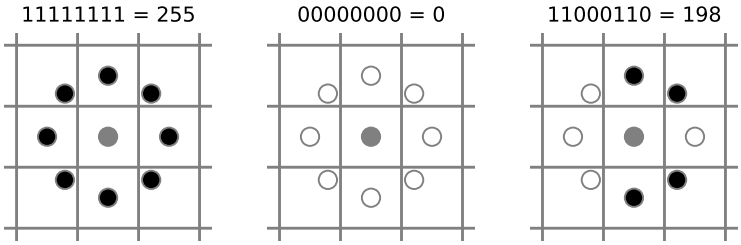


Figure 5.4: Illustration of the LBP descriptor, with a radius R of 1 and 8 points P, counting clockwise from the point to the right of the center point (gray). Black circles indicate values above the center point, and white indicates values below or equal. The binary string for each example and its decimal equivalent are shown.

For the texture filters, the input image is padded using reflected border padding, as to ensure that the output image is the same size as the input image. When all 30 channels have been extracted, the image was divided into 6 horizontal stripes, see *Figure: 5.5*, and for every stripe a 16 bin histogram for each channel is calculated. The 6 stripe histograms are each L_1 normalized and concatenated, in order to create the final ELF descriptor. The final ELF descriptor has a dimension of $6 \times 16 \times 30 = 2880$.

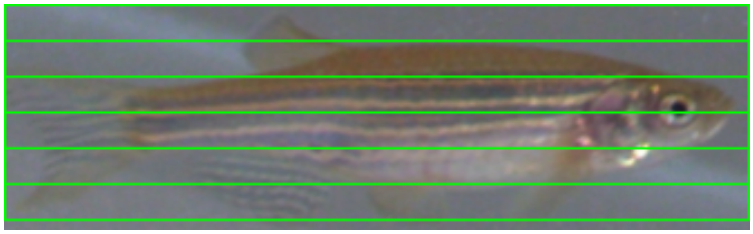


Figure 5.5: Illustration of how ELF splits the image into patches, here shown with 6 patches. Notice how parts of the bottom of the image is not included, due to the image height not being integer divisible with the number of stripes.

5.2.2 Local Maximal Occurrence

Liao et al. (2015) propose a feature representation method called the LOMO feature. This representation analyzes local features, in a patch-based manner. First, the image is preprocessed with the *multi-scale Retinex transformation* proposed by Jobson et al. (1997), which aims to make the lighting and colors of images from different camera views more consistent. Since the zebrafish data was recorded with a single camera and as to reduce processing time for each image, we omit this part in the implementation.

Prior to extracting features, the image is divided up into a set of patches. We use the same patch and stride as Liao et al. (2015): 10×10 patches, with a stride of 5 pixels. An illustration of how the patches are applied is shown in *Figure: 5.6*.

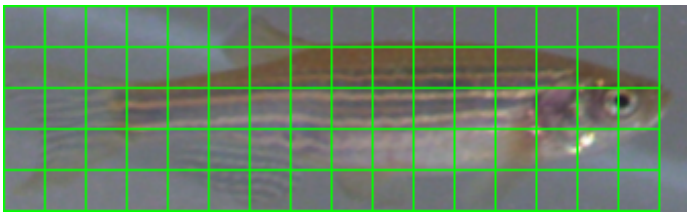


Figure 5.6: Illustration of how LOMO splits the image into patches, here shown with 20×20 patches and a stride of 20. Notice that the patches do not fully cover the image.

For every patch, a joint-HSV histogram and the Scale Invariant Local Ternary Pattern (SILTP) (Liao et al., 2010) are calculated. To help make the descriptor scale invariant, a three-scaled pyramid representation is created, where the input image is down sampled with a 2×2 mean filter, per step.

The joint-HSV histogram represents each pixel as a linear combination of the HSV values. Each channel in the HSV converted image is quantized into 8 bins, forcing them into the integer range $[0; 7]$. The HSV values for each pixel p are then represented as shown in *Equation: (5.33)*. The joint-HSV histogram is then determined for the investigated patch.

$$p_{HSV} = V \times 8^2 + S \times 8 + H \tag{5.33}$$

The SILTP descriptor is an improved version of LBP descriptor, which is invariant to intensity scaling as well as noise. This is achieved encoding each pixel value as a ternary string, instead of a binary string. Furthermore, the comparison with the center pixel is conducted using two thresholds, see *Equation: (5.34)*. If the compared pixel is fulfill either of the thresholds it is encoded as 2 or 1, depending on whether it has a value higher or lower than the center pixel, and if neither thresholds are fulfilled it is encoded as 0. This process is illustrated in *Figure: 5.7*. SILTP is applied on the value channel. Within each patch a 3^P-sized histogram is calculated. We use a radius R of 3 and 5, both with 4 points, as proposed by Liao et al. (2015).

$$\begin{aligned} I_k &> (1 + \tau) I_c \\ I_k &< (1 - \tau) I_c \end{aligned} \tag{5.34}$$

Where I_k is the investigated pixel, I_c is the center pixel and τ is a scale factor, which is set to 0.3, when the input pixels are in the range $[0; 1]$.

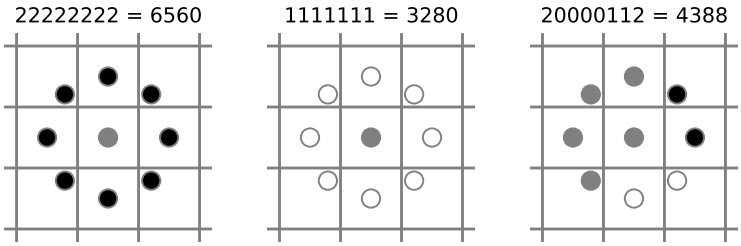


Figure 5.7: Illustration of the SILTP descriptor, with a radius R of 1 and 8 points P, counting clockwise from the point to the right of the center point (gray). Black circles indicate values above the center point threshold, white indicates values below the threshold, and gray values which are in between. The ternary string for each example and its decimal equivalent are shown.

In order to make the descriptor pose invariant, the "local maximal occurrence" is found per row. For every row of patches, the histograms of the patches are compared and only the maximum value for every bin is saved in the final histogram. Therefore, each row of patches in *Figure: 5.6* would be represented by a single joint-HSV histogram and a single SILTP histogram. The row histograms are then concatenated, which is done separately for the joint-HSV and SILTP histograms.

The SILTP and joint-HSV descriptors extracted at each scale are concatenated separately, and large bin values are suppressed by taking the log of the descriptor. The SILTP and joint-HSV descriptors are then L_2 normalized, and concatenated to create the final feature descriptor.

The dimensionality of the final LOMO feature, assuming a stride of 5, a patch size of 10×10 and the previously mentioned parameters for the joint-HSV and SILTP descriptors, can be calculated as shown in *Equation: (5.35)*. The dimensionality of the LOMO feature descriptor is dependent on the input image, unlike the ELF feature descriptor.

$$dim_{LOMO} = (8^3 + 2 \times 3^4) \times \sum_{j=0}^{2} \left(\left\lfloor \frac{I_{H,j}}{\text{str}} \right\rfloor - 1 \right) \tag{5.35}$$

Where $I_{H,j}$ is the height of the input image at the jth step of the down sampling pyramid (0th step being the original image), and str indicates the stride used.

6 | Experiments

In this chapter we validate the performance of the proposed methods by a set of widely used evaluation metrics for object tracking and object re-identification. Since the problem and its resolution can be seen as a novel one in the field, there are no other methods we can quantitatively and accurately compare our approach to. Therefore, we follow the best practices known for conducting experiments on the data and reporting the results. The evaluation is divided into two parts: a generalization test and a sequence association test.

6.1 Evaluation Metrics

This section goes over the evaluation metrics chosen to measure the system's performance. We took inspirations from person re-identification, and selected metrics that would fit our case: multiple gallery and potentially multiple probe images per identity. In contrast, many person re-id datasets consist of only two samples per identity (i.e. two images from two camera views of a single person), which are randomly assigned into the probe and gallery datasets. Below we describe the two chosen metrics.

6.1.1 Mean Average Precision

Mean Average Precision (mAP) is a standard metric used in a variety of use cases: information retrieval, object detection and object re-identification, based on the Average Precision (AP) metric. The average precision works as follows: taking a list of ordered recommendations l (in our case, gallery images of zebrafish, where there can be several of the same class) for a query, q, (in our case, a probe image of a zebrafish), the precision at every rank l_i is calculated, which is then, weighted by the change in recall from rank $i-1$ to i. Precision shows all the relevant samples from all selected positive candidates; and recall describes how well the method can choose all the relevant samples at k predictions. AP can also be calculated using *Equation: (6.1)*. Since the list is ordered, then the top predictions ideally should all be true positives. If there are several queries being investigated at the same time, then mAP can be calculated, by averaging the AP for each query, see *Equation: (6.2)*. The mAP can be computed at each rank i, but typically the mAP value when the entire gallery has been investigated is reported.

$$\mathrm{AP(q)} = \frac{1}{|R|}\sum_{i=1}^{n} Prec(i) \cdot relevance(i) \tag{6.1}$$

Where $Prec(i)$ is the precision at the ith rank, $relevance(i)$ indicates whether the sample at rank i belongs to the investigated query (whether is is *relevant*) and R is the set of relevant gallery images for probe q. $Prec(i)$ is calculated as the amount of relevant gallery images so far divided by the rank i.

$$\mathrm{mAP(Q)} = \frac{\sum_{q=1}^{Q} \mathrm{AP(q)}}{|Q|} \tag{6.2}$$

Where Q is the set of queries being investigated.

6.1.2 ID Switches

Another suitable metric to report the results is ID switch or mismatch rate. Bernardin and Stiefelhagen (2008) introduced two novel evaluation metrics: *multiple-object tracking accuracy* (MOTA) and *multiple object tracking precision* (MOTP). We have looked into those performance metrics, and found a mismatch part relevant for the project. MOTP aims to assess the object tracker ability to estimate the exact position of the object, which is not applicable in our case, with manually annotated data. MOTA combines three

types of errors: misses, false positives and mismatches over all objects in all frames. Again, since the data provided is already annotated, miss ratio and false positive ratio can not be used. However, the third component can be useful at describing how often the algorithm switched the identities of zebrafish. Refer to Figure 6.1, to see how the ID switches are counted for three zebrafish. The final ID switch count is reported as a percentage of all possible ID switches in the investigated sequences.

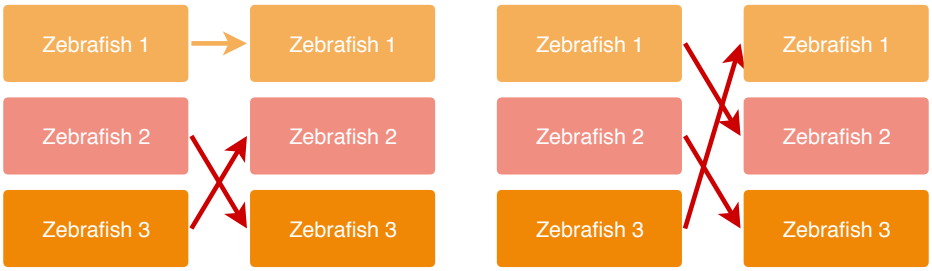


Figure 6.1: Illustration of how the mismatches are counted. On the left is the case of 1 mismatch; on the right is the case of 3 mismatches. For the dataset with only two zebrafish, there is at maximum 1 mismatch per assignment.

6.2 Preliminary Tests

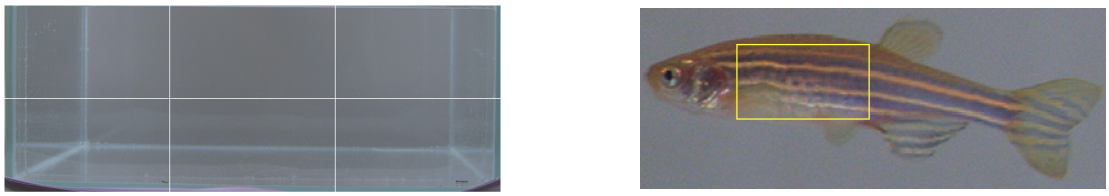
Since the data was collected manually, we needed to make sure that it follows some characteristics of the well-gathered data. Because of that, we performed a number of preliminary tests, looking into potential biases and proofs that the data can be considered plausible, to test the problem hypothesis and get valid results. The setup was laboratory-driven (refer to *Section 4.2*), yet still built from the available resources (e.g. time, lighting, filming area). Those factors potentially affect the data reliability, which can propagate or influence the final results, such as the accuracy of re-identification algorithm. From the tests described below, we can observe the data validity and can further decide upon the variable values for final performance tests of the system.

The following *PCA* and *background* tests are done on two separate datasets: 3-FISH-TRN and 2-FISH-TRN, respectively. Ideally, these tests would be done on the same dataset, as to reduce bias, since the output parameters of one test are the input parameters to the another test. However, in 3-FISH-TRN there are only 11 frames found for one of the fish within the ROI for the background test, which is not sufficient amount for training.

Moreover, we wanted to perform preliminary tests on the training datasets, and not include testing dataset in the parameter search in any way. This is done by splitting the training datasets into two equal half, and use one for training and the other for testing, and thereby avoid using the dedicated testing dataset. All tests are performed over 10 random splits, meaning that the dataset is shuffled 10 times. For each split a single probe is randomly selected per fish.

The training and testing datasets are also balanced, so that there is an equal amount of samples per fish. The metric learning methods from *Section 5.1* and feature descriptors from *Section 5.2* are all tested. The KISSME based methods are tested with and without the PSD constraint, while the DNS algorithm is tested with the linear and the RBF kernel functions. The L_1, L_2 and cosine distances are included as a baseline, tested on both the original and PCA-reduced features, to compare the performance of the learned metrics with.

For each test, the mAP curve over the different ranks are plotted, and the final mAP value, with its standard deviation over the 10 random splits, when all gallery images are considered is noted for each method, together with the percentage point standard deviation over the 10 splits, when suitable.



(a) (b)

Figure 6.2: (a) ROI in water tank, with divided regions for the color test. (b) An example of annotations for the color test.

6.2.1 Color Test

The color test aims to show whether there is a bias based on a position of the fish in a specific part of the water tank. If the color significantly changes from one corner of the tank to another, that may heavily influence the color part of the descriptors and produce biased results. Both descriptors and metric learning methods may pick up those differences and identify the fish according to its placement, rather than individual visual characteristics. In the dataset, fish tend to swim at different water tank areas. We chose the 2-FISH-TST dataset and extract areas of a fish body (i.e. from the head to the upper fin, see *Figure: 6.8c*) in 6 regions of the tank. *Figure: 6.8a* illustrates how the frame is divided into the regions, from where we extracted the fish. The dataset is chosen due to presence of fish in all six areas, for both fish. That way we can observe whether the fish color is changing relative to its positioning.

The extracted images were converted to HSV color space, and plotted against each combination of color space components, as seen in *Figure: 6.3*. The two fish have large overlaps in values, which indicate, that the colors are consistent between two identities. Moreover, the values are not randomly spread out, but forming two clusters, which can refer to the stripes on the fish body. The first plot in *Figure: 6.3* shows that two identities are close to being linearly separated, when mean values are calculated for each image.

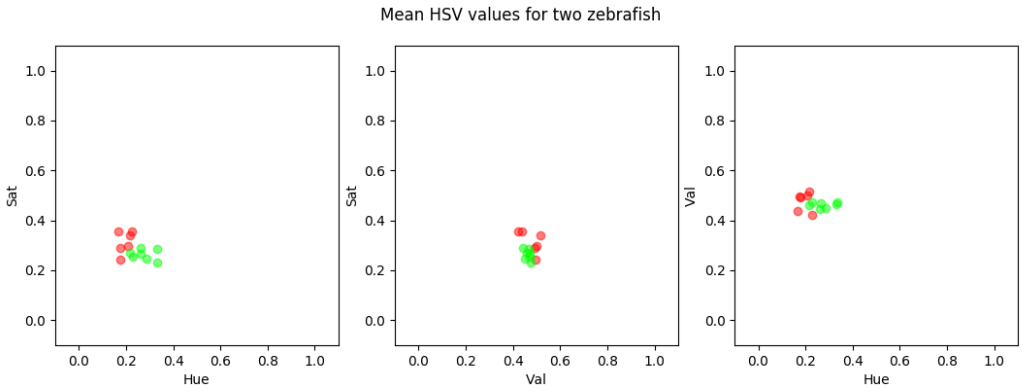


Figure 6.3: Mean values for each image in HSV color space. Red and green colors indicate two fish. The values are not spread out, indicating that the position of the fish in the tank does not add significant bias to the color. Different fish are close, yet form separate clusters.

6.2.2 PCA Test

As some of the tested methods require the supplied feature descriptors to be dimensionality reduced through PCA, it is necessary to investigate how aggressive this reduction can be, without degrading the results. This is done by investigating how the mAP and computation time is affected, when testing the LOMO and ELF feature descriptors on the 3-FISH-TRN dataset, and varying the amount of "explained variance" retained. The amount of explained variance retained is calculated by choosing just enough principal directions for

Equation: (6.3) to be fulfilled, from 1% to 100% in steps of 1 percentage point.

$$\arg\min_{k} \frac{\sum_1^k \lambda_k}{\sum_1^d \lambda_d} \geq i \tag{6.3}$$

Where i is a value between 0 and 1 indicating how much of the variance should be used, λ_j is the variance for the jth principal direction, sorted in descending order, d is the total number of principal directions and k is the number of principal directions for the inequality to be fulfilled.

As there are fewer samples than the dimensionality of the feature descriptor, that is $n < d$, there can at max be $n - 1$ non-zero principal directions. Therefore, PCA still provides a dimensionality reduction even when 100% of the variance is included. The effect on the mAP for ELF and LOMO can be seen in *Figure: 6.4* - *Figure: 6.5*, and the effect on the computation time can be seen in *Figure: 6.6* - *Figure: 6.7*. The computation plots are split into three steps: the time it takes to train the method, compare probe with gallery features, and calculate the mAP value.

LOMO can sustain a quite aggressive dimensionality reduction while retaining both computational speed and mAP, while ELF does not perform well until a large amount of the variance is included. It is also observed that there is a massive drop in mAP for the metric learning methods, when 100% of the variance is included. This is due to the inclusion of principal directions where the variance is near 0. The training and comparison time increase drastically when more than 95% variance and 60% variance is included for ELF and LOMO, respectively. Therefore, for all future tests, PCA is applied so that 60% of the variance is included for LOMO features, and 95% for ELF features.

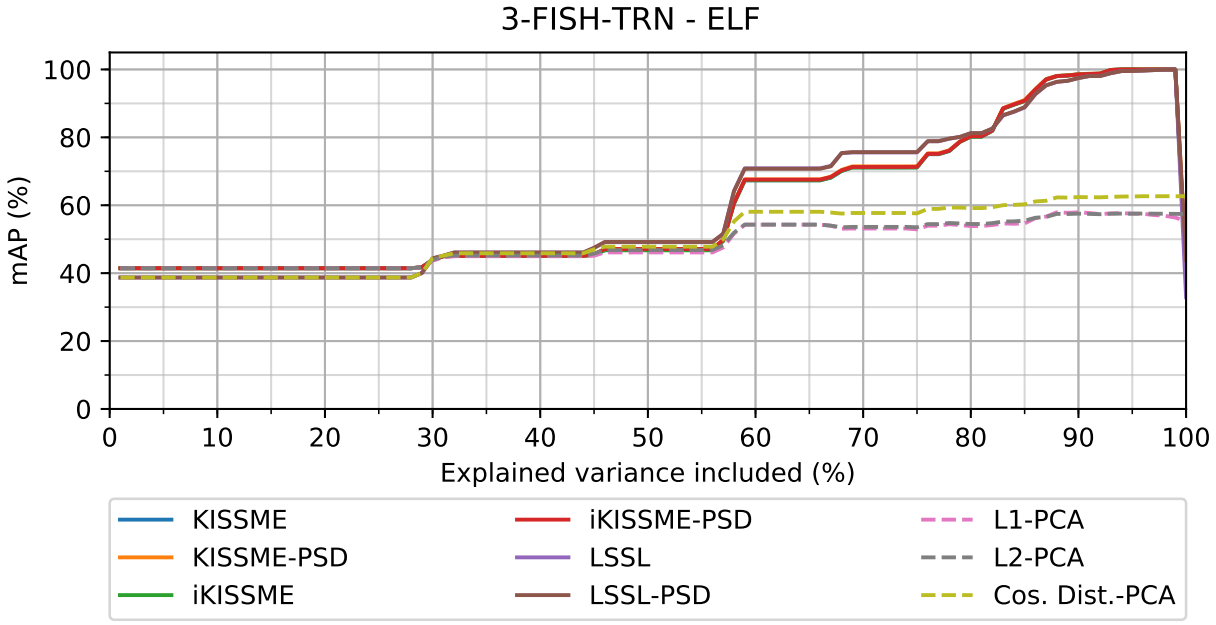


Figure 6.4: The mAP compared to the amount of explained variance included for PCA, when using ELF and the 3-FISH-TRN dataset

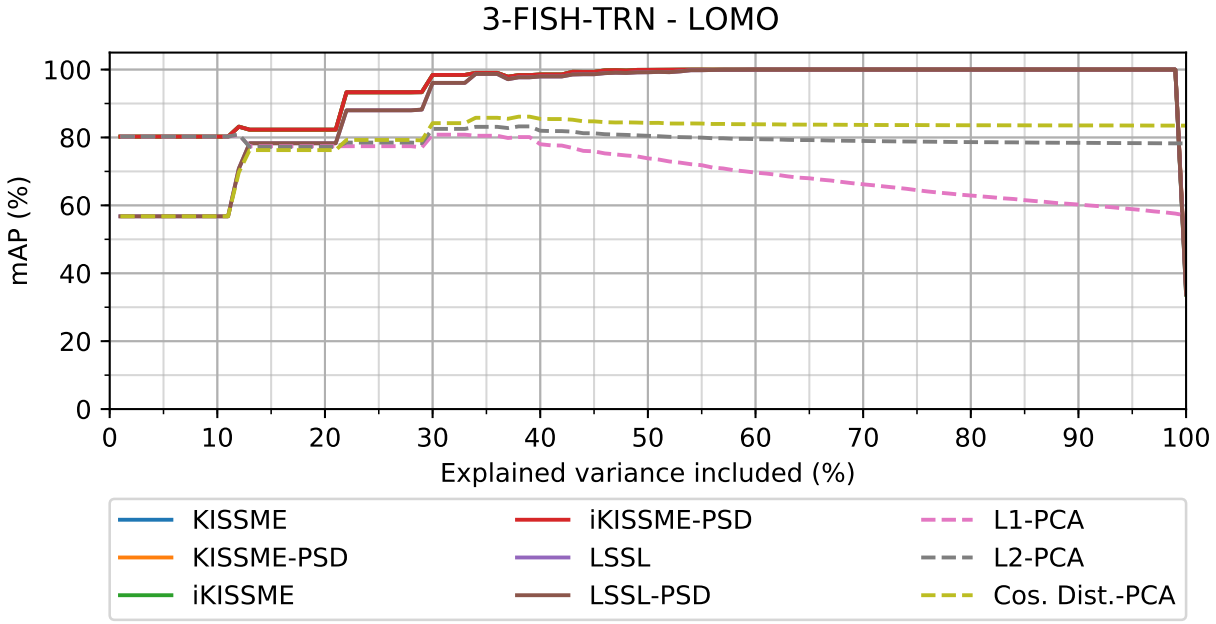


Figure 6.5: The mAP compared to the amount of explained variance included for PCA, when using LOMO and the 3-FISH-TRN dataset

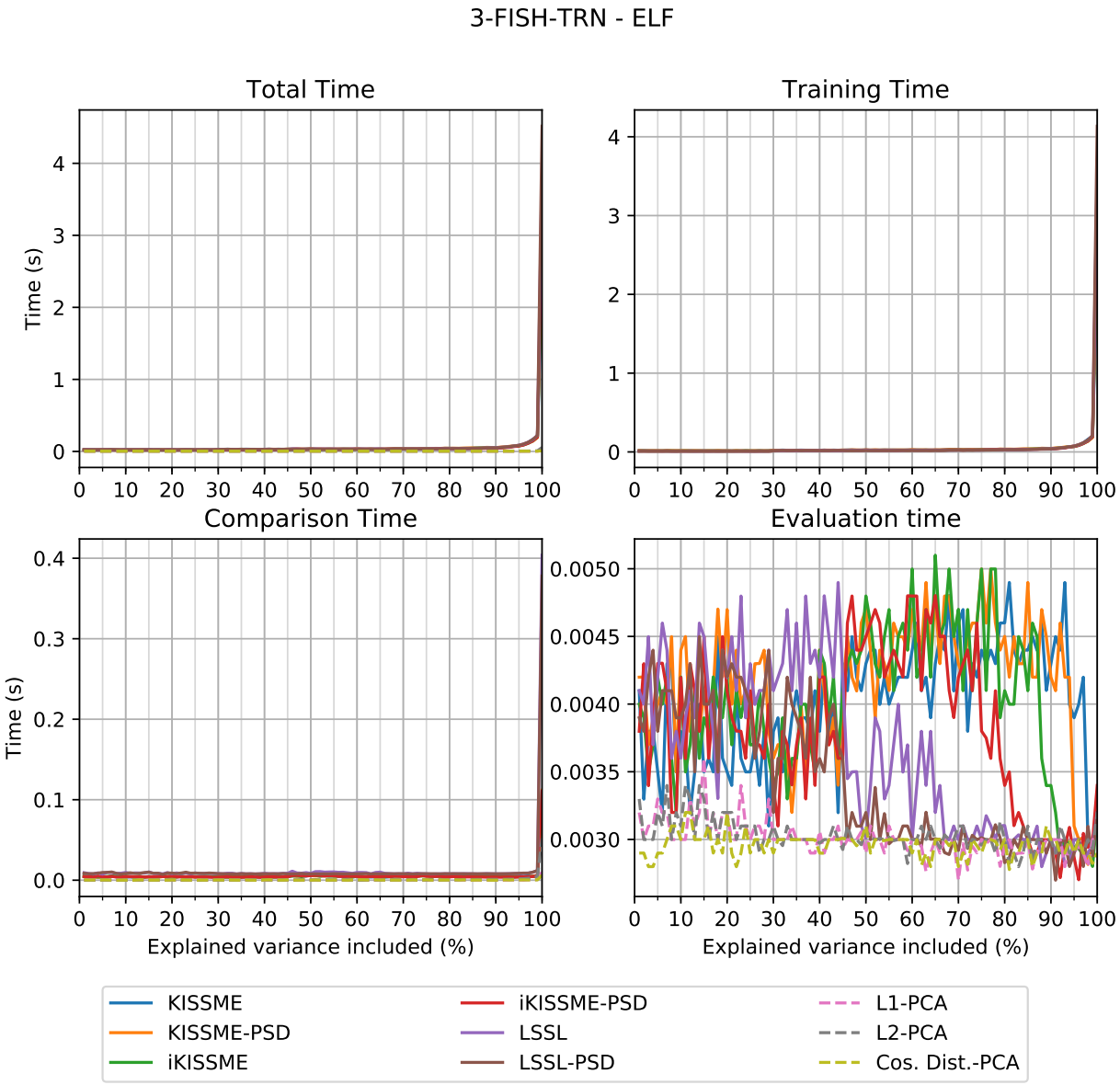


Figure 6.6: The time spent on different steps in the algorithms compared to the amount of explained variance included for PCA, when using ELF and the 3-FISH-TRN dataset

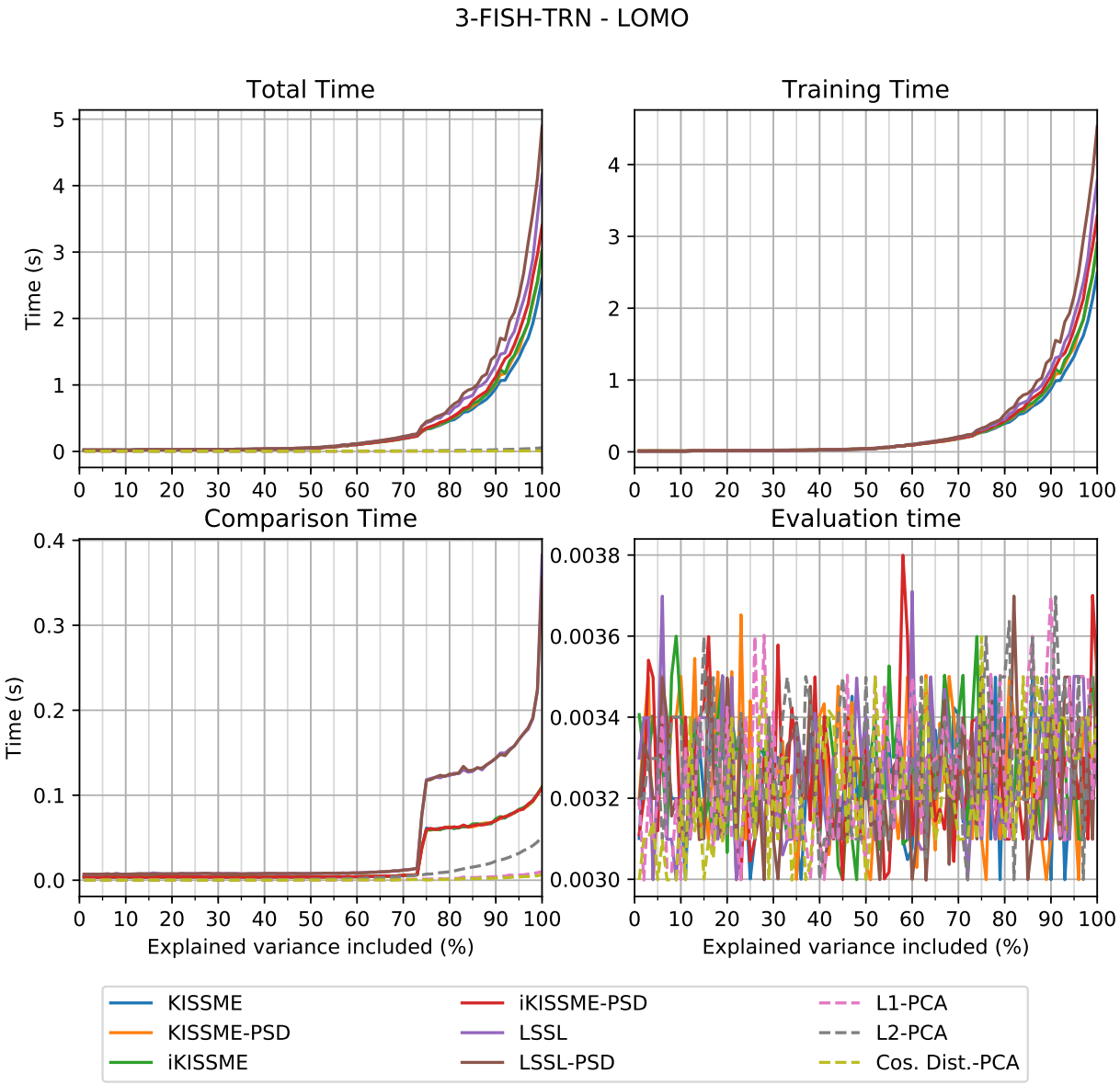


Figure 6.7: The time spent on different steps in the algorithms compared to the amount of explained variance included for PCA, when using LOMO and the 3-FISH-TRN dataset

6.2.3 Background Test

Even though the data collection was obtained within a laboratory setup (e.g. same framing for each dataset, controlled lighting), the background could still impact the results. The bounding boxes tend to include small parts of the background, such as edges of the tank (see *Figure: 6.9*), and as a preprocessing step we did not subtract the background for the images. The reason lies in the appearance of the fish: the zebrafish tail is semi-transparent, yet still holds a lot of the stripe patterns and color information, that can help the descriptors to distinguish fish. And it is difficult to decide on the threshold value, for subtraction, since the fins and the tail are movable parts of the fish, and can vary during fish movements. *Figure: 6.8* shows the mean values for three fish, where the shape and transparency difference is quite visible. Therefore, subtracting the background could result in removing the fish parts, which are valuable for identification

task.

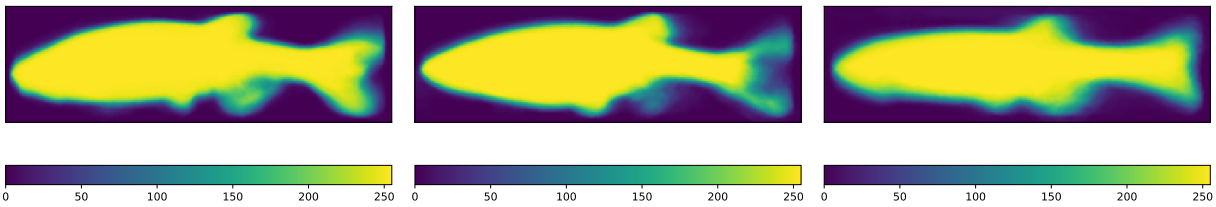


Figure 6.8: The average shape of fish in 3-FISH-TST dataset. The tail and fins are semi-transparent, which can result in loosing important details of the fish. a, b, and c refer to fish 1, 2, and 3.

Instead of removing the background, we restricted the area for processing, see *Figure: 6.9*. The annotations within this ROI for the 2-FISH-TRN dataset were then extracted, and tested on. Similar, we randomly selected the same amount of training and testing data from the original dataset, assuming some of the selected annotations will contain the lines from the aquarium. The test is conducted with both the LOMO and ELF features, using the PCA ratios from *Section 6.2.2*. The results of the tests are shown in *Figure: 6.10* - *Figure: 6.13*.

The results show that the performance for the baseline metrics is lower when not using the annotations from within the selected ROI, while the learned metrics, except XQDA, are not affected. Therefore, the lines from the aquarium do not assist in the re-identification, but actually decrease the accuracy at times. However, in order to not limit ourselves to only a subset of the frame, we will utilize all of the data available.

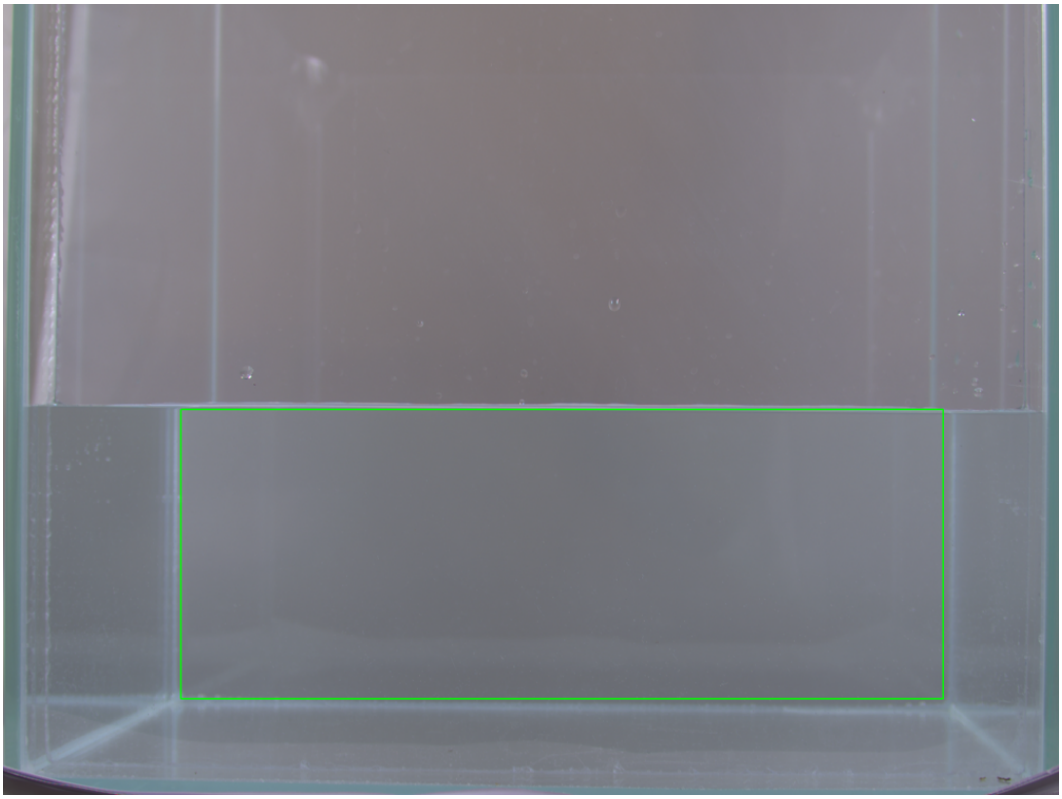


Figure 6.9: The selected ROI that does not include edges of the tank, resulting in a uniform background.

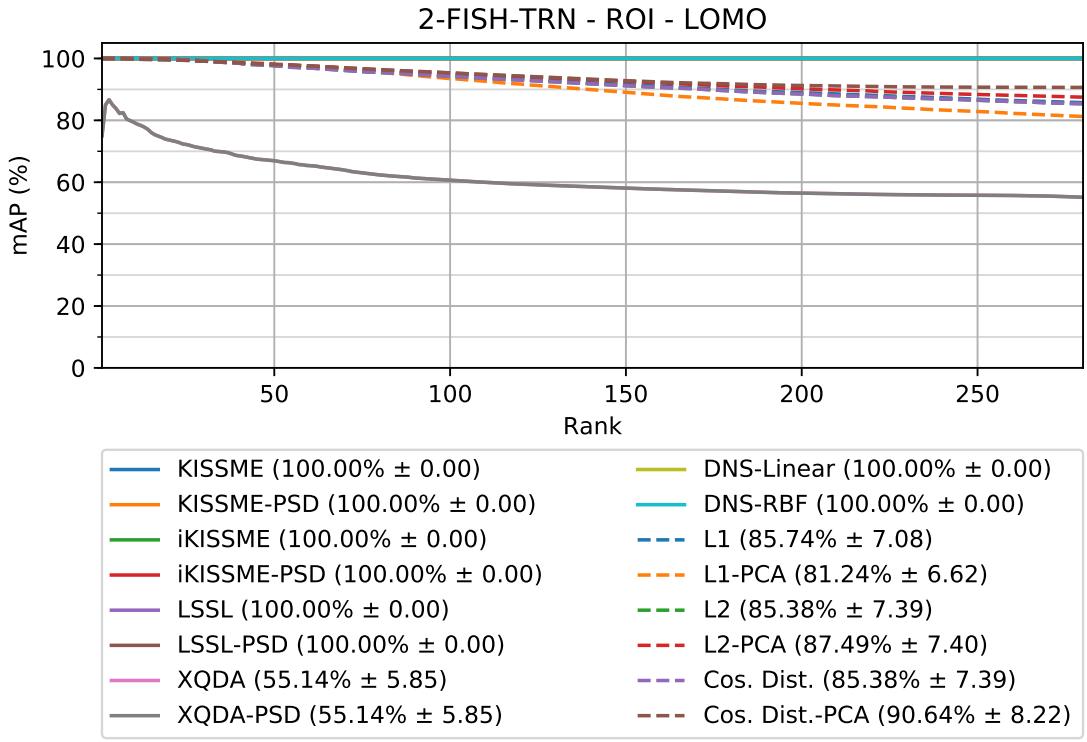


Figure 6.10: mAP using the LOMO descriptor and fish from just within the ROI in *Figure: 6.9*

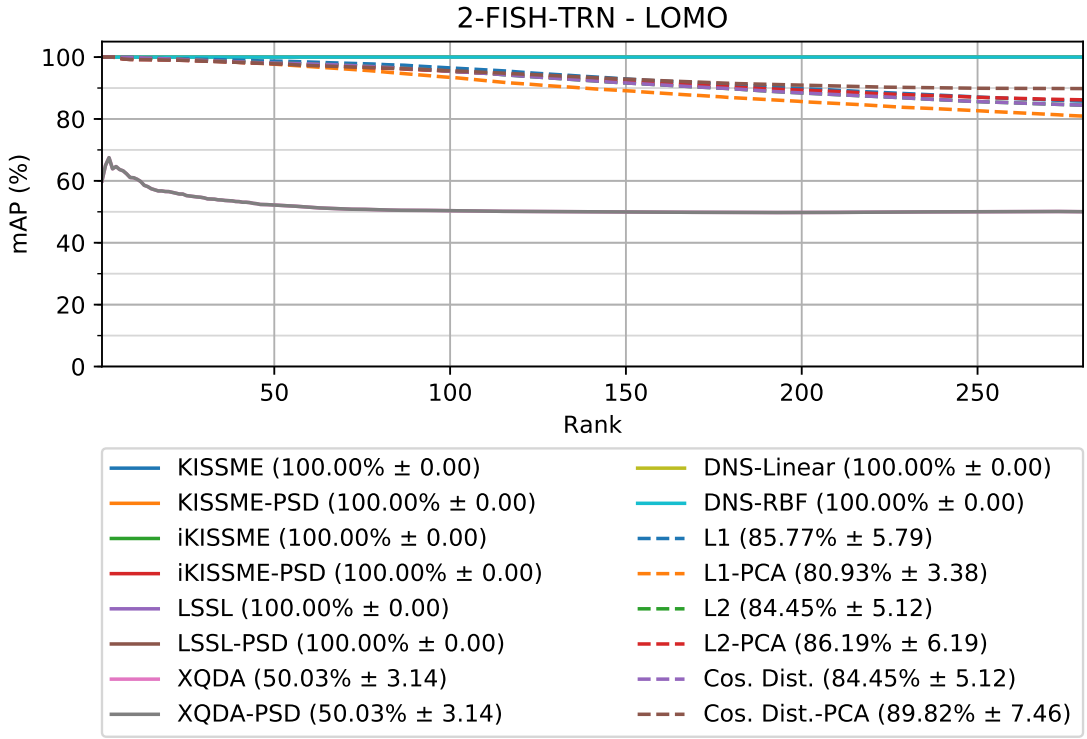


Figure 6.11: mAP using the LOMO descriptor and fish from the entire frame

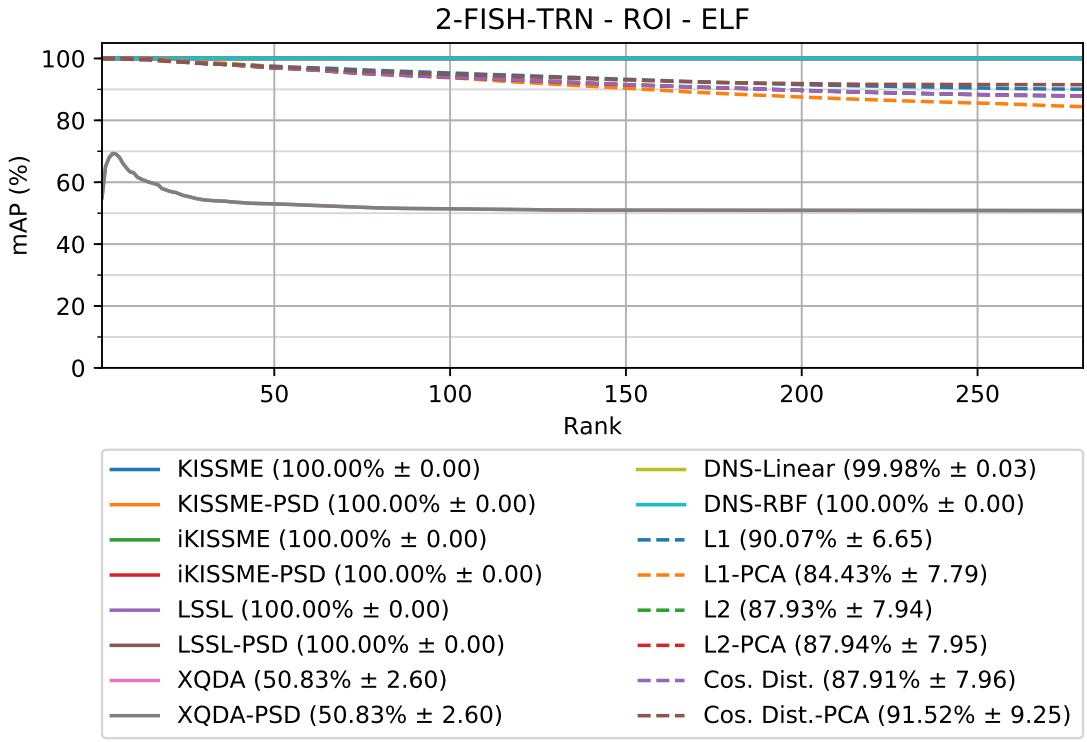


Figure 6.12: mAP using the ELF descriptor and fish from just within the ROI in *Figure: 6.9*

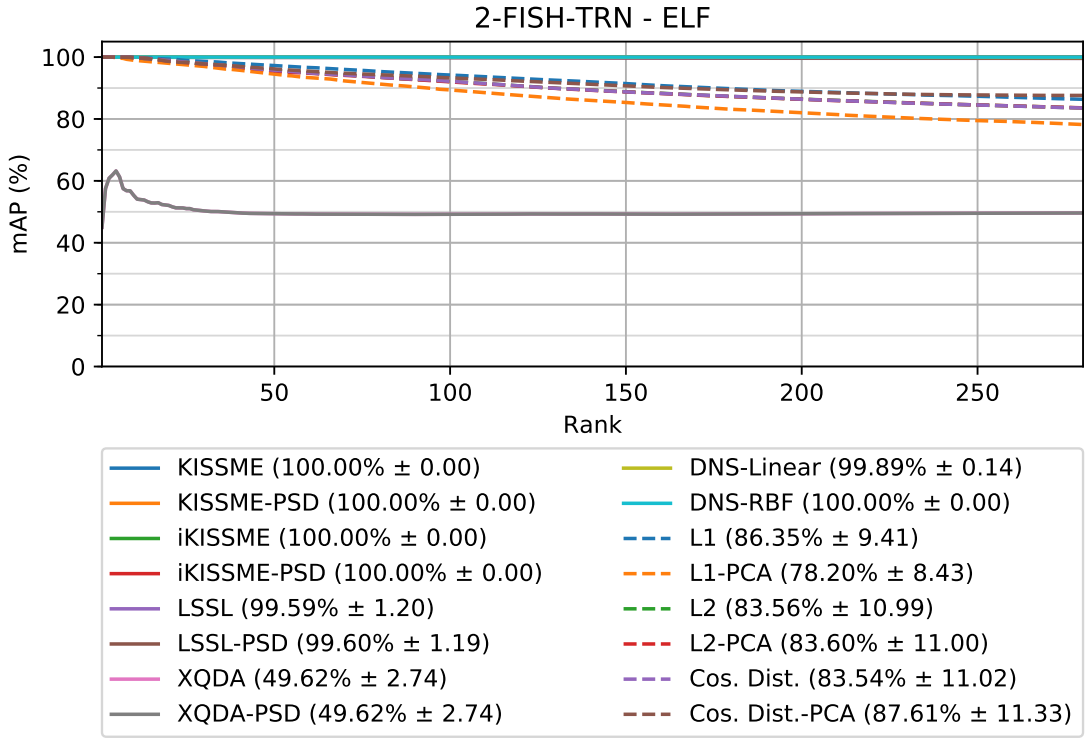


Figure 6.13: mAP using the ELF descriptor and fish from the entire frame

6.3 Generalization Test

The aim for this test it to see how well the knowledge from the methods generalizes across datasets, and thereby across different fish (which are only introduced in the testing part). A set of tests are conducted where the metric learning algorithms are trained on the training datasets, and tested on the testing dataset. For a baseline test, a training dataset is equally divided into two parts: training and testing subset. These tests would show how well the methods work, when testing on the same fish, and ignoring any temporal connection between samples. All tests are performed over 10 random splits and balanced, as in the preliminary tests.

Lastly, for all tests the feature descriptors are investigated both in their full extent, but also in their subparts: their color and texture components. This is to determine the contribution of each subpart of the feature descriptors. The subpart are denoted ELF/LOMO-COLOR and ELF/LOMO-TEXTURE.

6.3.1 2-FISH-TRN

The results from the baseline test using the 2-FISH-TRN dataset is shown in *Figure: 6.14* - *Figure: 6.19*, and the performance of the methods on the different descriptors are summarized in *Table: 6.1*. The results show that all learned metrics, except XQDA, perform nearly perfectly for all variations of LOMO and ELF, except the DNS-Linear algorithm (in the latter descriptor), which fails to learn the data structure sufficiently. It is also interesting, that the DNS-Linear approach gains a 3 percentage points increase using the ELF-COLOR descriptor, when compared to the full ELF descriptor, while it loses 18 percentage points when using ELF-TEXTURE descriptor. In general, the baseline metrics perform the best when the color components are used, followed by the full descriptor and lastly the texture components. For ELF-TEXTURE, the learned metrics in general perform 9 percentage points worse than the full ELF and ELF-COLOR descriptors.

Table 6.1: Overview of the performance for methods and feature descriptors in 2-FISH-TRN dataset. A checkmark shows in what descriptor configuration (column) the particular method (row) performs the best. The asterisk (*) shows which method performs the best in this particular feature descriptor configuration. This is done separately for ELF and LOMO.

2-FISH-TRN	LOMO			ELF		
	Full	Color	Texture	Full	Color	Texture
KISSME	✓*	✓*		✓*	✓*	
KISSME-PSD	✓*	✓*		✓*	✓*	
iKISSME	✓*	✓*		✓*	✓*	
iKISSME-PSD	✓*	✓*		✓*	✓*	
LSSL	✓*	✓*			✓	
LSSL-PSD	✓*	✓*			✓	
XQDA			✓			✓
XQDA-PSD			✓			✓
DNS-Linear	✓*	✓*	✓*		✓	
DNS-RBF	✓*	✓*	✓*	✓*	✓*	*
L1		✓			✓	
L1-PCA		✓			✓	
L2		✓			✓	
L2-PCA		✓			✓	
Cos. Dist.		✓			✓	
Cos. Dist. - PCA		✓		✓		

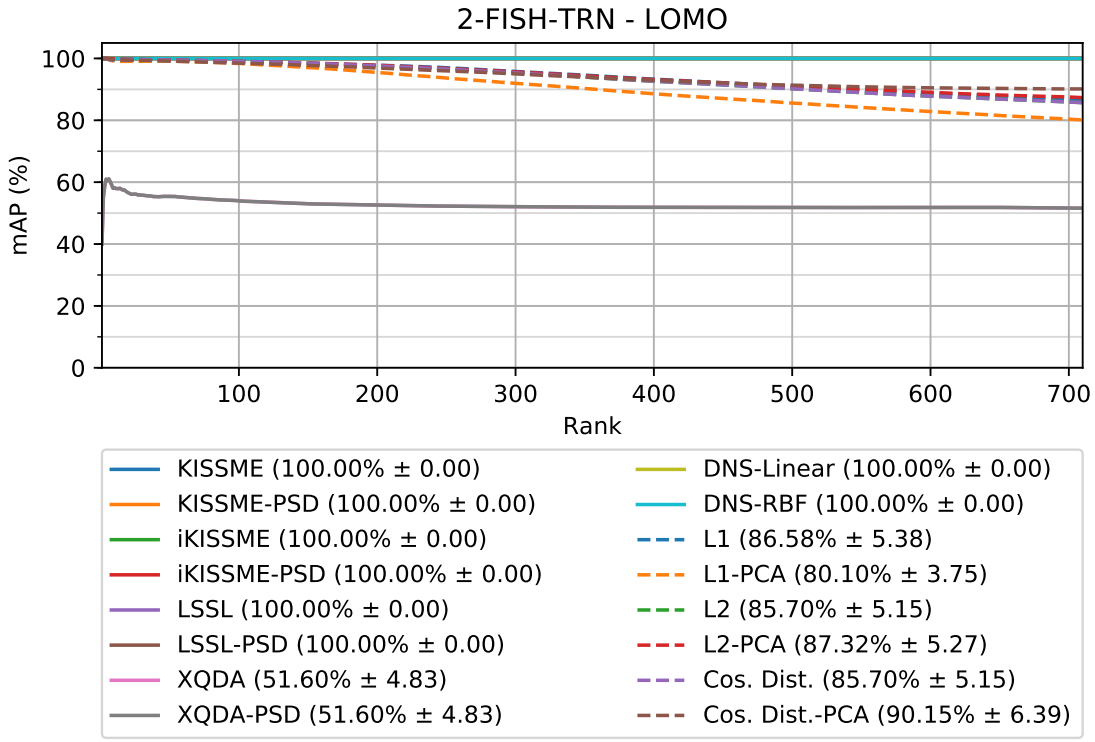


Figure 6.14: mAP results for 2-FISH-TRN with the LOMO feature descriptor.

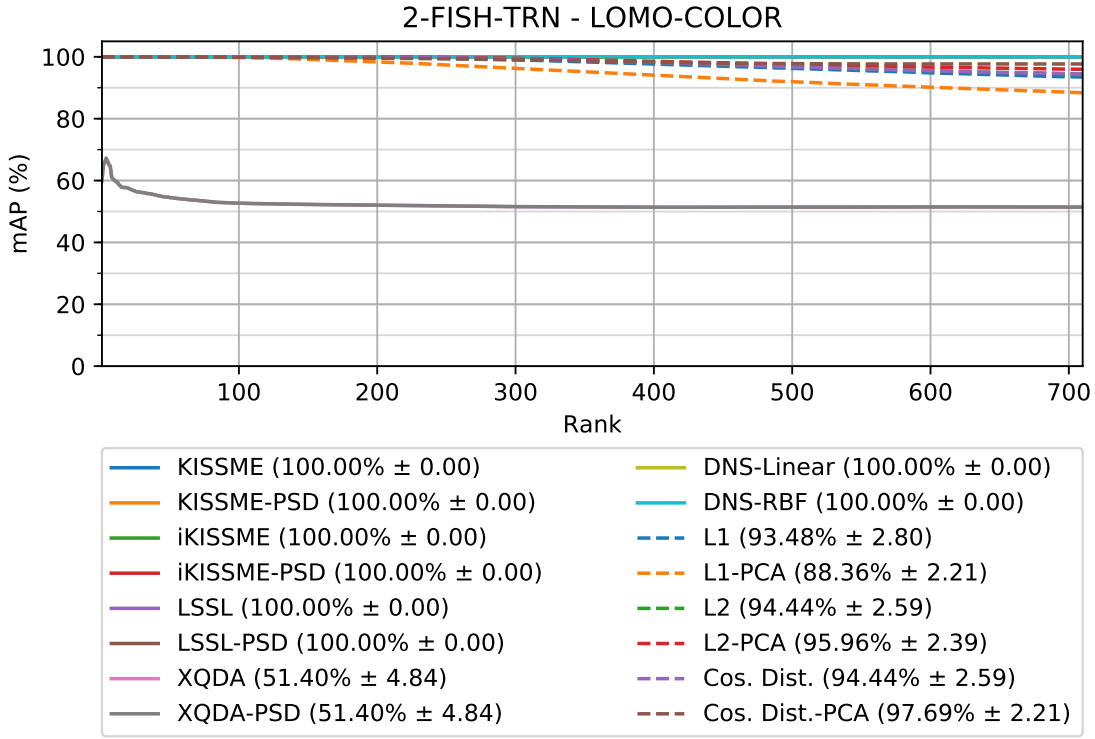


Figure 6.15: mAP results for 2-FISH-TRN with the LOMO-HSV feature descriptor.

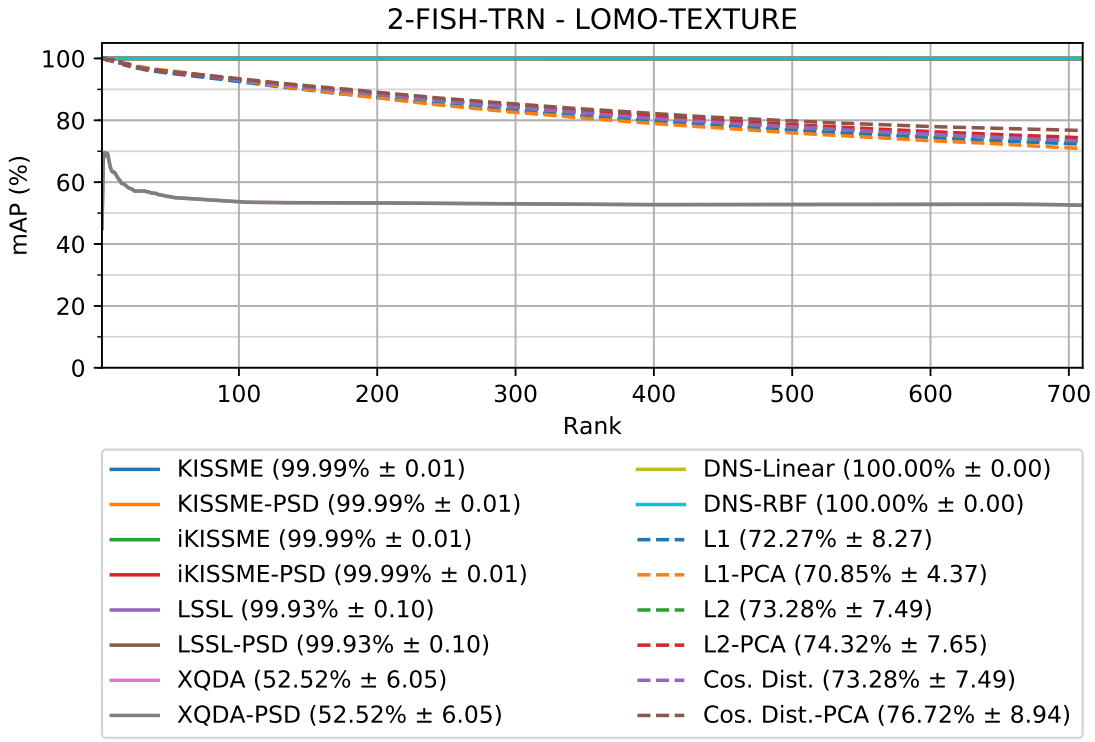


Figure 6.16: mAP results for 2-FISH-TRN with the LOMO-SILTP feature descriptor.

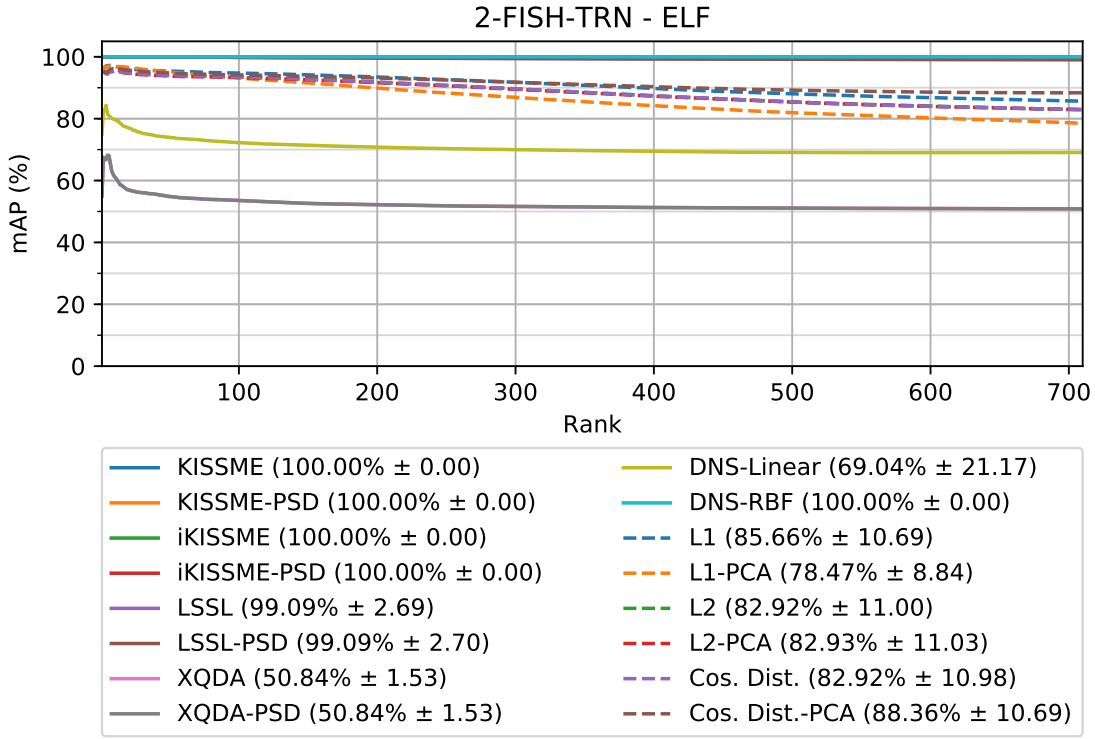


Figure 6.17: mAP results for 2-FISH-TRN with the ELF feature descriptor.

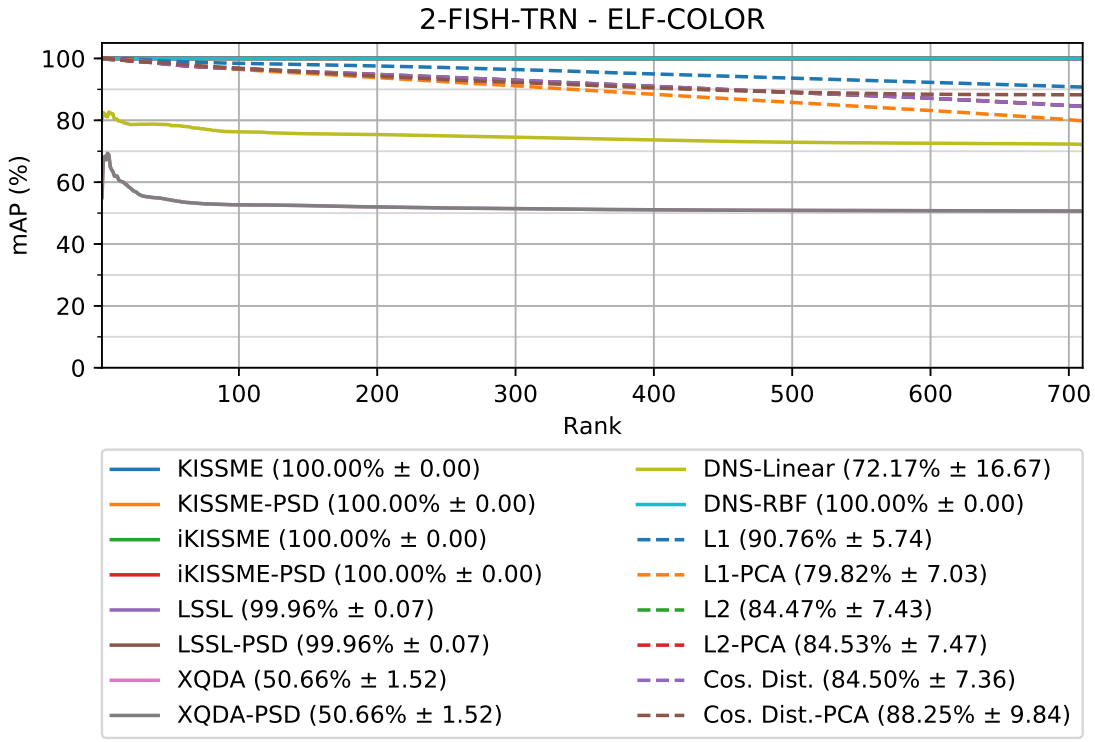


Figure 6.18: mAP results for 2-FISH-TRN with the ELF-COLOR feature descriptor.

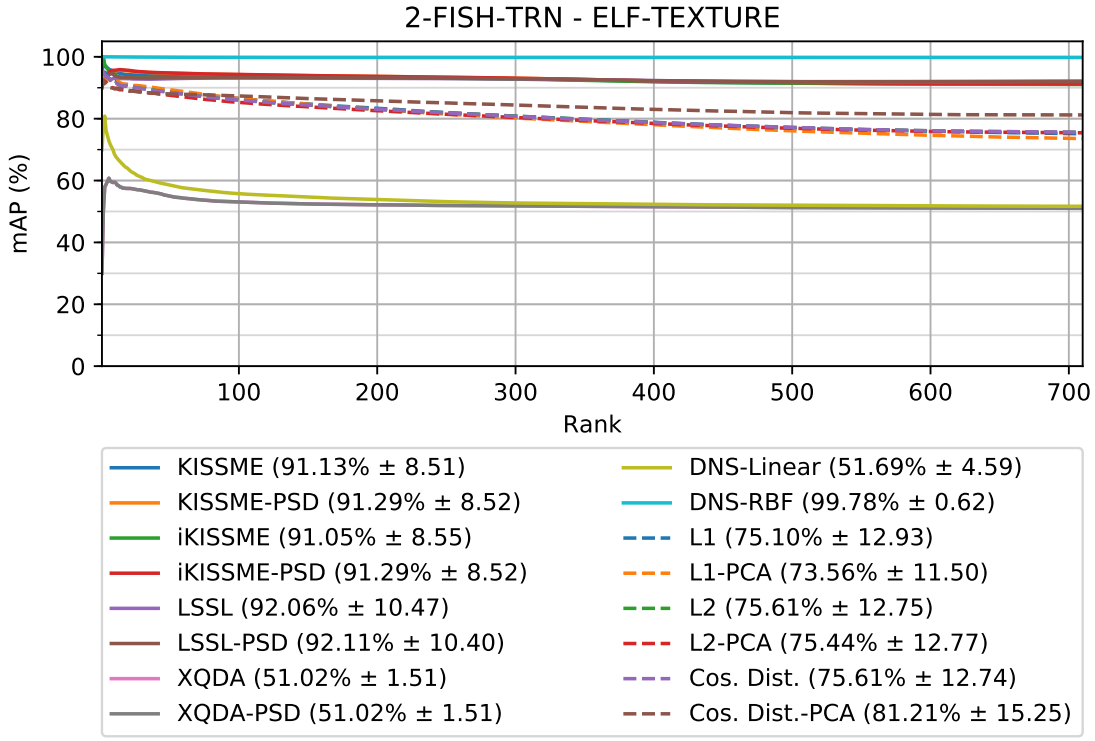


Figure 6.19: mAP results for 2-FISH-TRN with the ELF-TEXTURE feature descriptor.

6.3.2 2-FISH-TST

The results from the baseline test on the 2-FISH-TST dataset are shown in *Figure: 6.20* - *Figure: 6.25*. Overall, the best results achieved are 89.40% mAP for LOMO with the DNS-linear metric, and 100% mAP for ELF, with the DNS-RBF metric. On average, however, the results are not that high and can fall close to 50%, which is a random guess when testing with two fish. *Table: 6.2* helps to group the findings together. For LOMO descriptor, the color part outperforms both LOMO-texture and LOMO-full, in majority of the cases. DNS-Linear performs the best for the LOMO and LOMO-COLOR descriptors, while L_2 and cosine distance perform best for LOMO-TEXTURE. Overall, the baseline metrics outperform the learned metric by up to 31.27 percentage points, when subtracting the worst learned metric from the best baseline metric mAP. On average, the differences in performance between the learned and baseline metrics lie within 10-15 percentage points. Using ELF descriptor, the ELF and ELF-COLOR descriptors equally obtain the best performances for majority of the methods. Here, DNS-RBF produces the best results in two cases, while for ELF-TEXUTRE, the best performing algorithm is LSSL-PSD, obtaining 71.13% mAP.

Table 6.2: Overview of the performance for methods and feature descriptors in 2-FISH-TST dataset. A checkmark shows in what descriptor configuration (column) the particular method (row) performs the best. The asterisk (*) shows which method performs the best in this particular feature descriptor configuration. This is done separately for ELF and LOMO.

2-FISH-TST	LOMO			ELF		
	Full	Color	Texture	Full	Color	Texture
KISSME		✓		✓		
KISSME-PSD		✓		✓		
iKISSME		✓		✓		
iKISSME-PSD		✓		✓		
LSSL			✓			✓
LSSL-PSD			✓			✓*
XQDA		✓		✓		
XQDA-PSD		✓		✓		
DNS-Linear	*	✓*			✓	
DNS-RBF		✓		*	✓*	
L1		✓			✓	
L1-PCA		✓		✓		
L2		✓	*		✓	
L2-PCA		✓			✓	
Cos. Dist.		✓	*		✓	
Cos. Dist. - PCA		✓			✓	

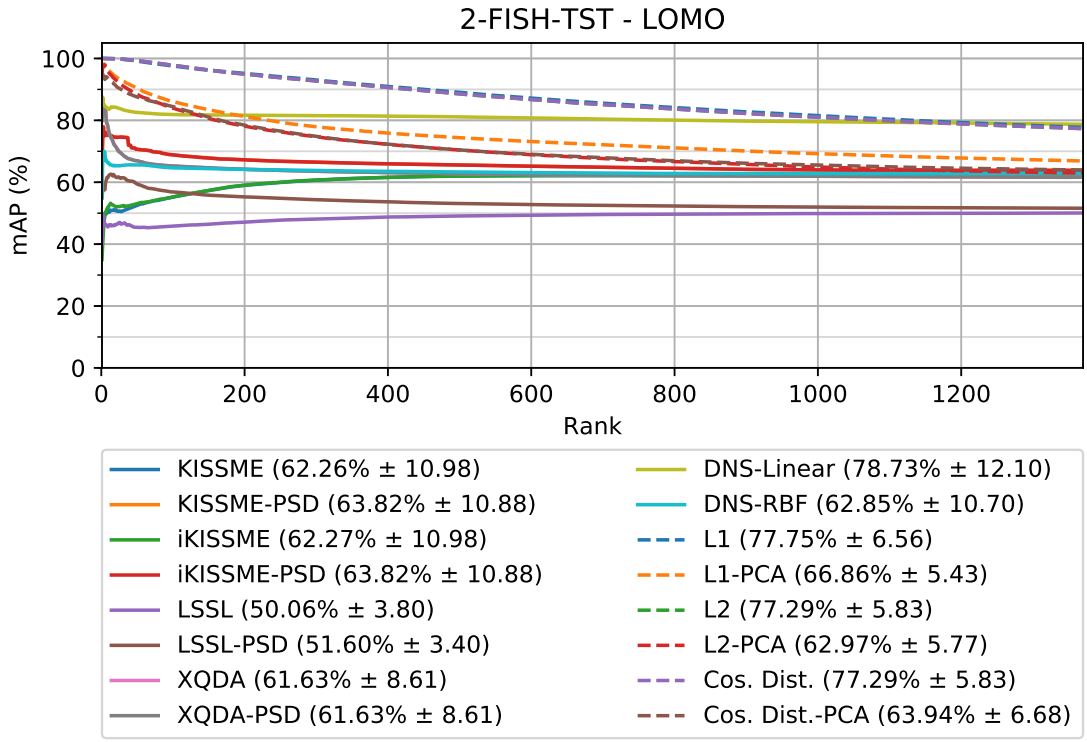


Figure 6.20: mAP results for 2-FISH-TST with the LOMO feature descriptor.

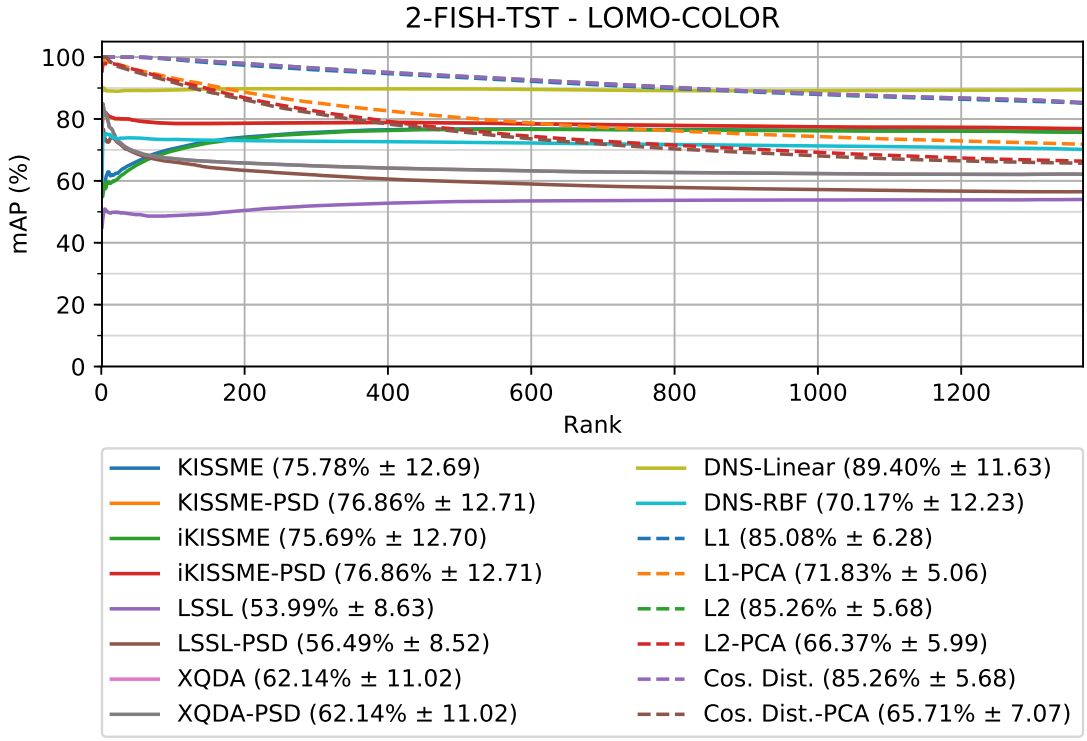


Figure 6.21: mAP results for 2-FISH-TST with the LOMO-HSV feature descriptor.

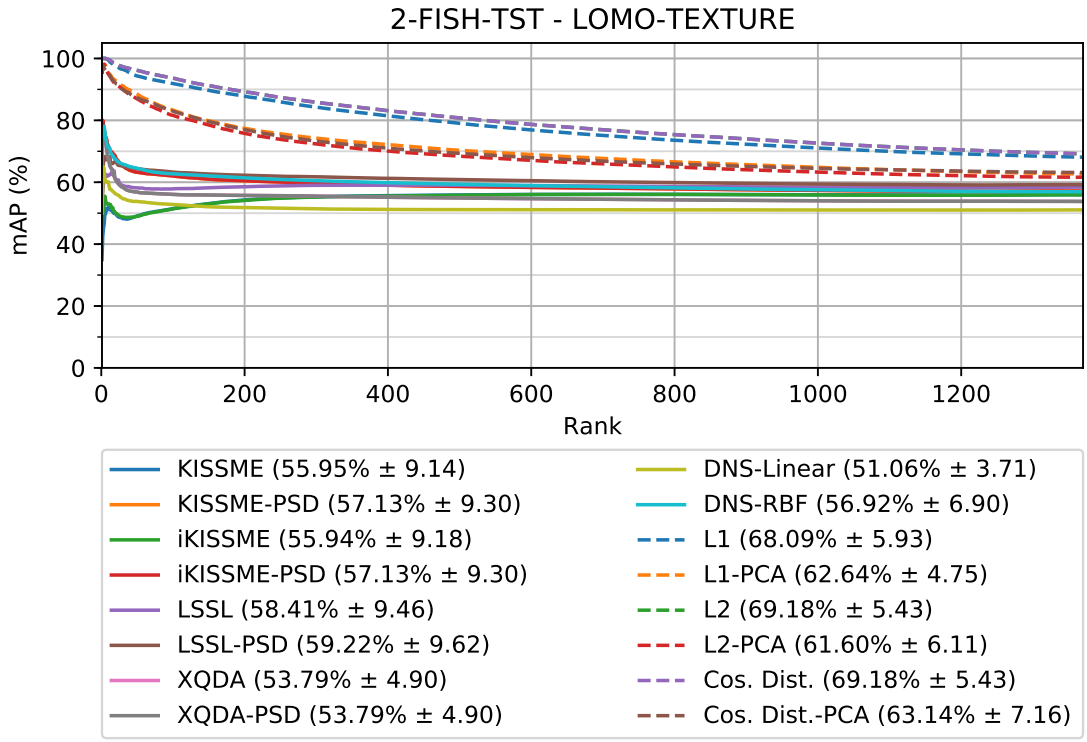


Figure 6.22: mAP results for 2-FISH-TST with the LOMO-SILTP feature descriptor.

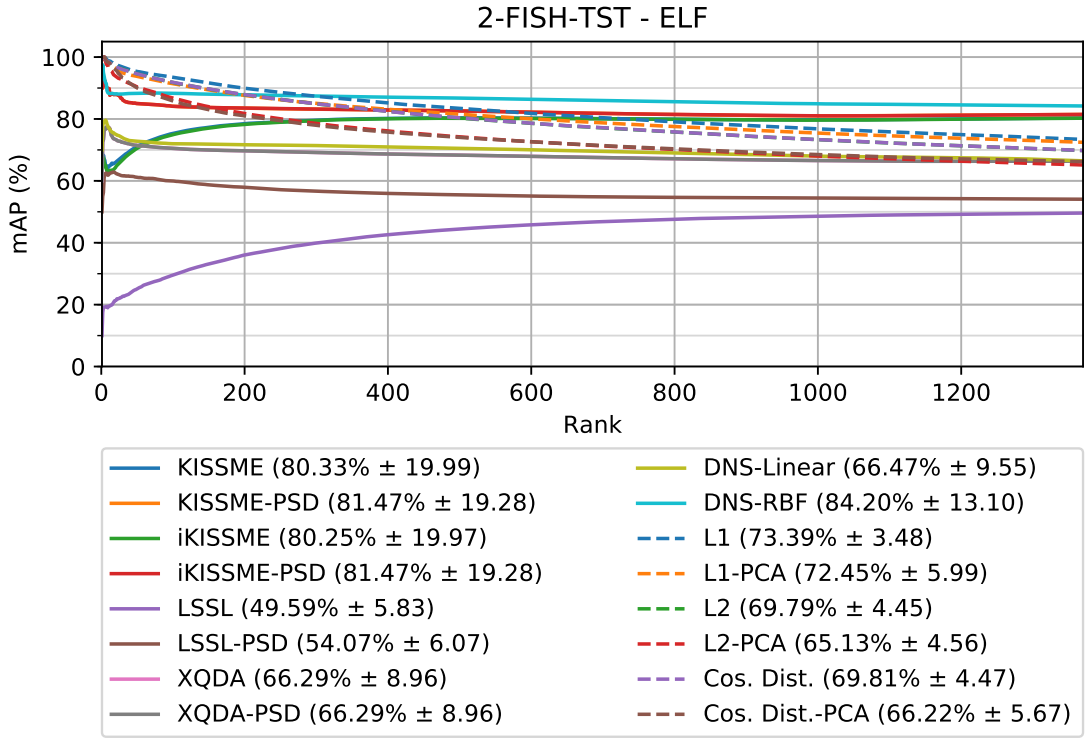


Figure 6.23: mAP results for 2-FISH-TST with the ELF feature descriptor.

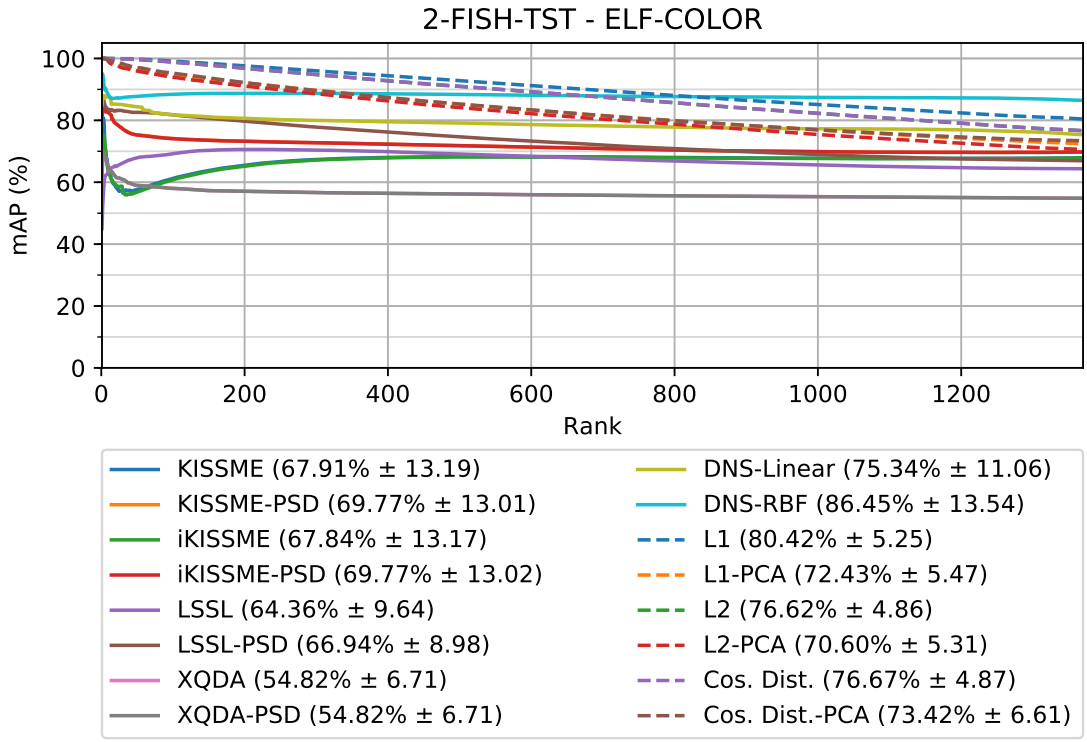


Figure 6.24: mAP results for 2-FISH-TST with the ELF-COLOR feature descriptor.

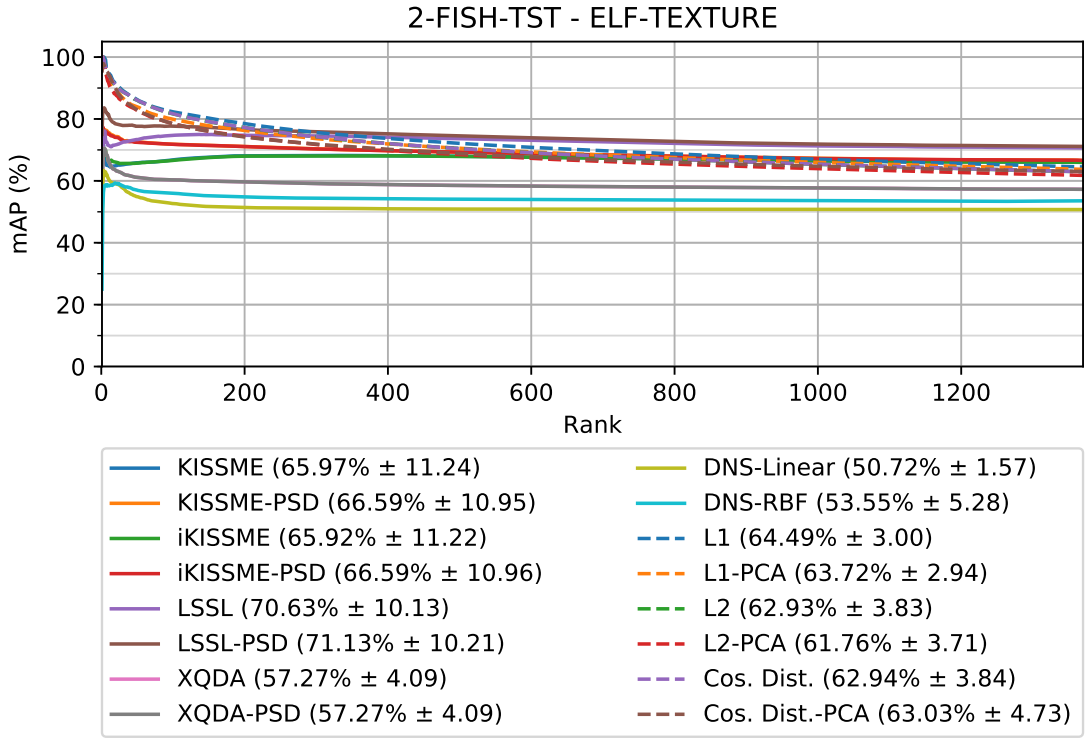


Figure 6.25: mAP results for 2-FISH-TST with the ELF-TEXTURE feature descriptor.

6.3.3 3-FISH-TRN

The results on the 3-FISH-TRN baseline test are shown in *Figure: 6.26* - *Figure: 6.31*, and summarized in *Table: 6.3*. Similar to the 2-FISH-TRN baseline test, all metrics, except XQDA, perform well on all LOMO descriptor variations, and specifically the LOMO-COLOR variation. All learned metrics obtain a mAP score near a 100% for all variations of the LOMO descriptor. For the ELF descriptor, the DNS-Linear algorithm again fails to sufficiently learn the data structure. The ELF-COLOR descriptor is still shown to be the best performing ELF variation, with ELF-TEXTURE scoring nearly 10 percentage point lower for all methods. The difference in performance for the baseline metrics in ELF-COLOR and LOMO-COLOR is 15 percentage point, in favor of LOMO descriptor.

Table 6.3: Overview of the performance for methods and feature descriptors in 3-FISH-TRN dataset. A checkmark shows in what descriptor configuration (column) the particular method (row) performs the best. The asterisk (*) shows which method performs the best in this particular feature descriptor configuration. This is done separately for ELF and LOMO.

3-FISH-TRN	LOMO			ELF		
	Full	Color	Texture	Full	Color	Texture
KISSME		✓			✓	
KISSME-PSD		✓			✓	
iKISSME		✓			✓	
iKISSME-PSD		✓			✓	
LSSL		✓		✓		
LSSL-PSD		✓		✓		
XQDA			✓		✓	
XQDA-PSD			✓		✓	
DNS-Linear	✓*	✓*	*	✓		
DNS-RBF	✓*	✓*		✓*	✓*	*
L1		✓			✓	
L1-PCA		✓			✓	
L2		✓			✓	
L2-PCA		✓			✓	
Cos. Dist.		✓			✓	
Cos. Dist. - PCA		✓			✓	

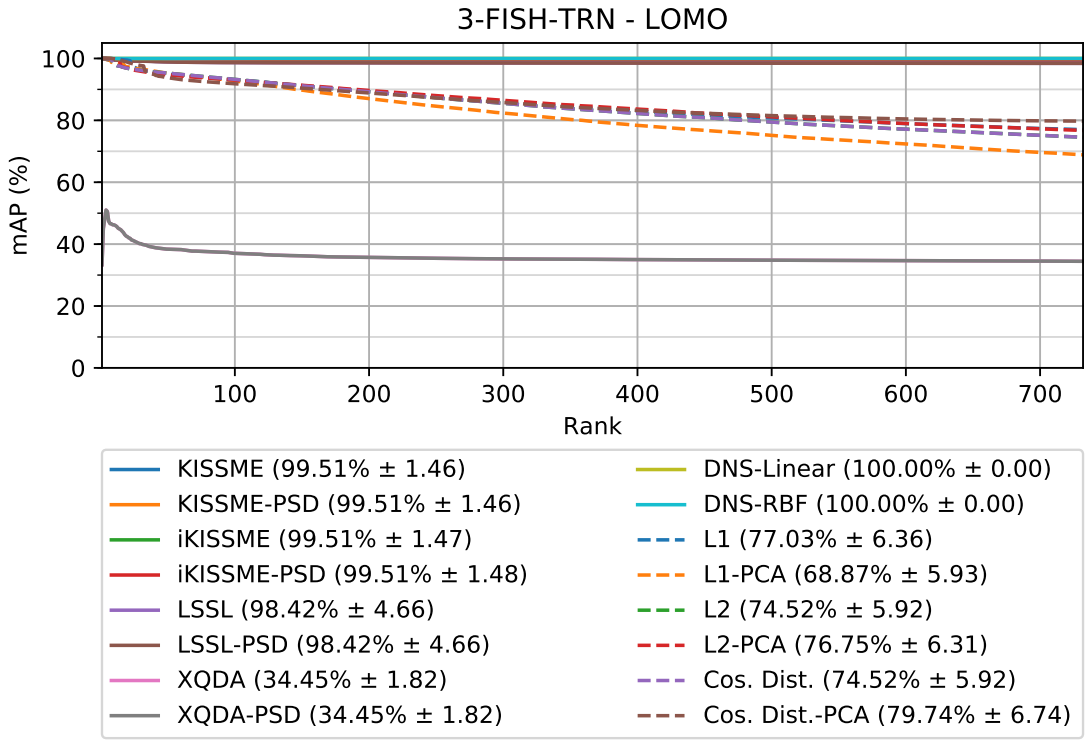


Figure 6.26: mAP results for 3-FISH-TRN with the LOMO feature descriptor.

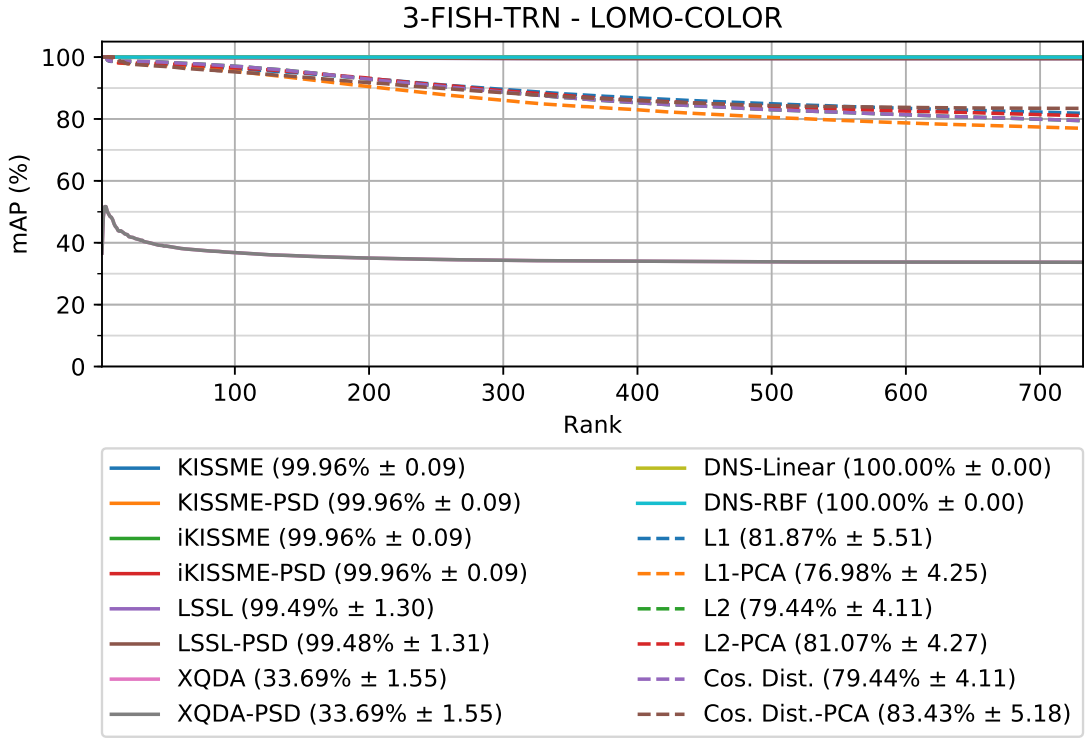


Figure 6.27: mAP results for 3-FISH-TRN with the LOMO-HSV feature descriptor.

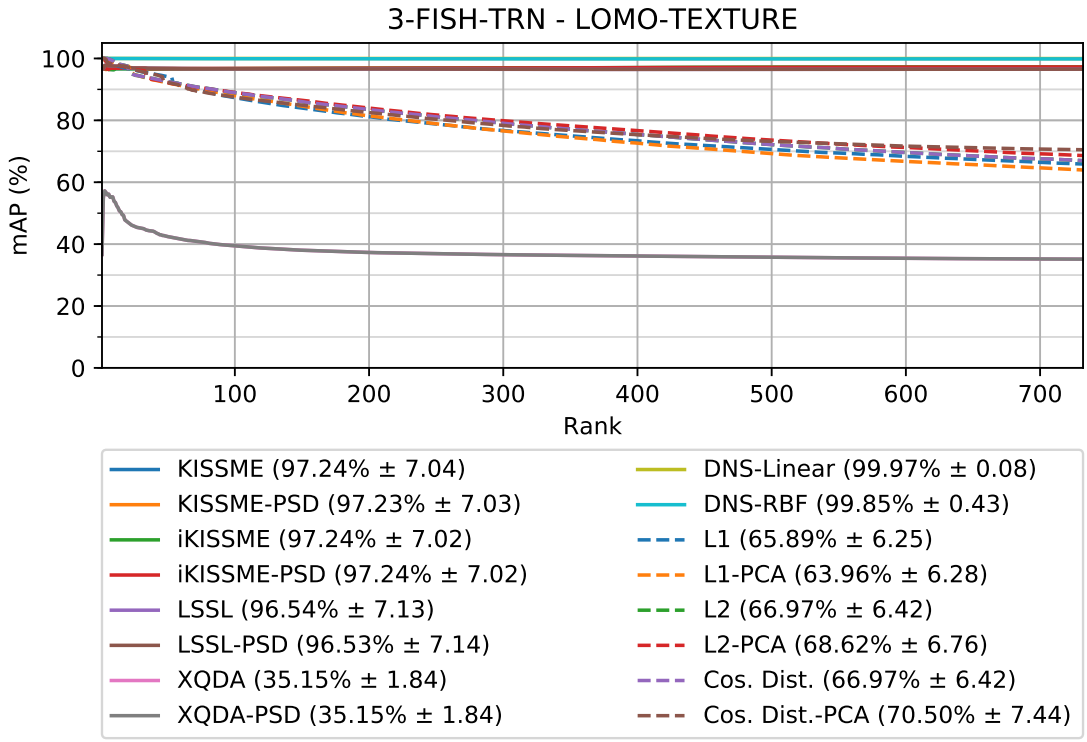


Figure 6.28: mAP results for 3-FISH-TRN with the LOMO-SILTP feature descriptor.

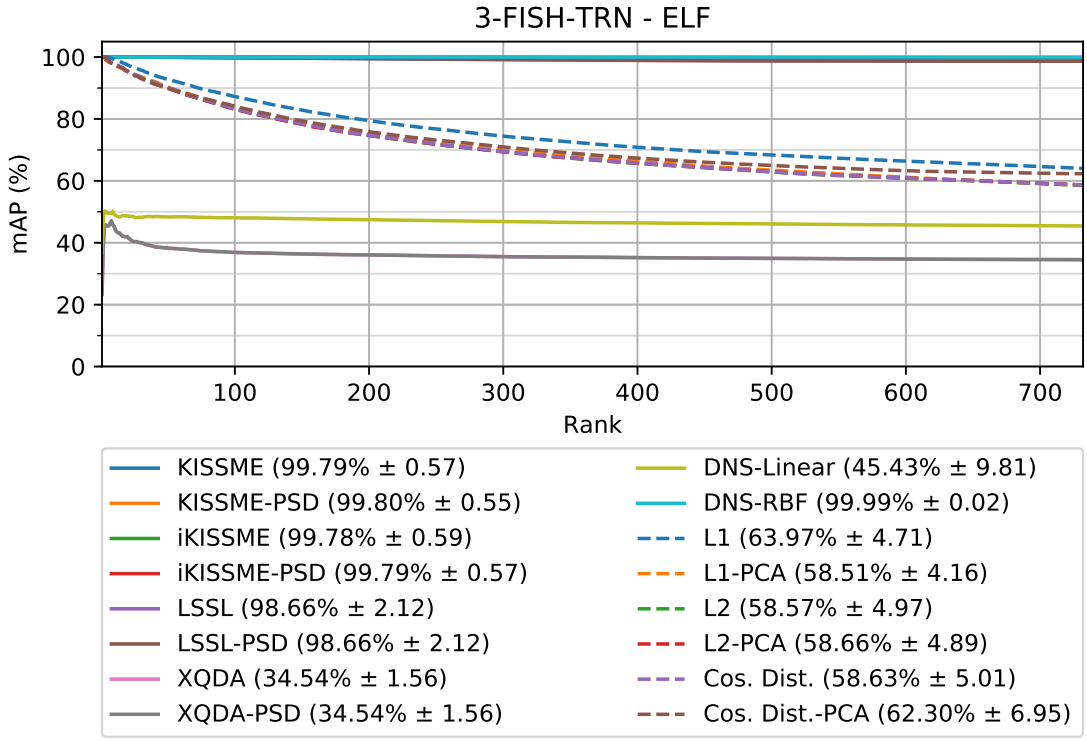


Figure 6.29: mAP results for 3-FISH-TRN with the ELF feature descriptor.

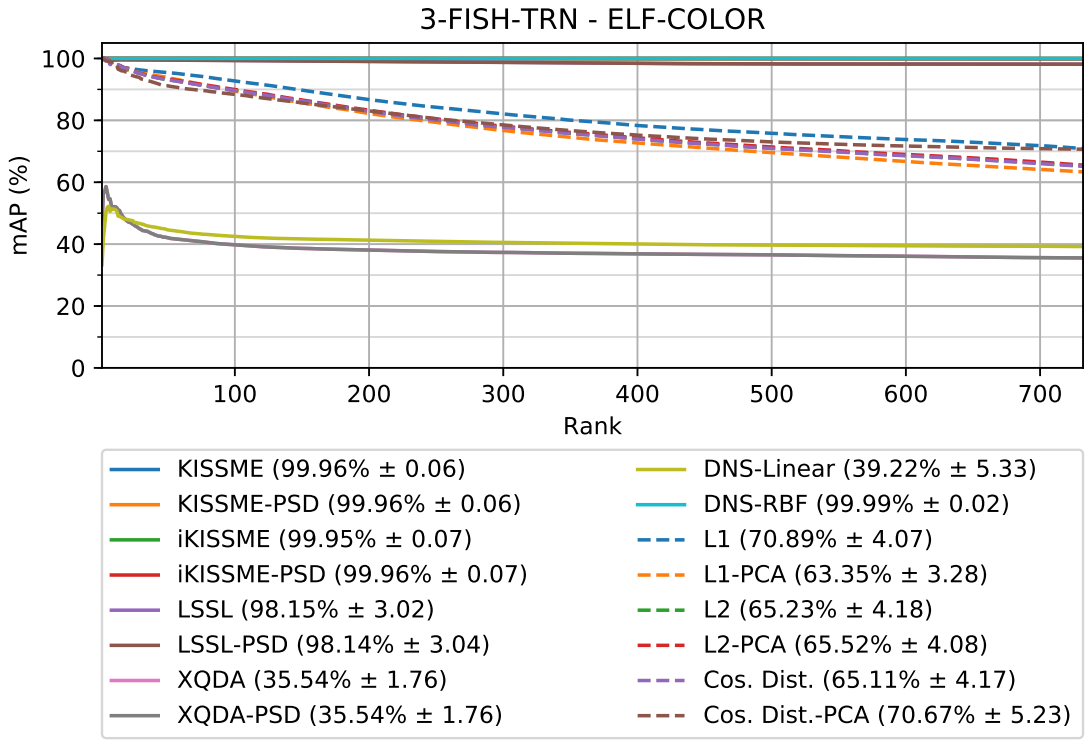


Figure 6.30: mAP results for 3-FISH-TRN with the ELF-COLOR feature descriptor.

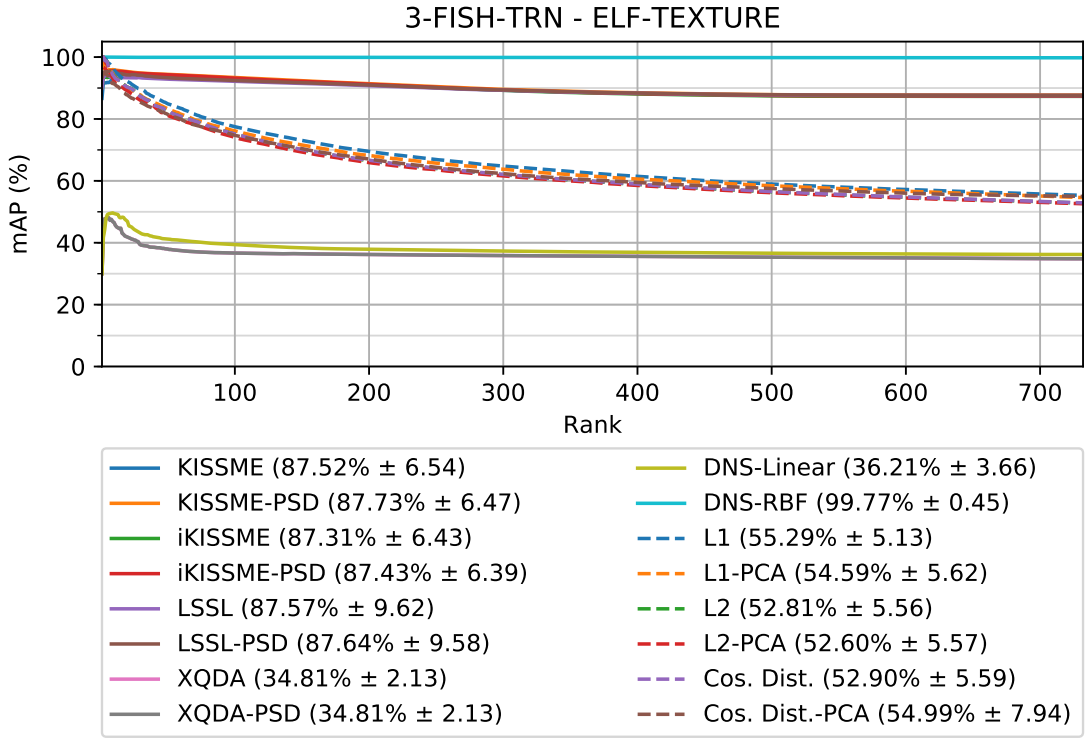


Figure 6.31: mAP results for 3-FISH-TRN with the ELF-TEXTURE feature descriptor.

6.3.4 3-FISH-TST

The results from the baseline test on the 3-FISH-TST dataset are shown in *Figure: 6.32* - *Figure: 6.37*. Overall, the learned metrics perform worse than the baseline metrics the majority of the time. Only in ELF-full the learned metric outperforms the baseline, with DNS-Linear, resulting in 79.88% mAP. The next highest mAP obtained by the learned metrics is 65.34% with the KISSSME-PSD and LOMO-COLOR. On average, the difference between the learned and baseline metrics are within 20 percentage points and can go up to 31,75 percentage points. In this dataset, the texture part does not produce the best results, not in a single example.

Table 6.4: Overview of the performance for methods and feature descriptors in 3-FISH-TST dataset. A checkmark shows in what descriptor configuration (column) the particular method (row) performs the best. The asterisk (*) shows which method performs the best in this particular feature descriptor configuration. This is done separately for ELF and LOMO.

3-FISH-TST	LOMO			ELF		
	Full	Color	Texture	Full	Color	Texture
KISSME		✓			✓	
KISSME-PSD		✓			✓	
iKISSME		✓			✓	
iKISSME-PSD		✓			✓	
LSSL		✓			✓	
LSSL-PSD		✓			✓	
XQDA	✓			✓		
XQDA-PSD	✓			✓		
DNS-Linear		✓		✓*		
DNS-RBF		✓			✓	
L1		✓			✓*	*
L1-PCA		✓		✓		
L2	*	✓*	*		✓	
L2-PCA	✓			✓		
Cos. Dist.	*	✓*	*		✓	
Cos. Dist. - PCA	✓			✓		

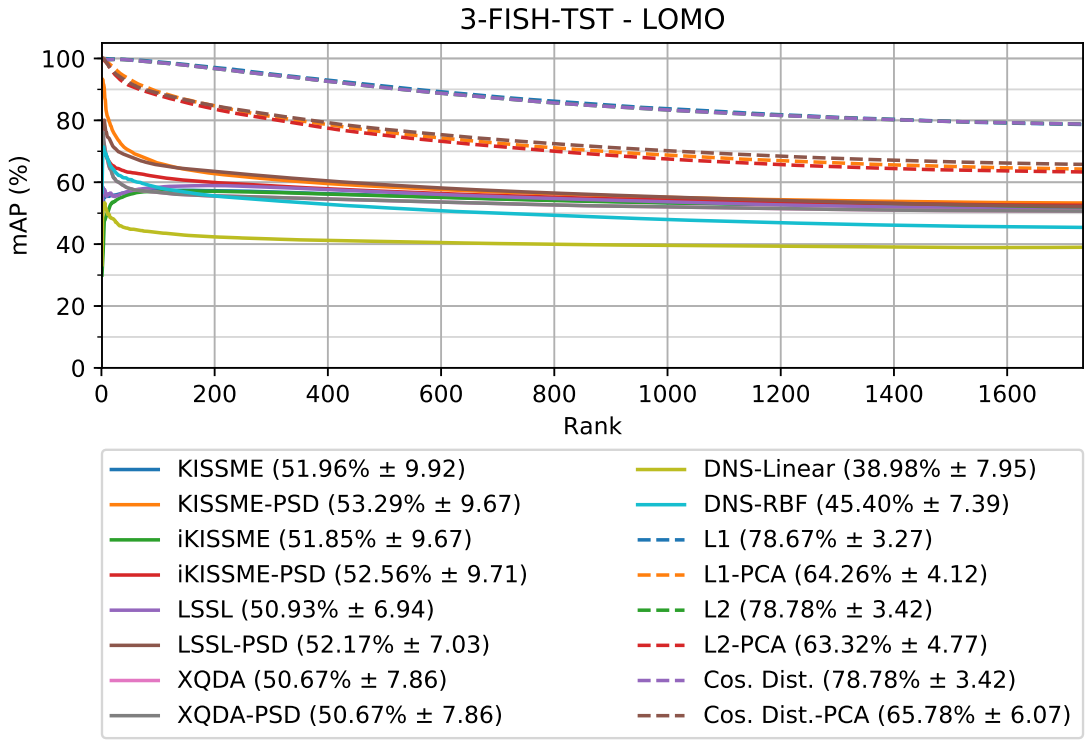


Figure 6.32: mAP results for 3-FISH-TST with the LOMO feature descriptor.

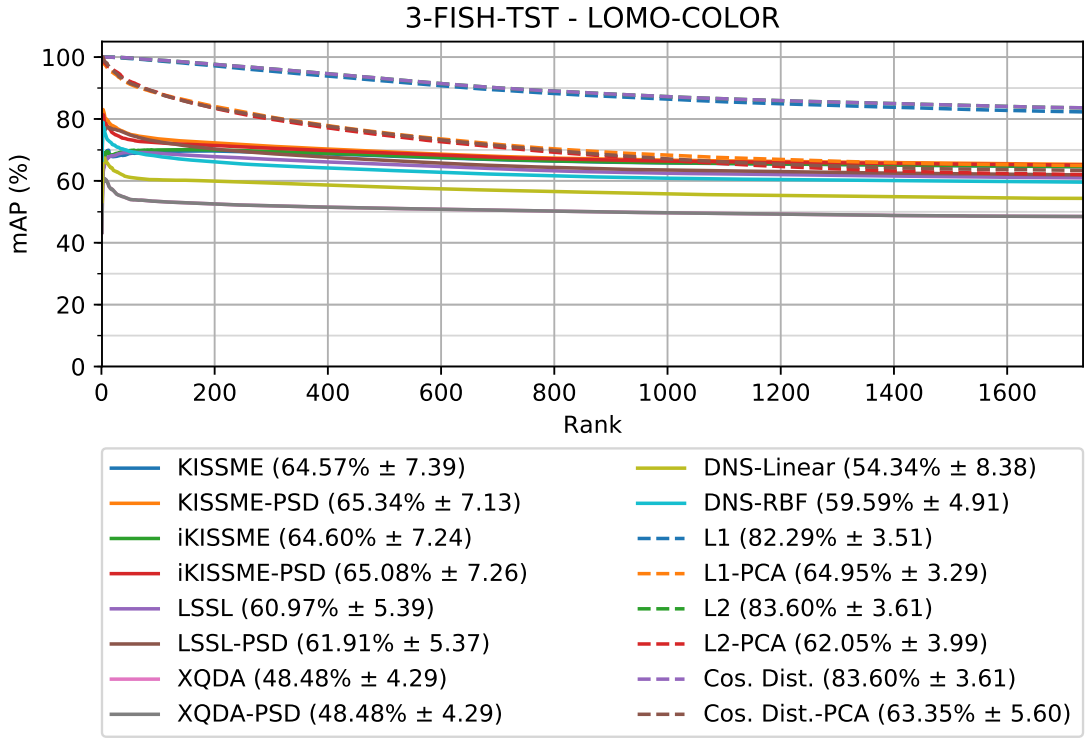


Figure 6.33: mAP results for 3-FISH-TST with the LOMO-HSV feature descriptor.

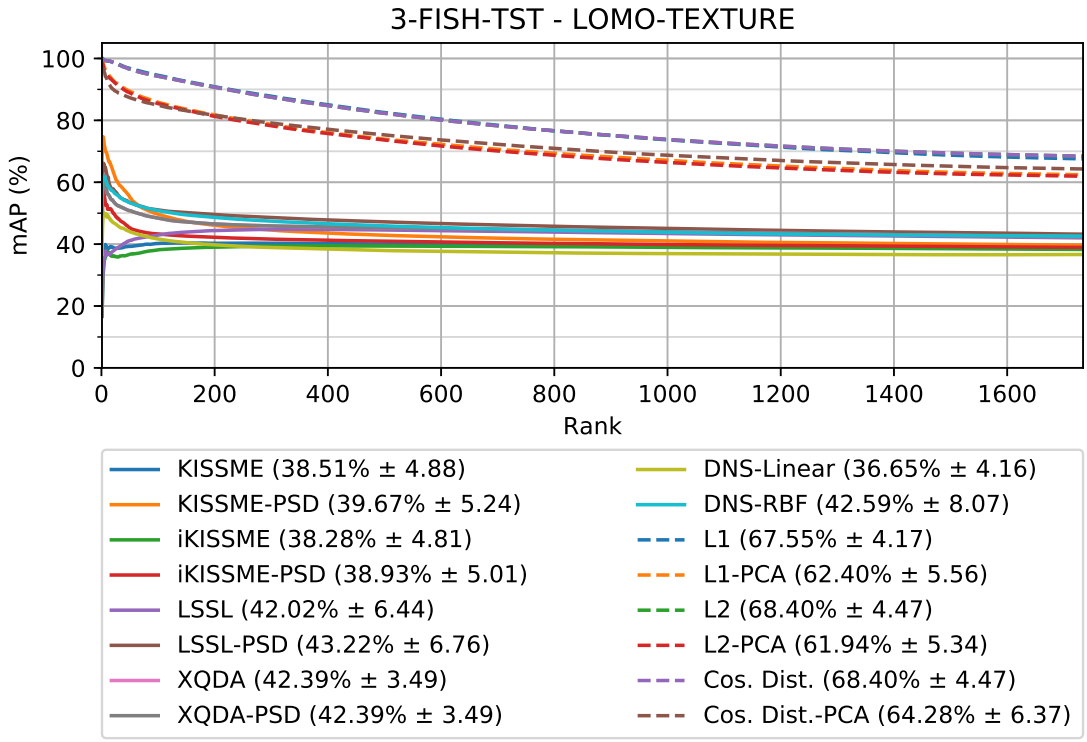


Figure 6.34: mAP results for 3-FISH-TST with the LOMO-SILTP feature descriptor.

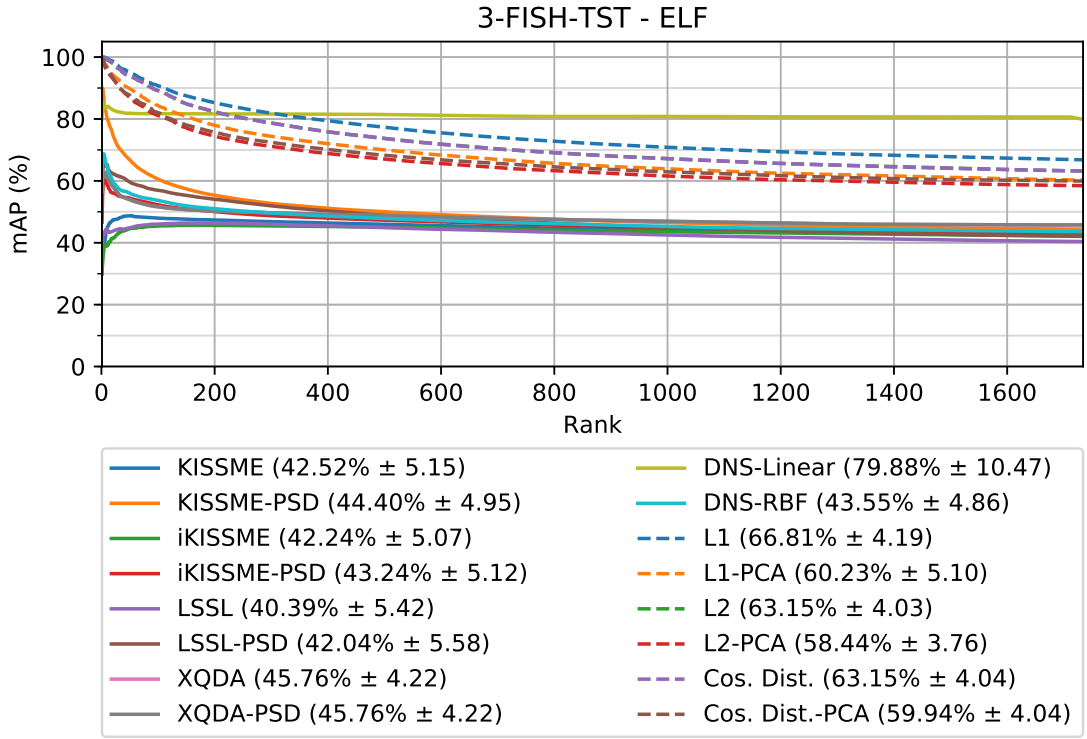


Figure 6.35: mAP results for 3-FISH-TST with the ELF feature descriptor.

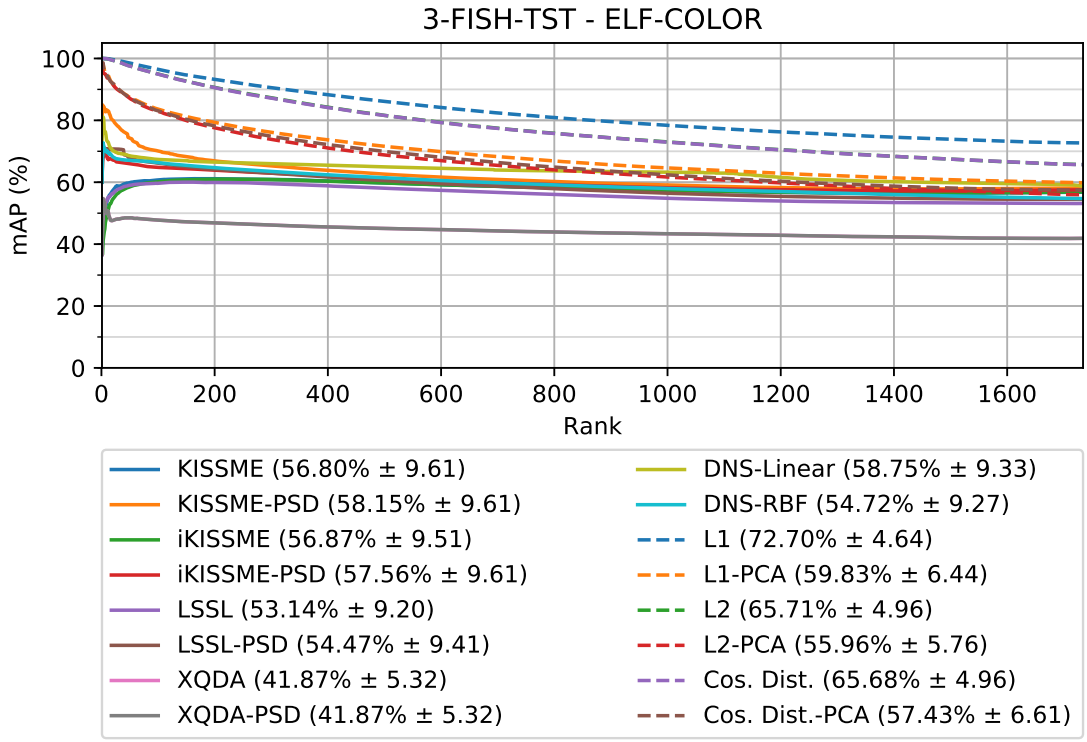


Figure 6.36: mAP results for 3-FISH-TST with the ELF-COLOR feature descriptor.

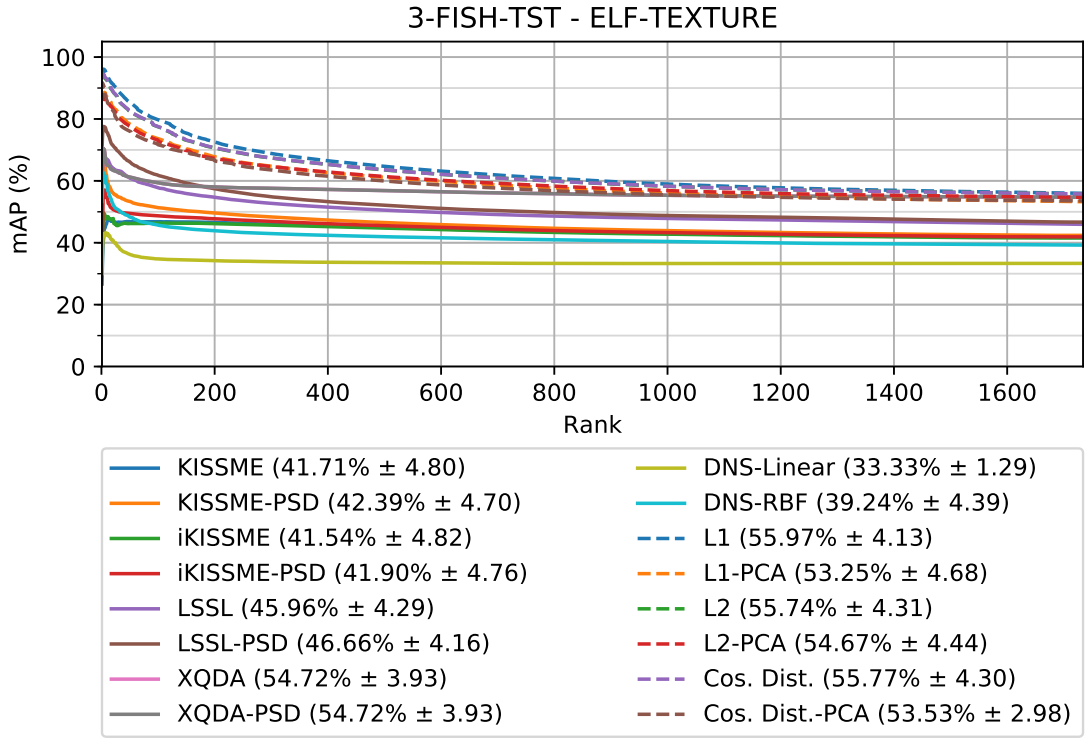


Figure 6.37: mAP results for 3-FISH-TST with the ELF-TEXTURE feature descriptor.

6.4 Sequence Association Test

In order to investigate how well the different approaches can associate tracklets, we designed a test where a sequence of frames before and after an occlusion need to be associated. The effect of the length of the frame sequences is also tested.

Since the dataset is quite fragmented, due to the glitches described in *Section 4.2*, we do not have long continuous tracklets. Therefore, if investigating the effect of a sequence consisting of X frames, we simply choose the first X annotated frames from before and after the occlusion, per fish. This does however mean that the sequences used per fish are not necessarily temporally aligned, as shown in *Figure: 6.38*.

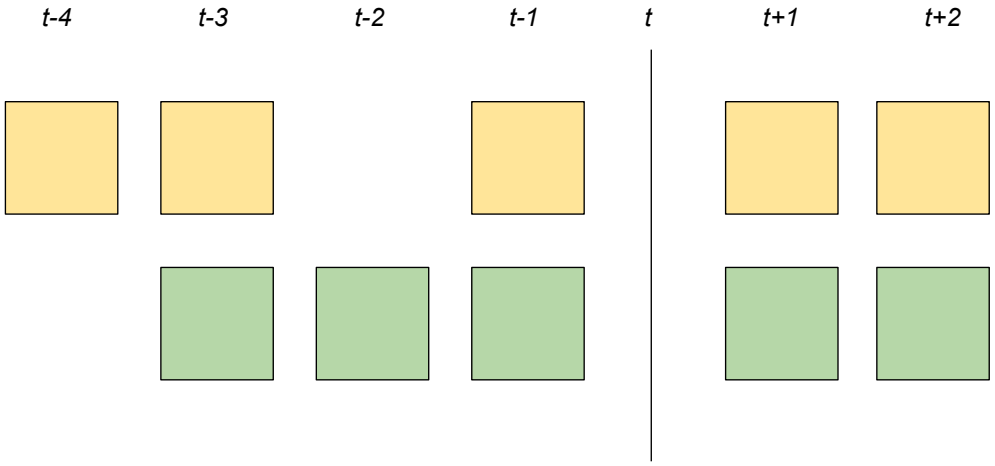


Figure 6.38: Illustration of how the selected sequence per fish are not necessarily temporally aligned. Each colored box signify a feature descriptor at the ith time step, with an occlusion happening at time step t. In the case of the fish, a feature descriptor was not extracted at time step $t-2$.

Furthermore, with the 3 fish datasets, an occlusion does not necessarily involve all fish. In order to simplify the test, we assume that all fish have to be reassigned after an occlusion event i.e. a linear assignment problem is assumed.

Ideally, the sequences should be constricted to only be between the investigated occlusion event and any previous/future occlusion or rotation event. However, due to the low frame rate, this limits the sequence lengths drastically. Therefore, we allow the sequences to extend into other rotation or occlusion events.

The sequence of frames before and after the occlusion are called the *training* and *testing* sequences, respectively, with length N_{train} and N_{test}. The tested methods are trained per occlusion, meaning that we do no look into using data from previous associations. Each fish in the training sequence is associated with a fish in the testing sequence, by solving a cost matrix, the distance matrix, using the Hungarian algorithm (Kuhn, 1955), thereby minimizing the total distance. If there are several frames in the test sequence, the cost matrix is solved per test sequence frame, and a voting scheme is applied. The used voting scheme is quite simple, as it counts how many times a training fish is assigned to each testing fish, and associates the fish so that the total amount of votes used are maximized, while ensuring that all fish are uniquely assigned.

In order to represent the training sequence, three different approaches are tested. Each representation are applied per fish sequence in the training sequences.

1. **Last Frame**: Use the feature representation from the last frame in the training sequence.
2. **Mean**: Calculate the mean feature representation, over the entire training sequence.
3. **Max**: Find the maximum value per element in the feature representations, over the entire training sequence.

The mean and max representations are based on the approach utilized by Zheng et al. (2015) to represent several probe images from the same class in a concise manner.

The sequence test was performed only with the color variations of the ELF and LOMO descriptors, as they performed the best in *Section 6.3*. Furthermore, we only tested on the 2-FISH-TRN and 3-FISH-TRN datasets due to time constraints.

N_{train} was varied from 2 to 50 frames, with a step of 2 frames at a time, while N_{test} was varied from 1 to 10 frames, with a step of 1. This results in a 3D plot which can be hard to interpret. Therefore, we only look at the amount of ID switches when iterating over either the test or training sequence length, while the other sequence length is set to its maximum value. Furthermore, we note the ID switch percentage at the max length sequences for each method. In order of brevity only the best performing feature representation per dataset and feature descriptor is shown in *Figure: 6.39* - *Figure: 6.42*. The remaining results can be found in *Appendix B*.

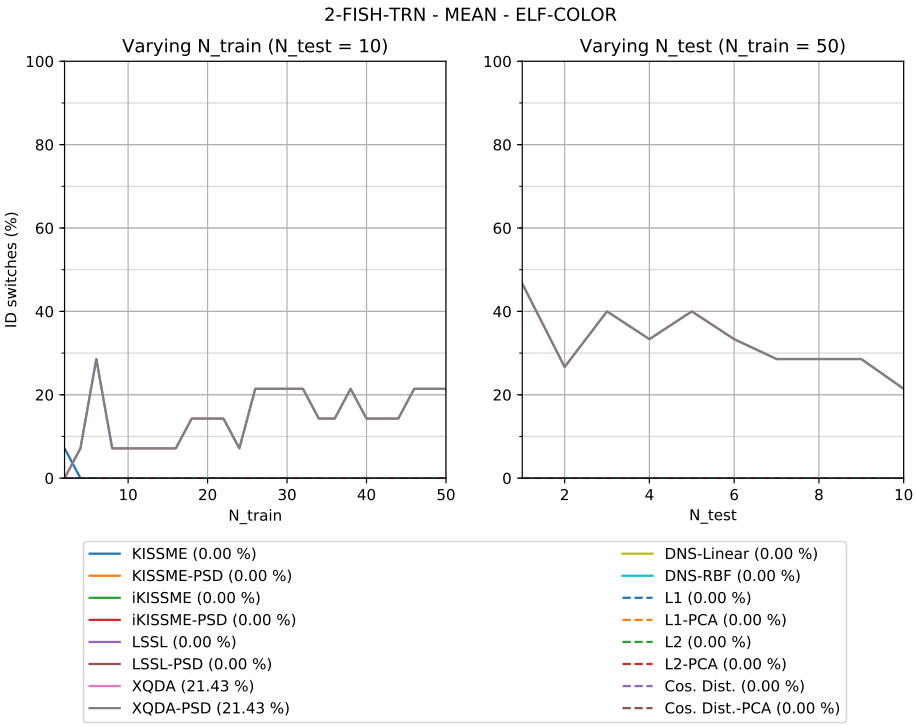


Figure 6.39: Sequence results for 2-FISH-TRN with the ELF-COLOR and "Mean" feature representation

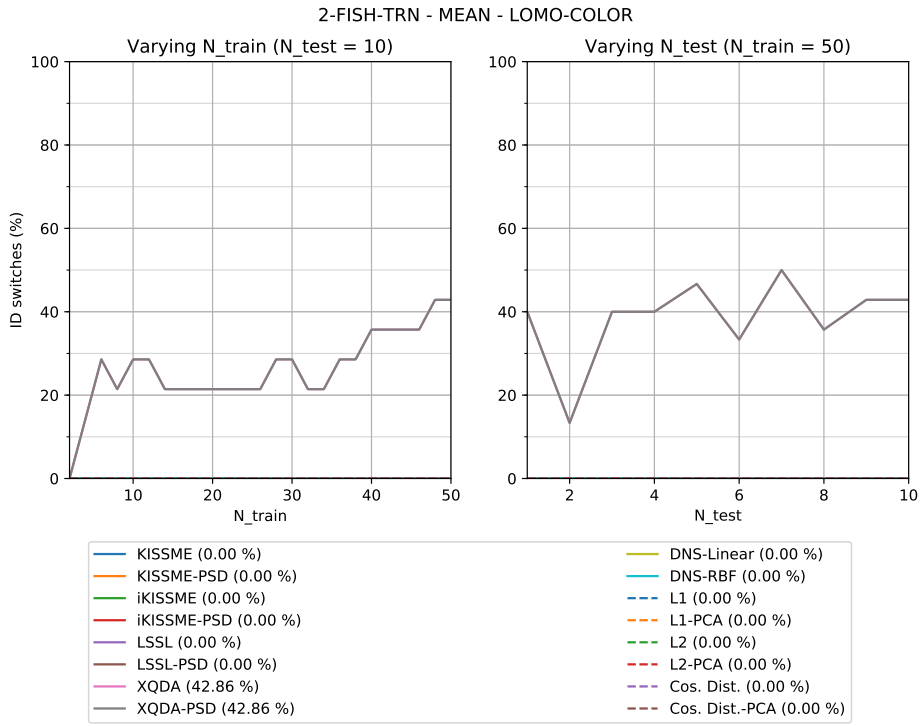


Figure 6.40: Sequence results for 2-FISH-TRN with the LOMO-COLOR and "Mean" feature representation

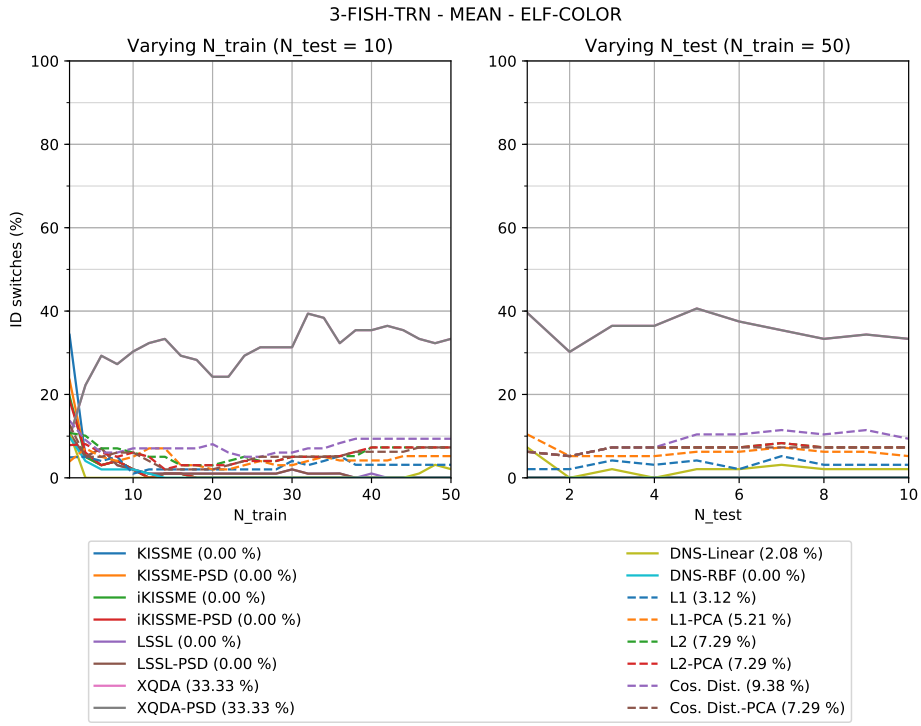


Figure 6.41: Sequence results for 3-FISH-TRN with the ELF-COLOR and "Mean" feature representation

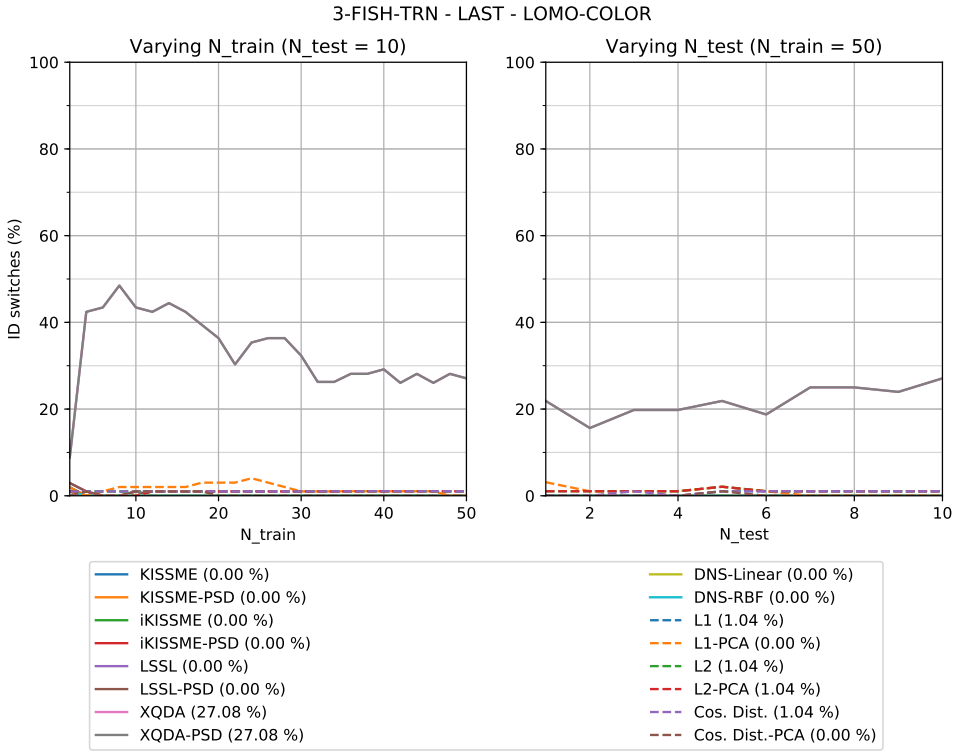


Figure 6.42: Sequence results for 3-FISH-TRN with the LOMO-COLOR and "Last Frame" feature representation

6.5 Discussion

In this section we interpret and discuss the results obtained from the experiments performed on four datasets, namely 2-FISH-TRN, 2-FISH-TST, 3-FISH-TRN and 3-FISH-TST. The section is divided into two sections: results from a generalization test and results from the sequence test.

The XQDA algorithm rarely managed to find a proper subspace projection, the eigenvalues of the eigenvalue problem in *Equation: (5.11)* rarely were above 1. Therefore, the subspace projection was instead just set to an 1D projection, corresponding to the eigenvector with the largest eigenvalue. This resulted in the algorithm assigning all gallery images to a single identity, signifying that the method did not learn properly.

6.5.1 Generalization test results

Overall, the best performance was achieved in 2-FISH-TRN and 3-FISH-TRN. The system can learn the representation of the specific two or three fish very well, achieving 100% in nearly all learned metrics. When with only two fish the results can be altered by bias of having only binary choice for identities; with three fish we reduce that risk of randomness, and still achieve nearly 100% accuracy. The performance suffers, when a completely new fish is introduced to the dataset. In general, the obtained results are better than a random guess, yet far from the best performance of the system on the training datasets. In the case of 2-FISH-TST and 3-FISH-TST, the learned metrics perform worse than the baseline metrics. This is hypothesized to be because the transformation learned through the Mahalanobis matrix, and the subspace projections, are too specific. The transformation that clusters the feature descriptors of the training fish, per fish, might simply also be a transformation which ends up in scattering of the test fish feature descriptors.

Another observation that might have influenced the results of 3 fish datasets is that one fish was a quite smaller size, than two others. Learning representations of them, and then applying to three similar-sized fish may have lower the ability to differentiate the fish.

The results also showed a clear tendency of a color part of both descriptors contributing more often to better results, than its texture part. This did not significantly change between training and testing datasets. This indicates that colors hold more descriptive information than the texture parts. If one look at the fish closely, it is possible to notice that even though the stripes are present and differ from an individual to the individual, the lines can be seen as uniformed lines of blue and yellow colors. There is not much texture along the stripes, yet it may differ in colors throughout the entire line. Specifically, it is thought that it is the color patterns of the fish that are discriminative, as it was found in *Section 6.2.1* that the HSV color values of two fish, while different, were still quite close to each other.

Overall, the algorithm that proved to perform the best in training datasets and partially in testing datasets is DNS-RBF. DNS-Linear also performed well for the LOMO feature descriptor, but generally not for the ELF descriptor. The ELF descriptor representation are less likely to be linearly separable.

The LOMO feature descriptor performs generally better than the ELF descriptor. This could be because the LOMO descriptor representations are closer to being linearly separable, due to its dimensionality being 10 or more times larger than ELF, and more descriptive components.

6.5.2 Sequence test results

The results in *Figure: 6.39* - *Figure: 6.42* show that all methods, except XQDA, perform well across all datasets. The length of the testing sequence has little effect on the number of ID switches, while the training sequence should be at least 10 frames long. In general, all feature representations performed well, but the mean representation seemed to have a slightly better performance in most cases.

While the results indicate that all the KISSME-based algorithms performed well, when the training sequence was quite small, the covariance matrices, which the methods compute, risked becoming ill-conditioned, as indicated by condition numbers that were at times observed to be 10^{16} and above.

In the case that the covariance matrices were singular, which could happen when the number of samples were lower than their dimensionality, a small regularizer value of 0.001, as described by Liao et al. (2015), was added to the diagonal elements of the matrix. In case the covariance matrix was ill-conditioned, the inverse of the covariance matrix was ensured to be symmetric, by adding it with its transpose and dividing by 2. These problems can in the future be circumvented by ensuring that the PCA projection does not result in a higher dimension, than the number of data samples.

When the methods were not forced to be PSD, they would at times results in negative distances. This is hypothesized to be because the covariance matrices were not calculated from enough samples to properly ensure the PSD property. This would be consistent with the statement from Yang et al. (2016) that the PSD constraint is not required for large datasets.

These problems were not encountered for the DNS approach, as it is not dependent on the inversion of any covariance matrices, while also designed to handle a small sample size problem. Furthermore, the caveat that kernel methods have to keep their training data in order to evaluate the testing data, is not considered a problem when associating sequences, as the amount of training data is quite small and might not even have to be saved, if the method are retrained for each sequence.

7 | Conclusion

Zebrafish are highly usable fish for a large variety of problem, but hard to reliably track due to their social behavior and unpredictable movements. Few systems have tried to associate the fish using visual clues, from monochrome top-view images. The purpose of this thesis project has therefore been to investigate whether it is possible to reliably associate zebrafish tracklets, by using color images from a side view, and thereby utilize the potential benefits of this information, when discriminating between fish identities. We have looked into applying metric learning, which was proven to obtain positive results in person re-identification task. Moreover, it requires only a small amount of data, compared to deep learning approaches, which have become state-of-the-art in many computer vision applications.

For this thesis project, we recored six fish for at least 1000 frames. The fish were divided into four datasets: 2 datasets with two fish, and 2 datasets with three fish. Two feature descriptors - LOMO and ELF, consisting of color and texture components, were implemented and used with five metric learning algorithms (with two variations for each) - KISSME, iKISSME, LSSL, XQDA and DNS. All of the methods were chosen from the person re-identification literature, and were compared to the L_1, L_2 and cosine distances, called the baseline metrics.

In the experiments we wanted to see, how well the methods can perform on the same dataset, visualizing its ability to re-identify the identities correctly, when the metric is learned from the same data. Additionally, we wanted to see how well the methods can perform on a new dataset. The results show that using the same dataset produces nearly 100% mAP for two and three fish; and for the new dataset the methods perform worse, able to achieve 89.40% and 86.45% at best, for two and three fish, respectively. The results also show that color component of both feature descriptors can achieve the best performance quite often, whereas the texture component did not perform that well on its own.

The metric learning approaches ability to associate sequences before and after an occlusion were also investigated. This was done by training on small sequences (up to 50 frames) before an occlusion event, and then associating each fish sequences from before and after the occlusion. It is assumed that the number of fish sequences are consistent before and after the occlusion. The metric learning approaches performed the identity assignment nearly flawlessly, proving that they can also work with small amount of training data.

Based on the conducted tests, we conclude that the majority of the tested metric learning approaches perform the re-identification task well, both for large amount of training data and small training sequences. Specifically, it was found that the DNS algorithm performs well. However, the metric learning algorithms do not seem to generalize well when trained on a different set of fish, than those they are tested on. The baseline metrics however performed better in this cases, when using the LOMO descriptor, indicating that the LOMO descriptor is quite discriminative and powerful. The ELF descriptor performed nearly as good as the LOMO descriptor, even though it is approximately 10% the size of the LOMO descriptor.

The color components of the descriptors contributed the most to the results, while the texture elements seemed to have little effect. This is hypothesized to be due to the differing patterns in the color of the fish, while the texture of the fish can roughly be seen as a set of uniformed stripes.

In the future, it would be interesting to see how simple a feature descriptor can be used with the metric learning approaches, and still obtain competitive results. It would also be interesting to investigate how well the algorithms will work when the fish are allowed to move freely in a 3D environment (as in the aquarium without restrictions). Lastly, it could be interesting to integrate the metric learning approaches with the system developed by Bengtson and Pedersen (2017) in order to associate tracklets.

Bibliography

Ahmed, E., Jones, M., and Marks, T. K. (2015). An improved deep learning architecture for person re-identification. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3908–3916.

Ardekani, R., Biyani, A., Dalton, J. E., Saltz, J. B., Arbeitman, M. N., Tower, J., Nuzhdin, S., and Tavaré, S. (2013). Three-dimensional tracking and behaviour monitoring of multiple fruit flies. *Journal of The Royal Society Interface*, 10(78).

Audira, G., Sampurna, B. P., Juniardi, S., Liang, S.-T., Lai, Y.-H., and Hsiao, C.-D. (2018). A simple setup to perform 3d locomotion tracking in zebrafish by using a single camera. *Inventions*, 3(1).

Bak, S. and Carr, P. (2017a). Deep deformable patch metric learning for person re-identification. *IEEE Transactions on Circuits and Systems for Video Technology*, pages 1–1.

Bak, S. and Carr, P. (2017b). One-shot metric learning for person re-identification. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1571–1580.

Baltieri, D., Vezzani, R., and Cucchiara, R. (2011). 3dpes: 3d people dataset for surveillance and forensics. In *Proceedings of the 2011 Joint ACM Workshop on Human Gesture and Behavior Understanding*, J-HGBU '11, pages 59–64, New York, NY, USA. ACM.

Bellet, A., Habrard, A., and Sebban, M. (2015). *Metric Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

Bengtson, S. H. and Pedersen, M. (2017). Tracking Zebrafish in 3D using Stereo Vision. Master's thesis, Aalborg University, Denmark.

Bernardin, K. and Stiefelhagen, R. (2008). Evaluating multiple object tracking performance: The clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008(1):246309.

Chen, Y. C., Zheng, W. S., and Lai, J. (2015). Mirror representation for modeling view-specific transform in person re-identification. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 3402–3408.

Cheng, X. E., Qian, Z.-M., Wang, S. H., Jiang, N., Guo, A., and Chen, Y. Q. (2015). A novel method for tracking individuals of fruit fly swarms flying in a laboratory flight arena. *PLOS ONE*, 10(6):1–18.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1.

Donahue, J. (2014). CaffeNet. `https://github.com/BVLC/caffe/tree/master/models/bvlc_reference_caffenet`. Accessed: 2018-05-30.

Egan, R. J., Bergner, C. L., Hart, P. C., Cachat, J. M., Canavello, P. R., Elegante, M. F., Elkhayat, S. I., Bartels, B. K., Tien, A. K., Tien, D. H., Mohnot, S., Beeson, E., Glasgow, E., Amri, H., Zukowska, Z., and Kalueff, A. V. (2009). Understanding behavioral and physiological phenotypes of stress and anxiety in zebrafish. *Behav Brain Res*, 205(1):38–44. 19540270[pmid].

Fogel, I. and Sagi, D. (1989). Gabor filters as texture discriminator. *Biological Cybernetics*, 61(2):103–113.

Gray, D., Brennan, S., and Tao, H. (2007). Evaluating appearance models for recognition, reacquisition, and tracking. In *10th IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS)*.

Gray, D. and Tao, H. (2008). Viewpoint invariant pedestrian recognition with an ensemble of localized features. In Forsyth, D., Torr, P., and Zisserman, A., editors, *Computer Vision – ECCV 2008*, pages 262–275, Berlin, Heidelberg. Springer Berlin Heidelberg.

Green, J., Collins, C., Kyzar, E. J., Pham, M., Roth, A., Gaikwad, S., Cachat, J., Stewart, A. M., Landsman, S., Grieco, F., Tegelenbosch, R., Noldus, L. P., and Kalueff, A. V. (2012). Automated high-throughput neurophenotyping of zebrafish social behavior. *Journal of Neuroscience Methods*, 210(2):266 – 271.

Guo, Y.-F., Wu, L., Lu, H., Feng, Z., and Xue, X. (2006). Null foley–sammon transform. *Pattern Recognition*, 39(11):2248 – 2251.

Hill, A. J., Teraoka, H., Heideman, W., and Peterson, R. E. (2005). Zebrafish as a model vertebrate for investigating chemical toxicity. *Toxicological Sciences*, 86(1):6–19.

Hirzer, M., Beleznai, C., Roth, P. M., and Bischof, H. (2011). Person re-identification by descriptive and discriminative classification. In Heyden, A. and Kahl, F., editors, *Image Analysis*, pages 91–102, Berlin, Heidelberg. Springer Berlin Heidelberg.

Howe, K., Clark, M. D., Torroja, C. F., Torrance, J., Berthelot, C., Muffato, M., Collins, J. E., Humphray, S., McLaren, K., Matthews, L., McLaren, S., Sealy, I., Caccamo, M., Churcher, C., Scott, C., Barrett, J. C., Koch, R., Rauch, G.-J., White, S., Chow, W., Kilian, B., Quintais, L. T., Guerra-Assunção, J. A., Zhou, Y., Gu, Y., Yen, J., Vogel, J.-H., Eyre, T., Redmond, S., Banerjee, R., Chi, J., Fu, B., Langley, E., Maguire, S. F., Laird, G. K., Lloyd, D., Kenyon, E., Donaldson, S., Sehra, H., Almeida-King, J., Loveland, J., Trevanion, S., Jones, M., Quail, M., Willey, D., Hunt, A., Burton, J., Sims, S., McLay, K., Plumb, B., Davis, J., Clee, C., Oliver, K., Clark, R., Riddle, C., Elliot, D., Threadgold, G., Harden, G., Ware, D., Begum, S., Mortimore, B., Kerry, G., Heath, P., Phillimore, B., Tracey, A., Corby, N., Dunn, M., Johnson, C., Wood, J., Clark, S., Pelan, S., Griffiths, G., Smith, M., Glithero, R., Howden, P., Barker, N., Lloyd, C., Stevens, C., Harley, J., Holt, K., Panagiotidis, G., Lovell, J., Beasley, H., Henderson, C., Gordon, D., Auger, K., Wright, D., Collins, J., Raisen, C., Dyer, L., Leung, K., Robertson, L., Ambridge, K., Leongamornlert, D., McGuire, S., Gilderthorp, R., Griffiths, C., Manthravadi, D., Nichol, S., Barker, G., Whitehead, S., Kay, M., Brown, J., Murnane, C., Gray, E., Humphries, M., Sycamore, N., Barker, D., Saunders, D., Wallis, J., Babbage, A., Hammond, S., Mashreghi-Mohammadi, M., Barr, L., Martin, S., Wray, P., Ellington, A., Matthews, N., Ellwood, M., Woodmansey, R., Clark, G., Cooper, J. D., Tromans, A., Grafham, D., Skuce, C., Pandian, R., Andrews, R., Harrison, E., Kimberley, A., Garnett, J., Fosker, N., Hall, R., Garner, P., Kelly, D., Bird, C., Palmer, S., Gehring, I., Berger, A., Dooley, C. M., Ersan-Ürün, Z., Eser, C., Geiger, H., Geisler, M., Karotki, L., Kirn, A., Konantz, J., Konantz, M., Oberländer, M., Rudolph-Geiger, S., Teucke, M., Lanz, C., Raddatz, G., Osoegawa, K., Zhu, B., Rapp, A., Widaa, S., Langford, C., Yang, F., Schuster, S. C., Carter, N. P., Harrow, J., Ning, Z., Herrero, J., Searle, S. M. J., Enright, A., Geisler, R., Plasterk, R. H. A., Lee, C., Westerfield, M., de Jong, P. J., Zon, L. I., Postlethwait, J. H., Nüsslein-Volhard, C., Hubbard, T. J. P., Roest Crollius, H., Rogers, J., and Stemple, D. L. (2013). The zebrafish reference genome sequence and its relationship to the human genome. *Nature*, 496(7446):498—503.

IDS (n.d.). Ui-3070cp-c-hq rev.2 datasheet. https://en.ids-imaging.com/IDS/datasheet_pdf.php?sku=AB00968. Accessed: 2018-05-30.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 448–456. JMLR.org.

Joachims, T., Finley, T., and Yu, C.-N. J. (2009). Cutting-plane training of structural svms. *Machine Learning*, 77(1):27–59.

Jobson, D. J., Rahman, Z., and Woodell, G. A. (1997). A multiscale retinex for bridging the gap between color images and the human observation of scenes. *IEEE Transactions on Image Processing*, 6(7):965–976.

Kalueff, A. V., Stewart, A. M., and Gerlai, R. (2014). Zebrafish as an emerging model for studying complex brain disorders. *Trends in Pharmacological Sciences*, 35(2):63–75.

Kawanishi, Y., Wu, Y., Mukunoki, M., and Minoh, M. (2014). Shinpuhkan2014: A Multi-Camera Pedestrian Dataset for Tracking People across Multiple Cameras. In *The 20th Korea-Japan Joint Workshop on Frontiers of Computer Vision*, pages 1–5.

Kino Flo (2017a). Diva-lite 401 datasheet. https://www.kinoflo.com/PDF/Spec%20Sheet/2017%20Diva-Lite%20401%207-07-2017.pdf. Accessed: 2018-05-31.

Kino Flo (2017b). Diva-lite 401 quick start guide. https://www.kinoflo.com/PDF/News/Operational%20Manual%20for%20the%20web/3100048%20Diva-Lite%20401%20Rev%20D%206-09-2017.pdf. Accessed: 2018-05-31.

Klaser, A., Marszalek, M., and Schmid, C. (2008). A Spatio-Temporal Descriptor Based on 3D-Gradients. In Everingham, M., Needham, C., and Fraile, R., editors, *BMVC 2008 - 19th British Machine Vision Conference*, pages 275:1–10, Leeds, United Kingdom. British Machine Vision Association.

Korenius, T., Laurikkala, J., and Juhola, M. (2007). On principal component analysis, cosine and euclidean measures in information retrieval. *Information Sciences*, 177(22):4893 – 4905.

KOWA (n.d.a). Kowa lm16hc datasheet. https://lenses.kowa-usa.com/hc-series/474-lm16hc.html. Accessed: 2018-05-30.

KOWA (n.d.b). Kowa lm25hc datasheet. https://lenses.kowa-usa.com/hc-series/475-lm25hc.html. Accessed: 2018-05-30.

KOWA (n.d.c). Kowa lm8hc datasheet. https://lenses.kowa-usa.com/hc-series/472-lm8hc.html. Accessed: 2018-05-30.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97.

Köstinger, M., Hirzer, M., Wohlhart, P., Roth, P. M., and Bischof, H. (2012). Large scale metric learning from equivalence constraints. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2288–2295.

Li, N., Jin, R., and Zhou, Z.-H. (2014a). Top rank optimization in linear time. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 1502–1510. Curran Associates, Inc.

Li, W., Zhao, R., and Wang, X. (2013). Human reidentification with transferred metric learning. In Lee, K. M., Matsushita, Y., Rehg, J. M., and Hu, Z., editors, *Computer Vision – ACCV 2012*, pages 31–44, Berlin, Heidelberg. Springer Berlin Heidelberg.

Li, W., Zhao, R., Xiao, T., and Wang, X. (2014b). Deepreid: Deep filter pairing neural network for person re-identification. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 152–159.

Liao, S., Hu, Y., Zhu, X., and Li, S. Z. (2015). Person re-identification by local maximal occurrence representation and metric learning. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2197–2206.

Liao, S., Zhao, G., Kellokumpu, V., Pietikäinen, M., and Li, S. Z. (2010). Modeling pixel process with scale invariant local patterns for background subtraction in complex scenes. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1301–1306.

MacRì, S., Neri, D., Ruberto, T., Mwaffo, V., Butail, S., and Porfiri, M. (2017). Three-dimensional scoring of zebrafish behavior unveils biological phenomena hidden by two-dimensional analyses. *Scientific Reports*, 7(1):1–10.

McLaughlin, N., d. Rincon, J. M., and Miller, P. (2016). Recurrent convolutional network for video-based person re-identification. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1325–1334.

Miller, N. and Gerlai, R. (2012). From schooling to shoaling: Patterns of collective motion in zebrafish (danio rerio). *PLOS ONE*, 7(11):1–6.

Møgelmose, A. and Bahnsen, C. H. (2018). Aau vap bounding box annotator. `https://bitbucket.org/aauvap/bounding-box-annotator/`. Accessed: 2018-05-31.

Ohayon, S., Avni, O., Taylor, A. L., Perona, P., and Egnor, S. R. (2013). Automated multi-day tracking of marked mice for the analysis of social behaviour. *Journal of Neuroscience Methods*, 219(1):10 – 19.

Ojala, T., Pietikainen, M., and Harwood, D. (1994). Performance evaluation of texture measures with classification based on kullback discrimination of distributions. In *Proceedings of 12th International Conference on Pattern Recognition*, volume 1, pages 582–585 vol.1.

Ojala, T., Pietikainen, M., and Maenpaa, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987.

Pérez-Escudero, A., Vicente-Page, J., Hinz, R. C., Arganda, S., and De Polavieja, G. G. (2014). IdTracker: Tracking individuals in a group by automatic identification of unmarked animals. *Nature Methods*, 11(7):743–748.

Qian, Z.-M. and Chen, Y. Q. (2017). Feature point based 3d tracking of multiple fish from multi-view images. *PLOS ONE*, 12(6):1–18.

Qian, Z.-M., Cheng, X. E., and Chen, Y. Q. (2014). Automatically detect and track multiple fish swimming in shallow water with frequent occlusion. *PLOS ONE*, 9(9):1–12.

Rowena, S., Gabriele, G., Christian, L., and Carl, S. (2007). The behaviour and ecology of the zebrafish, danio rerio. *Biological Reviews*, 83(1):13–34.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

Sadeghian, A., Alahi, A., and Savarese, S. (2017). Tracking the untrackable: Learning to track multiple cues with long-term dependencies. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 300–311.

Schmid, C. (2001). Constructing models for content-based image retrieval. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 2, pages II–39–II–45 vol.2.

Schölkopf, B., Smola, A., and Müller, K.-R. (1997). Kernel principal component analysis. In Gerstner, W., Germond, A., Hasler, M., and Nicoud, J.-D., editors, *Artificial Neural Networks — ICANN'97*, pages 583–588, Berlin, Heidelberg. Springer Berlin Heidelberg.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826.

van de Weijer, J., Schmid, C., Verbeek, J., and Larlus, D. (2009). Learning color names for real-world applications. *IEEE Transactions on Image Processing*, 18(7):1512–1523.

Wang, S. H., Zhao, J. W., and Chen, Y. Q. (2017). Robust tracking of fish schools using cnn for head identification. *Multimedia Tools and Applications*, 76(22):23679–23697.

Wang, T., Gong, S., Zhu, X., and Wang, S. (2014). Person re-identification by video ranking. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Computer Vision – ECCV 2014*, pages 688–703, Cham. Springer International Publishing.

Weinberger, K. Q., Blitzer, J., and Saul, L. K. (2006). Distance metric learning for large margin nearest neighbor classification. In Weiss, Y., Schölkopf, B., and Platt, J. C., editors, *Advances in Neural Information Processing Systems 18*, pages 1473–1480. MIT Press.

Wixon, J. (2000). Danio rerio, the zebrafish. *Yeast*, 17(3):225–231. 11025533[pmid].

Wu, S., Chen, Y. C., Li, X., Wu, A. C., You, J. J., and Zheng, W. S. (2016). An enhanced deep feature representation for person re-identification. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–8.

Xiao, T., Li, H., Ouyang, W., and Wang, X. (2016). Learning deep feature representations with domain guided dropout for person re-identification. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1249–1258.

Xu, Z. and Cheng, X. E. (2017). Zebrafish tracking using convolutional neural networks. *Scientific Reports*, 7:1–11.

Yang, Y., Liao, S., Lei, Z., and Li, S. Z. (2016). Large scale similarity learning using similar pairs for person verification. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 3655–3661. AAAI Press.

Yang, Y., Yang, J., Yan, J., Liao, S., Yi, D., and Li, S. Z. (2014). Salient color names for person re-identification. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Computer Vision – ECCV 2014*, pages 536–551, Cham. Springer International Publishing.

You, J., Wu, A., Li, X., and Zheng, W. S. (2016). Top-push video-based person re-identification. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1345–1353.

Zhang, L., Xiang, T., and Gong, S. (2016). Learning a discriminative null space for person re-identification. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1239–1248.

Zheng, L., Shen, L., Tian, L., Wang, S., Wang, J., and Tian, Q. (2015). Scalable person re-identification: A benchmark. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1116–1124.

Zheng, W.-S., Gong, S., and Xiang, T. (2009). Associating groups of people. In *Proceedings of the British Machine Vision Conference*, pages 23.1–23.11. BMVA Press. doi:10.5244/C.23.23.

Zhu, J., Zeng, H., Liao, S., Lei, Z., Cai, C., and Zheng, L. X. (2017). Deep hybrid similarity learning for person re-identification. *IEEE Transactions on Circuits and Systems for Video Technology*, pages 1–1.

Appendices

A | Experiment Results - Generalization Test

In this appendix the results of the tested datasets are shown when the images were not flipped, i.e. the fish are not pointing the same direction in the images. The results are in general worse compared to the results in *Section 6.3*, but better than expected.

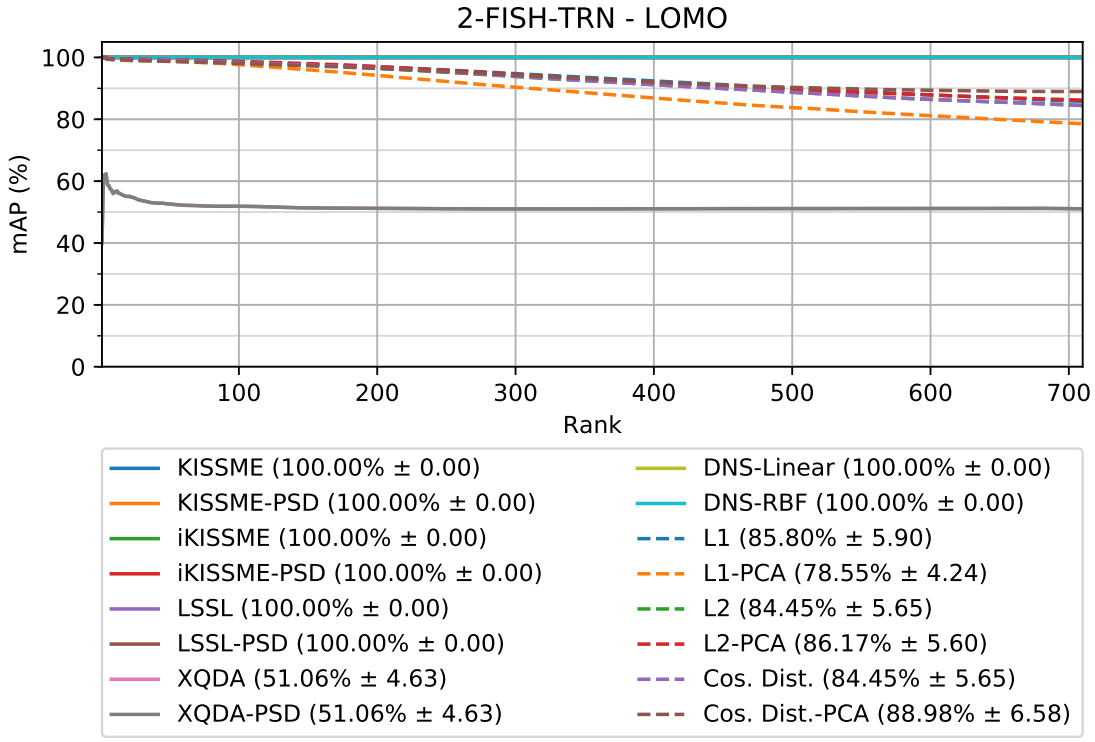


Figure A.1: mAP results for 2-FISH-TRN with the LOMO feature descriptor.

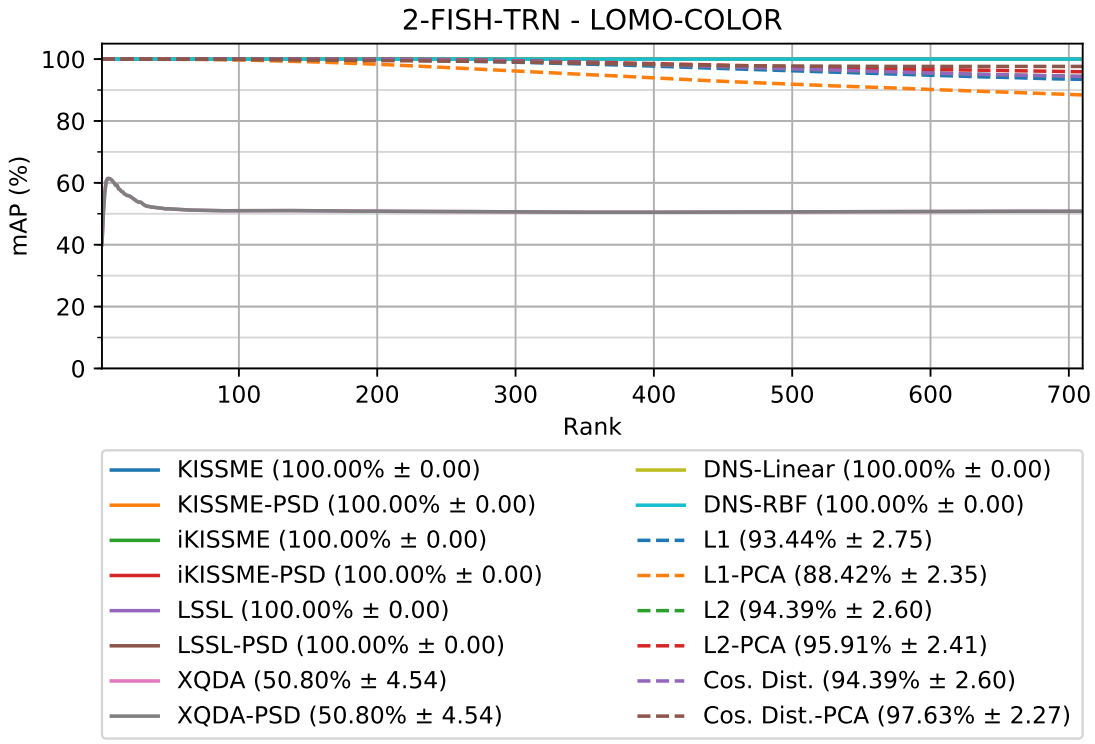


Figure A.2: mAP results for 2-FISH-TRN with the LOMO-COLOR feature descriptor.

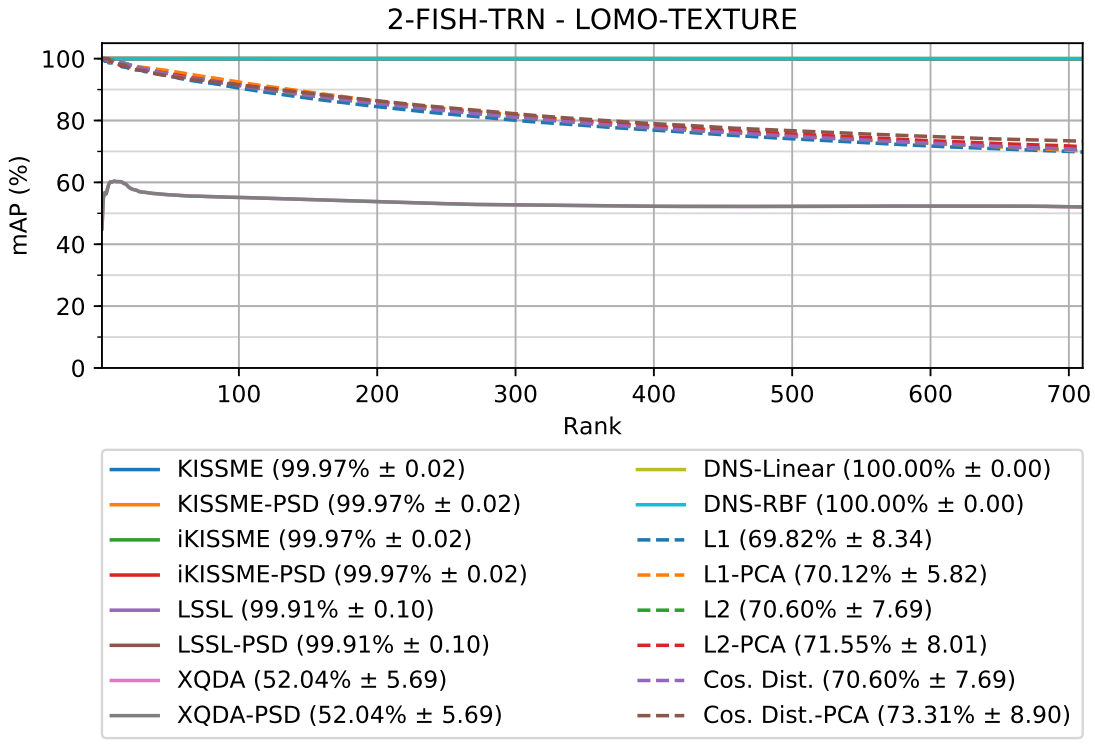


Figure A.3: mAP results for 2-FISH-TRN with the LOMO-TEXTURE feature descriptor.

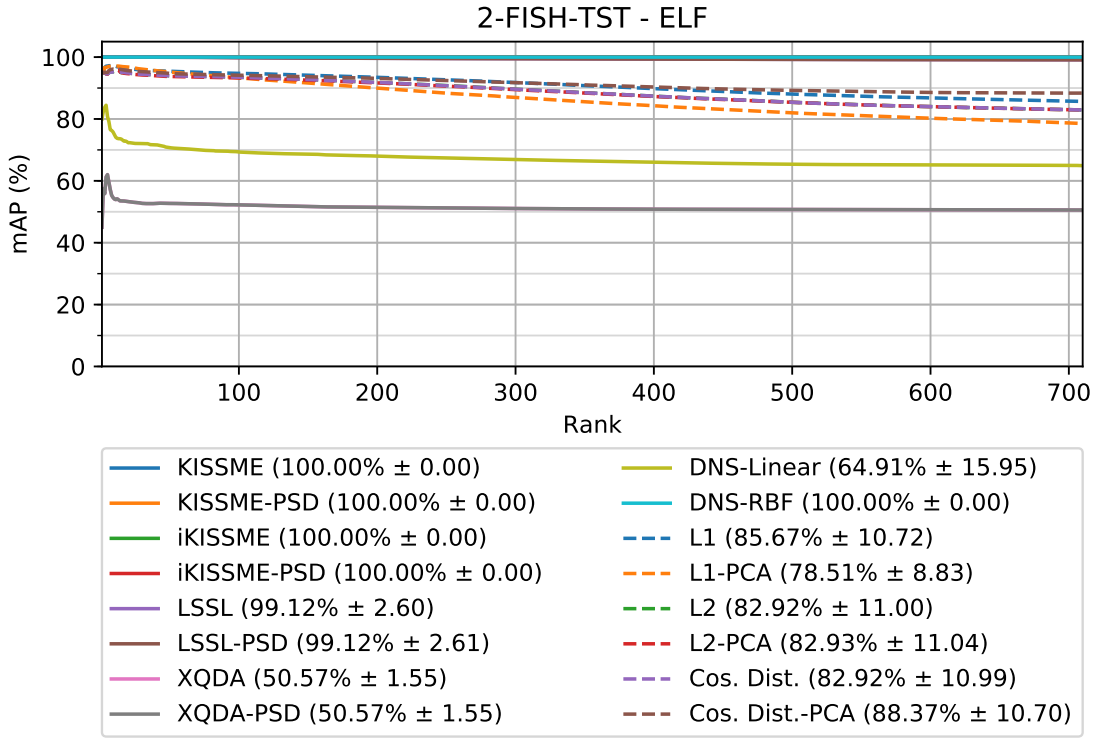


Figure A.4: mAP results for 2-FISH-TRN with the ELF feature descriptor.

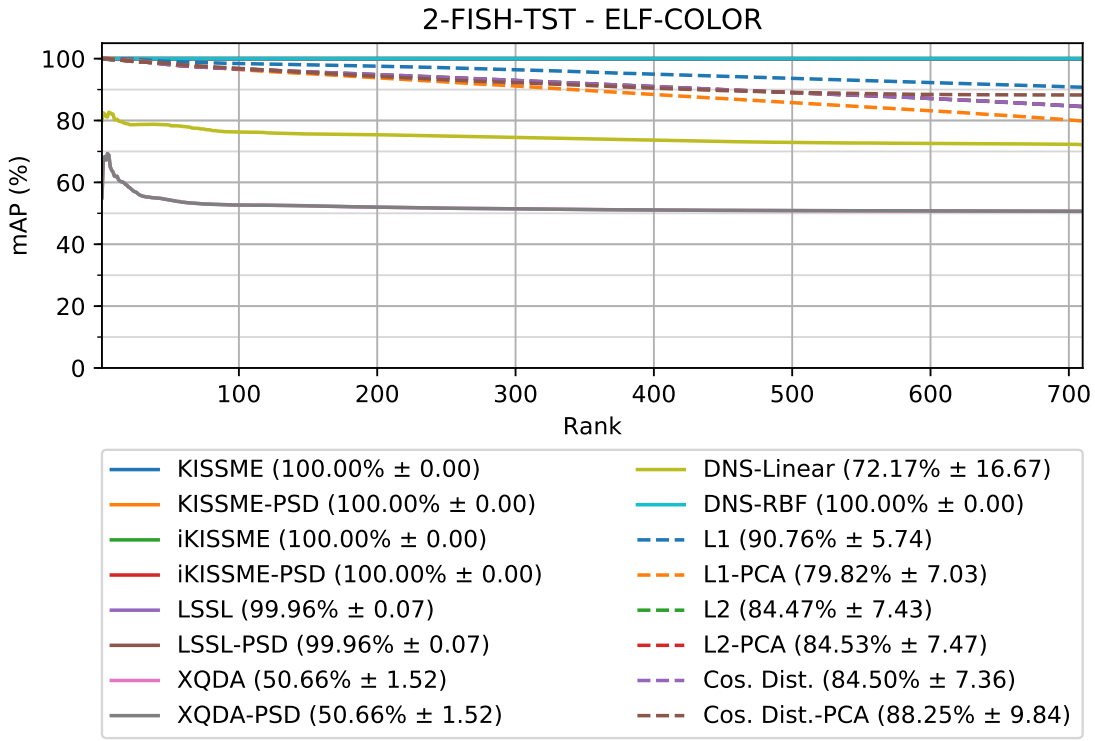


Figure A.5: mAP results for 2-FISH-TRN with the ELF-COLOR feature descriptor.

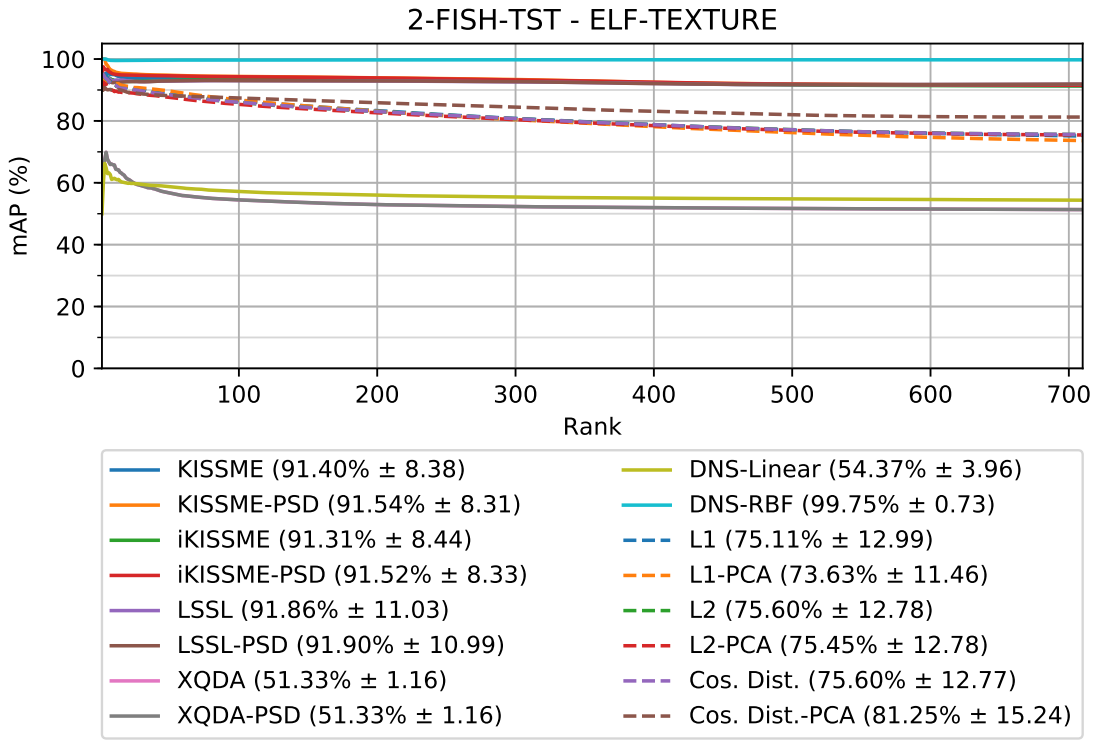


Figure A.6: mAP results for 2-FISH-TRN with the ELF-TEXTURE feature descriptor.

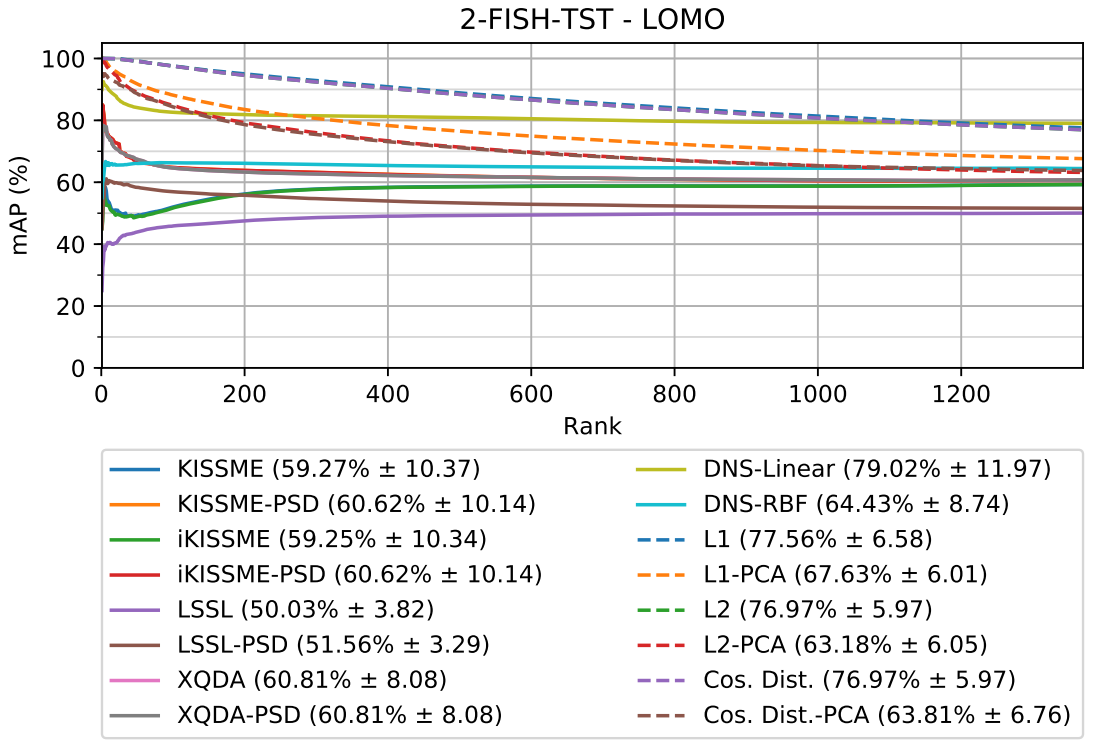


Figure A.7: mAP results for 2-FISH-TST with the LOMO feature descriptor.

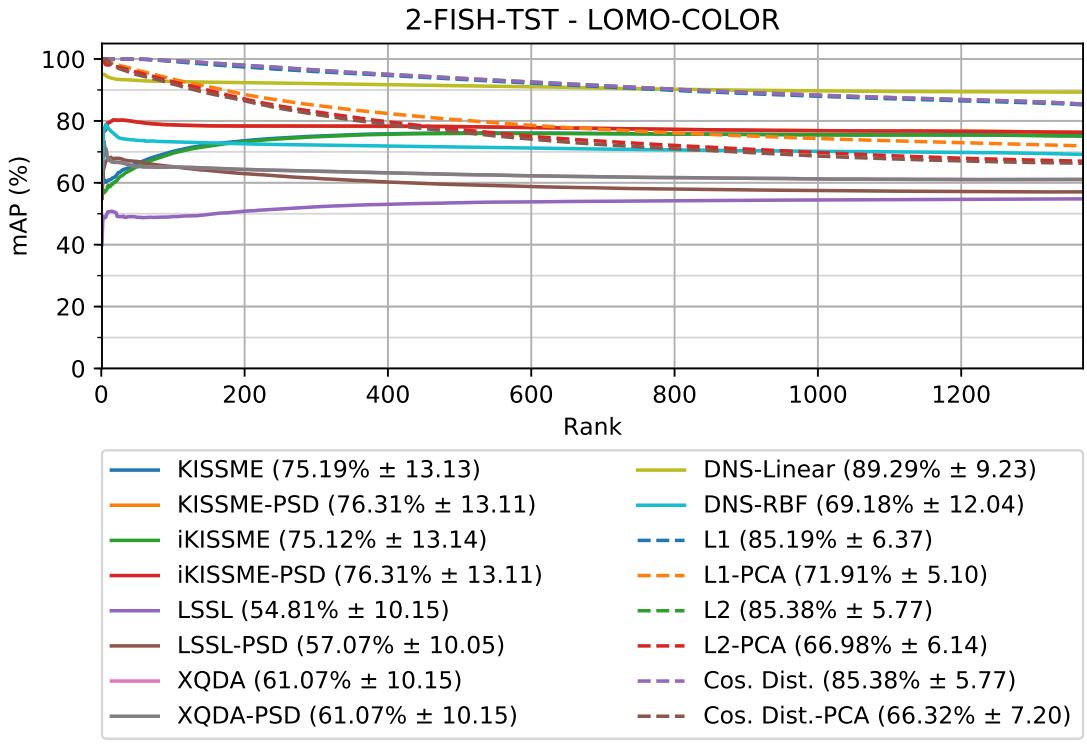


Figure A.8: mAP results for 2-FISH-TST with the LOMO-COLOR feature descriptor.

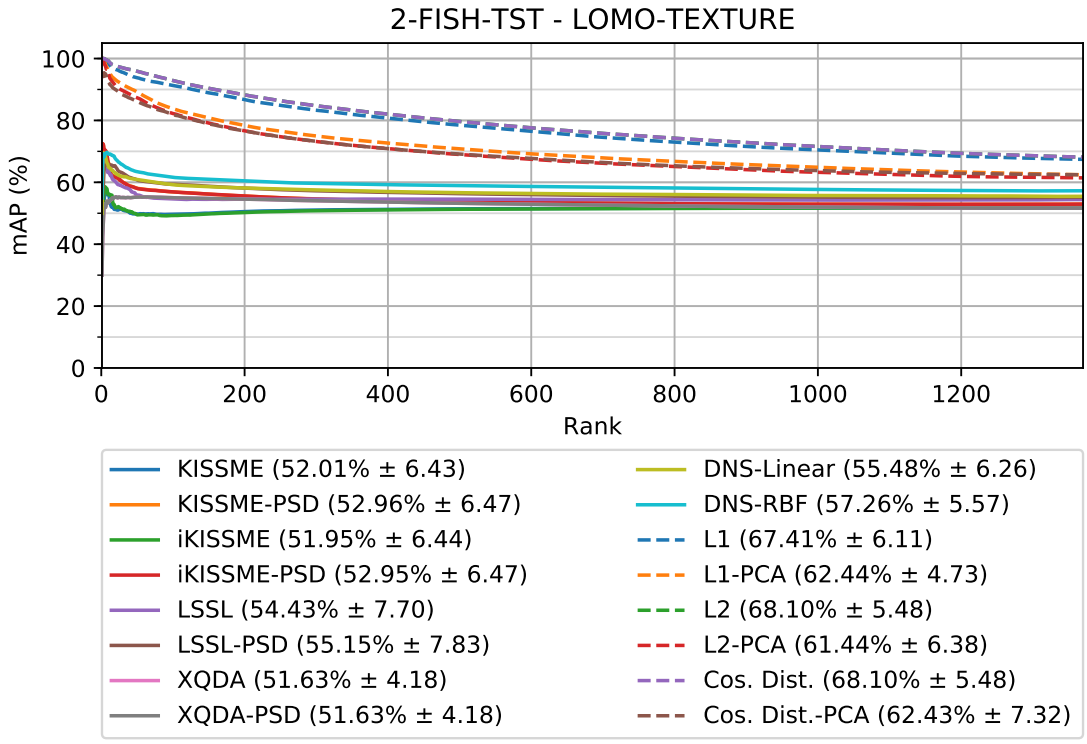


Figure A.9: mAP results for 2-FISH-TST with the LOMO-TEXTURE feature descriptor.

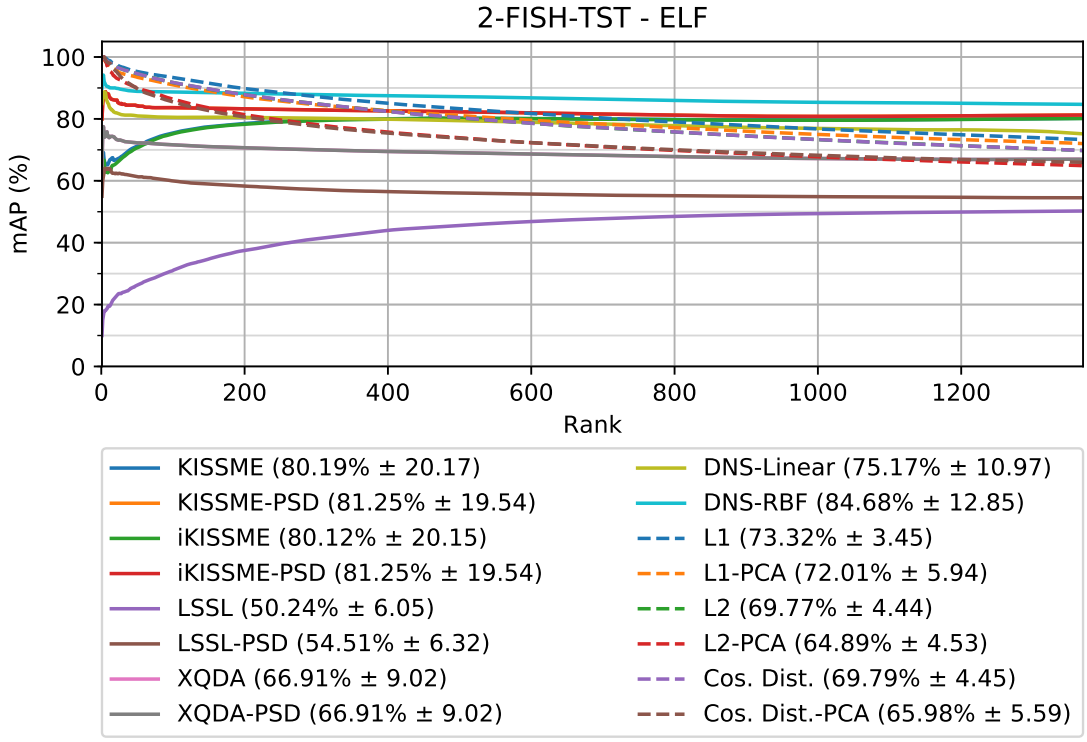


Figure A.10: mAP results for 2-FISH-TST with the ELF feature descriptor.

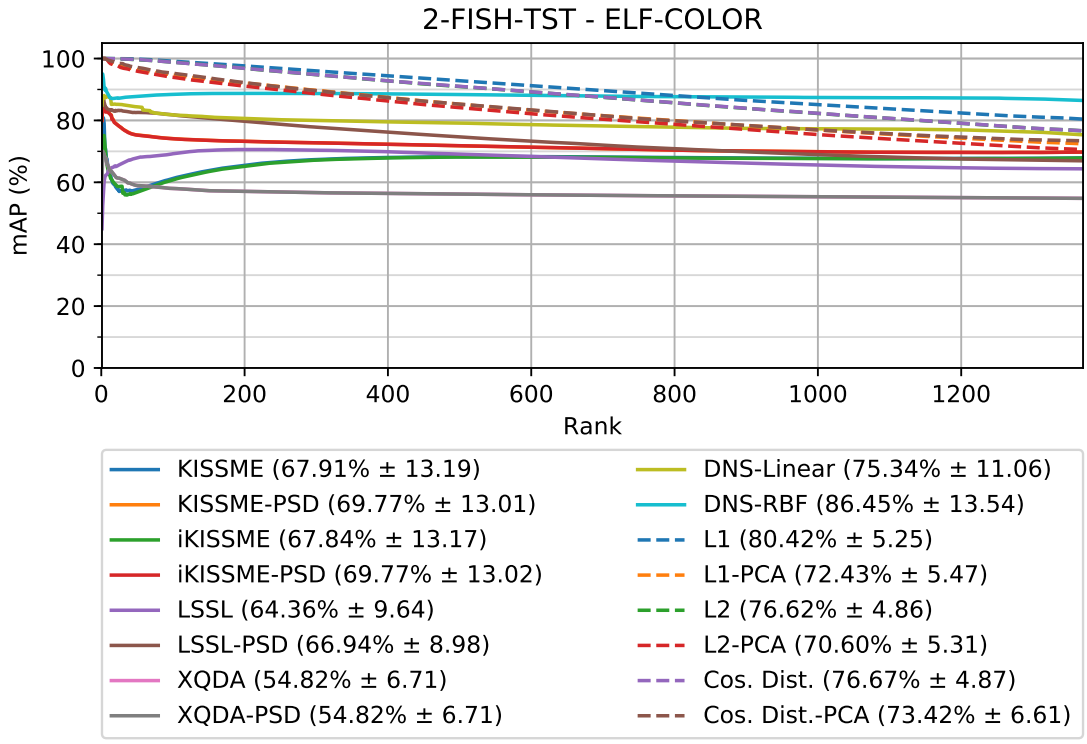


Figure A.11: mAP results for 2-FISH-TST with the ELF-COLOR feature descriptor.

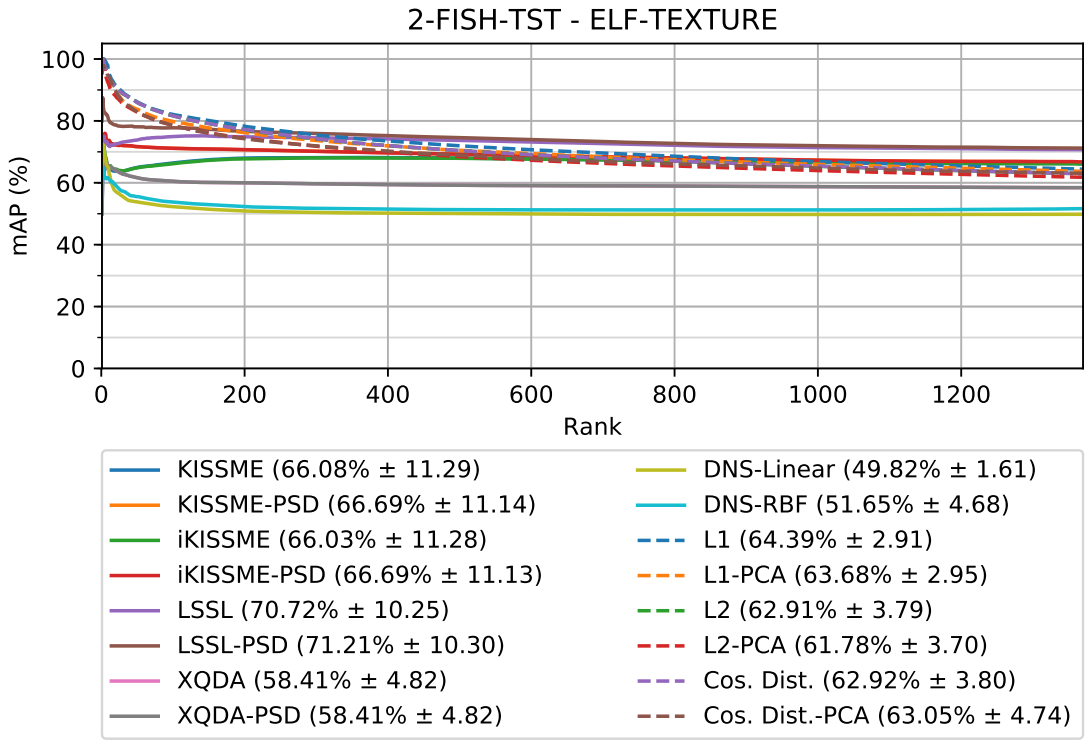


Figure A.12: mAP results for 2-FISH-TST with the ELF-TEXTURE feature descriptor.

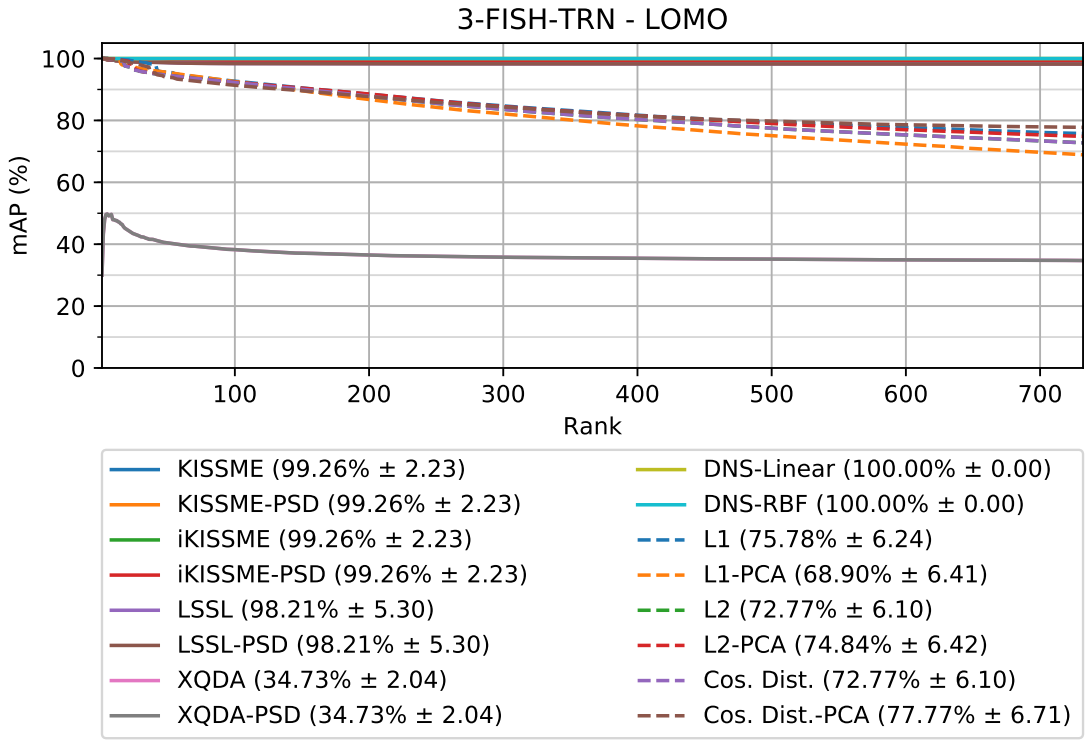


Figure A.13: mAP results for 3-FISH-TRN with the LOMO feature descriptor.

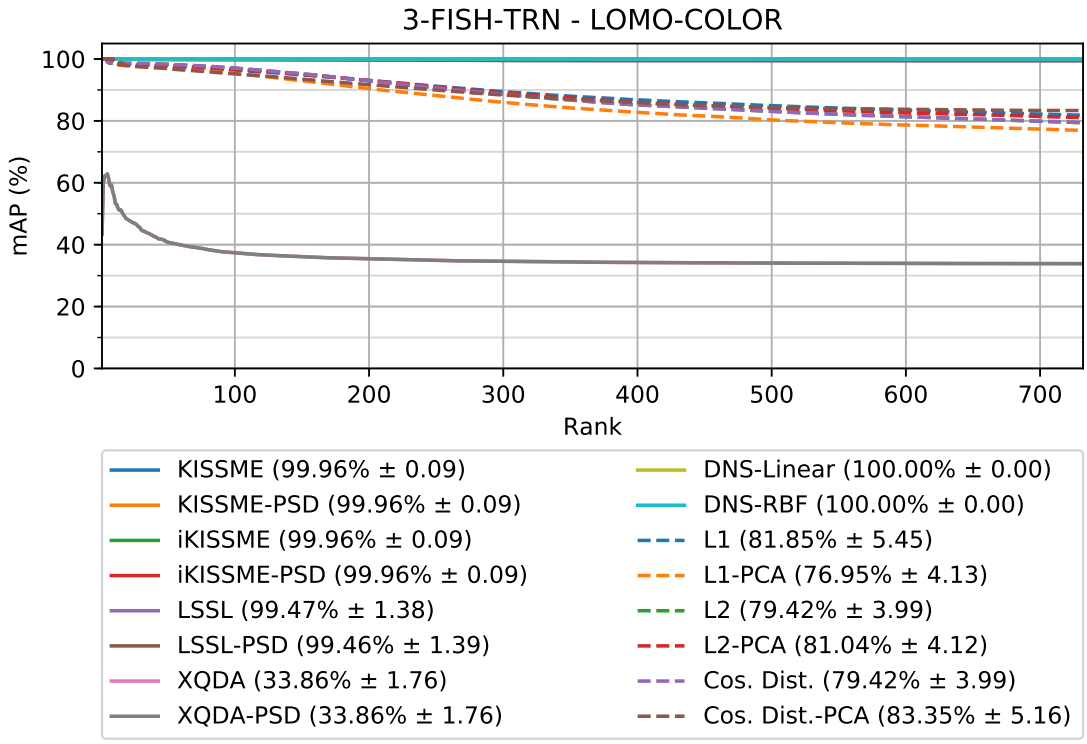


Figure A.14: mAP results for 3-FISH-TRN with the LOMO-COLOR feature descriptor.

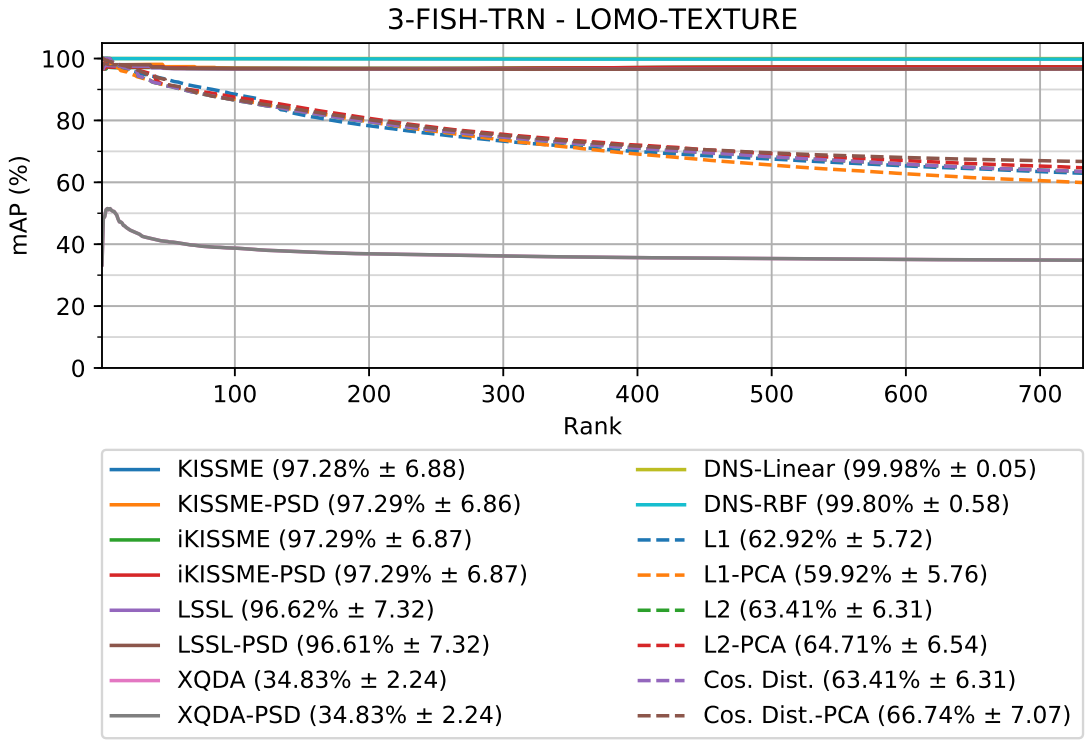


Figure A.15: mAP results for 3-FISH-TRN with the LOMO-TEXTURE feature descriptor.

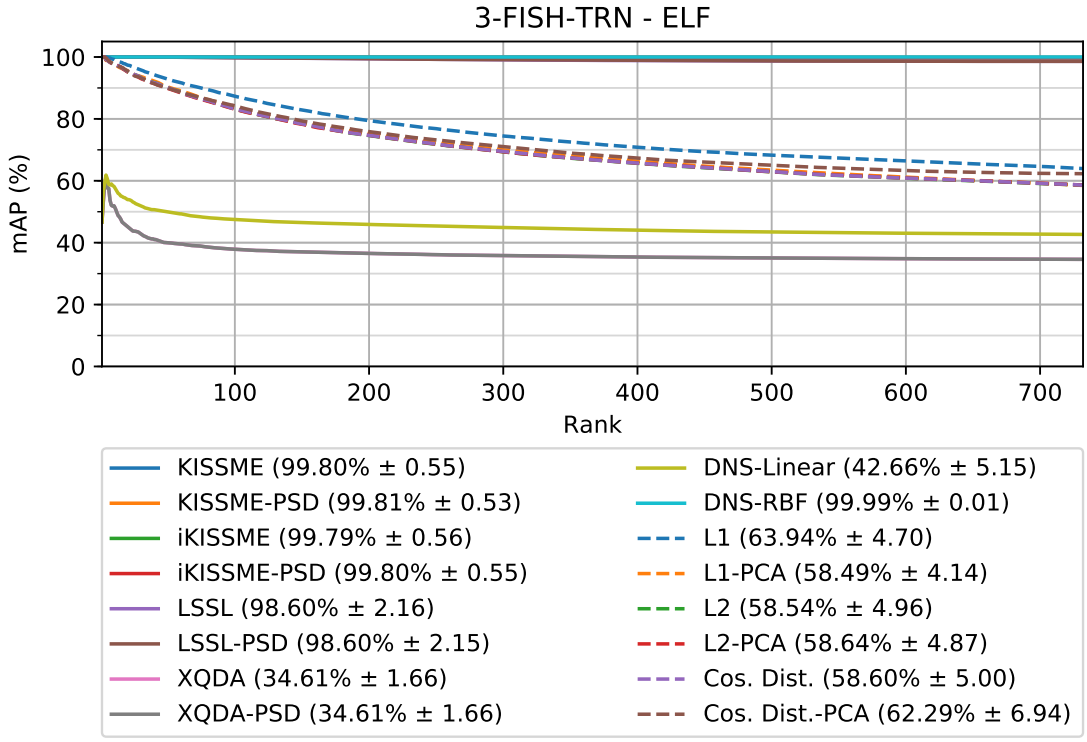


Figure A.16: mAP results for 3-FISH-TRN with the ELF feature descriptor.

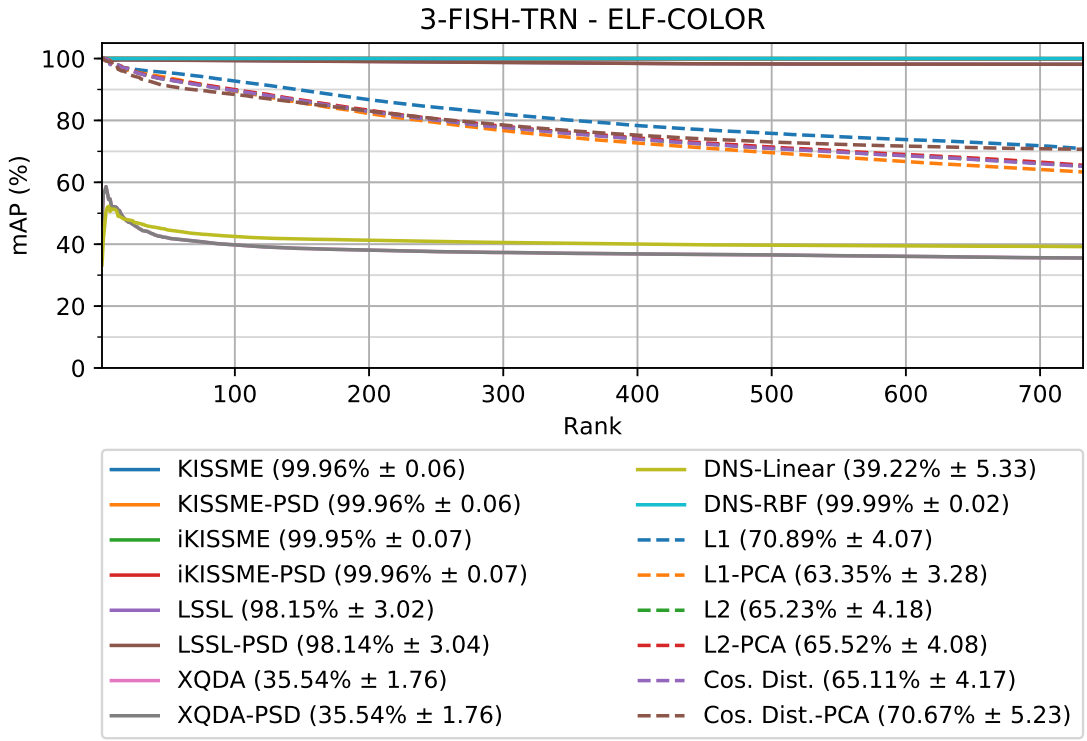


Figure A.17: mAP results for 3-FISH-TRN with the ELF-COLOR feature descriptor.

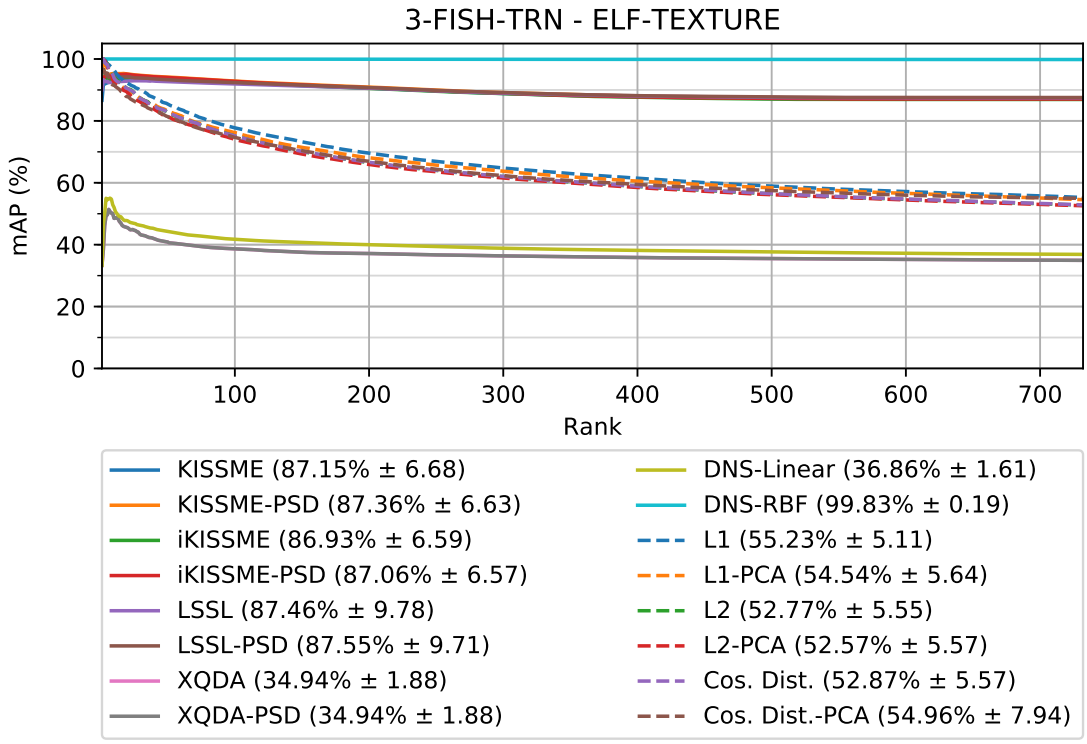


Figure A.18: mAP results for 3-FISH-TRN with the ELF-TEXTURE feature descriptor.

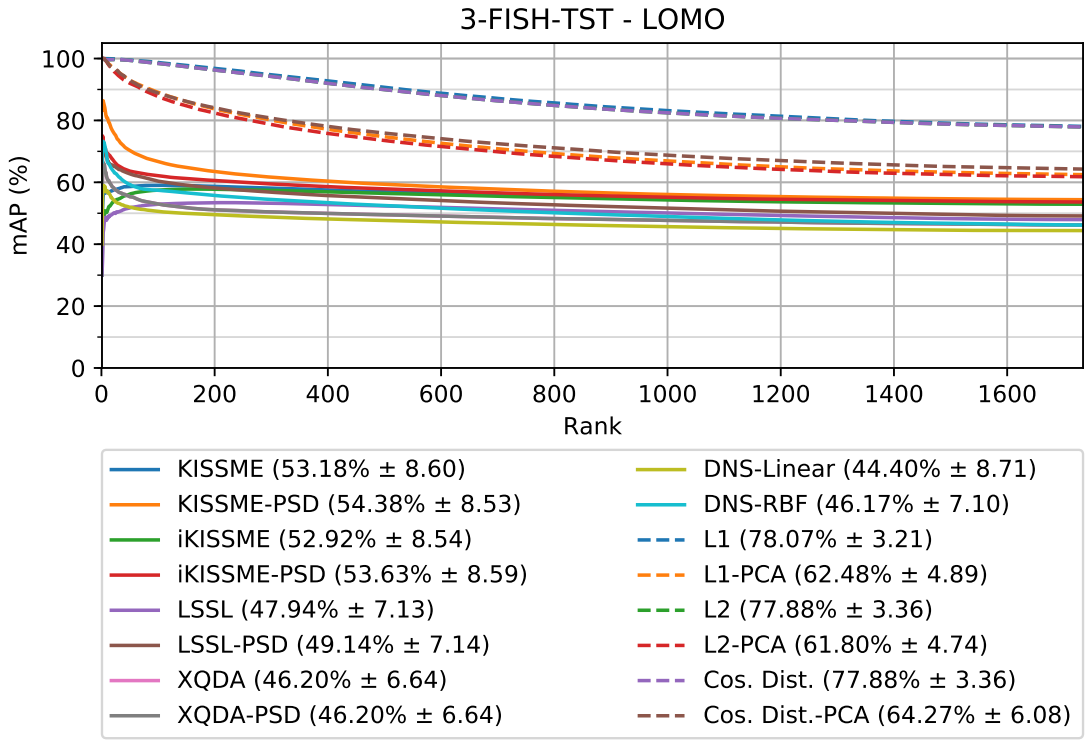


Figure A.19: mAP results for 3-FISH-TST with the LOMO feature descriptor.

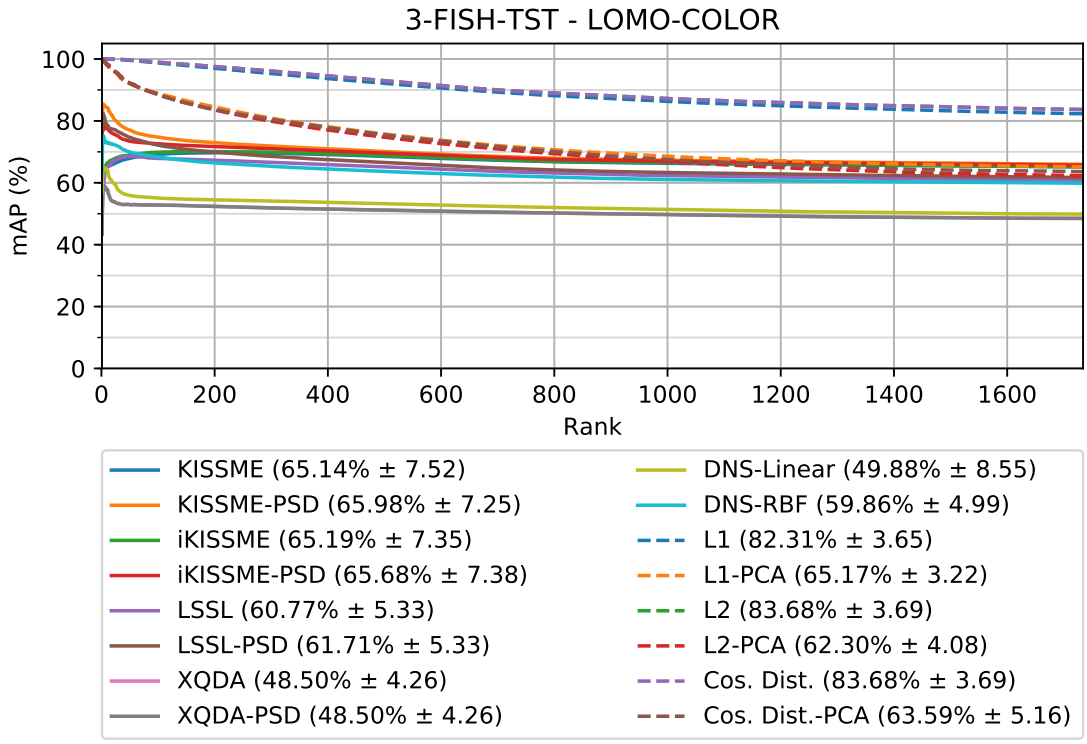


Figure A.20: mAP results for 3-FISH-TST with the LOMO-COLOR feature descriptor.

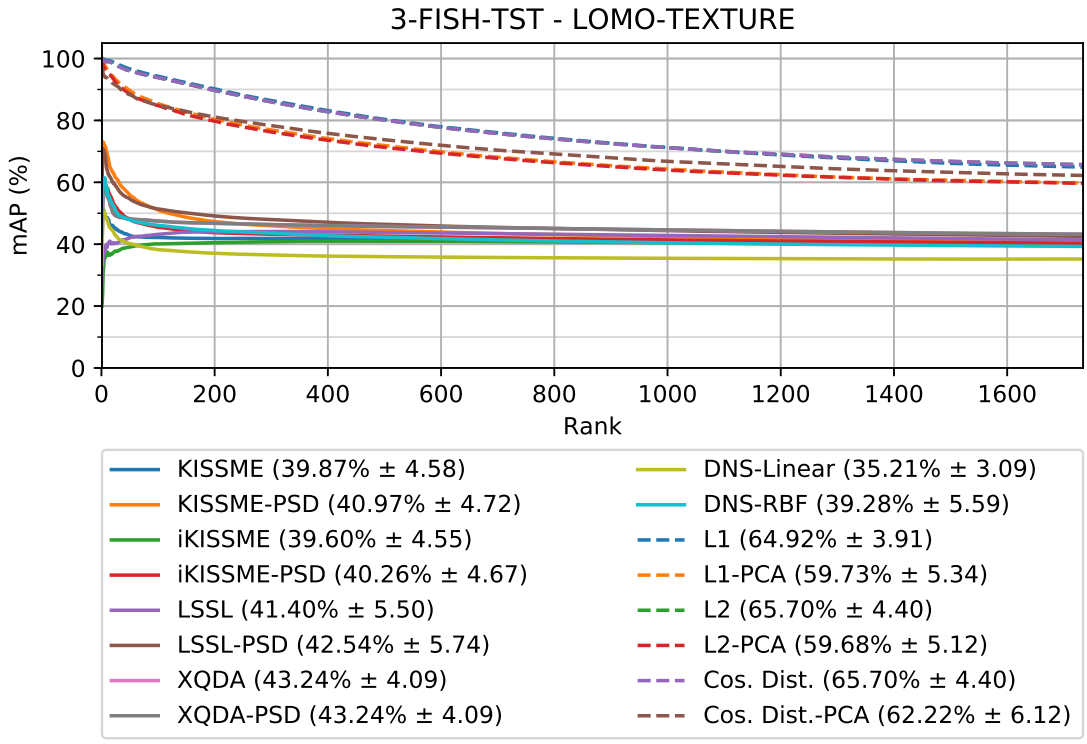


Figure A.21: mAP results for 3-FISH-TST with the LOMO-TEXTURE feature descriptor.

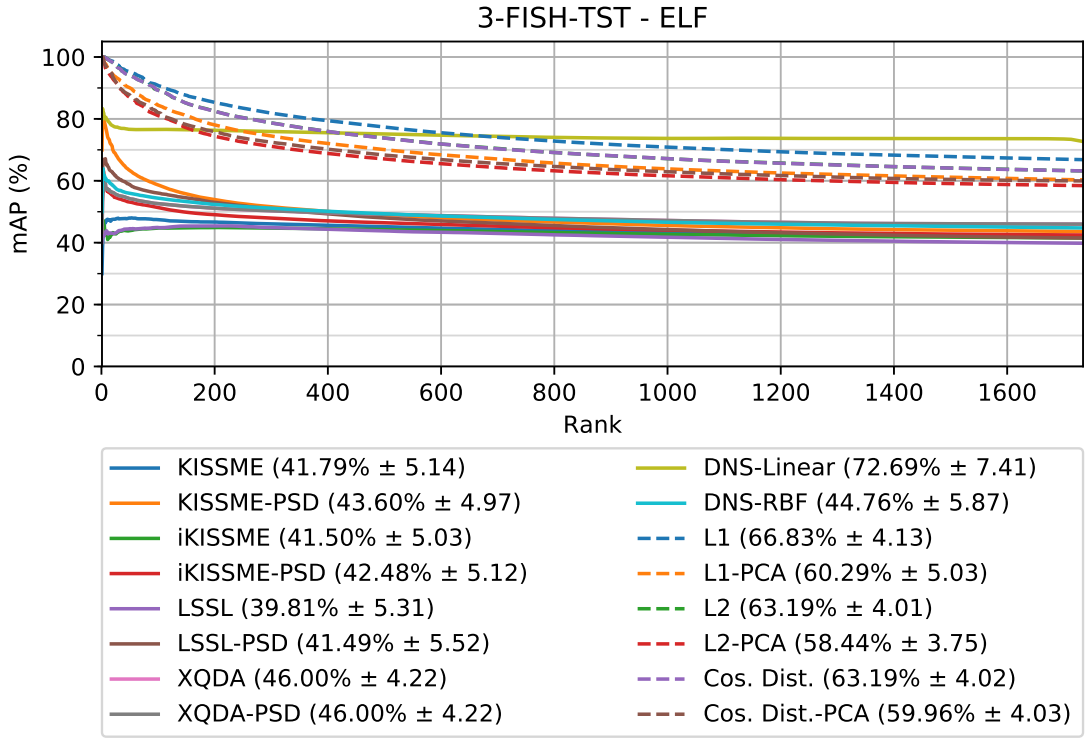


Figure A.22: mAP results for 3-FISH-TST with the ELF feature descriptor.

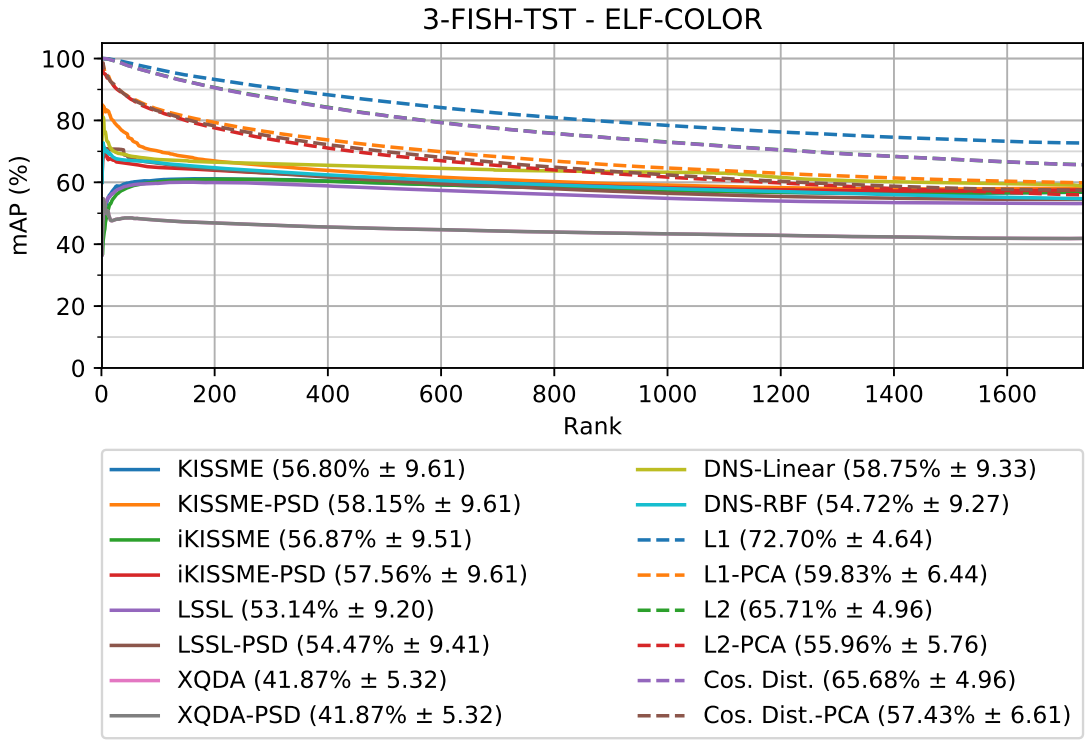


Figure A.23: mAP results for 3-FISH-TST with the ELF-COLOR feature descriptor.

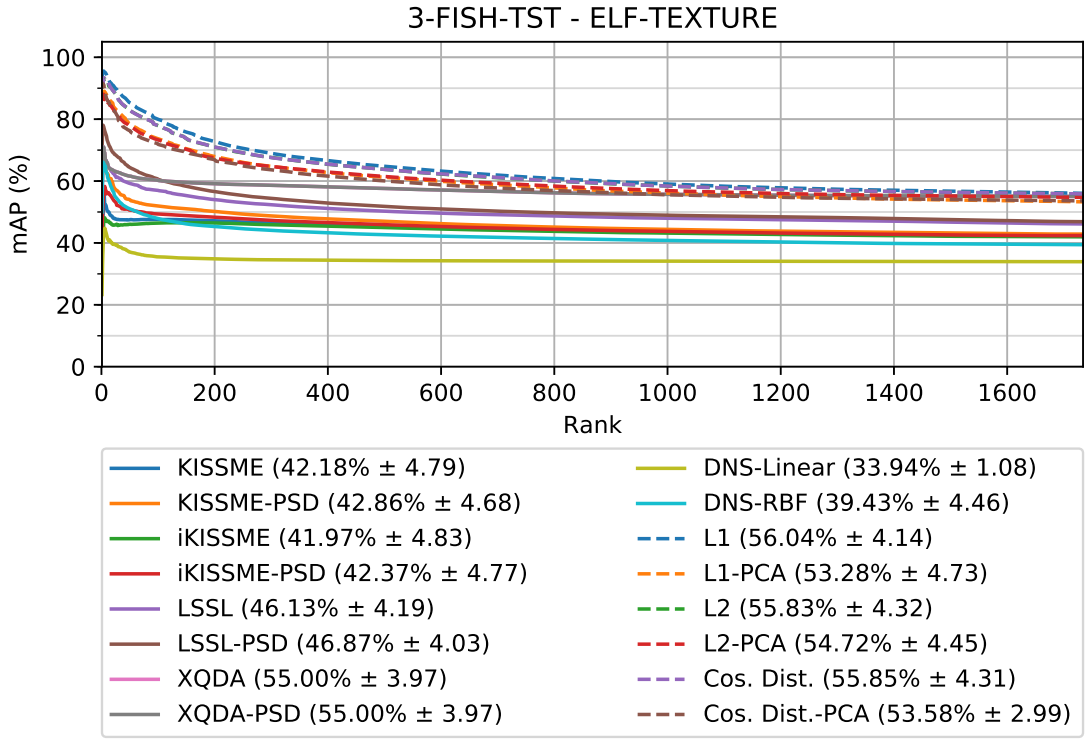


Figure A.24: mAP results for 3-FISH-TST with the ELF-TEXTURE feature descriptor.

B | Experiment Results - Sequence Association Test

In this appendix the results of the different feature representation used in the sequence test, that did not perform the best are shown. The results are in general quite similar to the results in *Section 6.4*, but with slightly worse results for DNS-Linear with the ELF-COLOR feature descriptor.

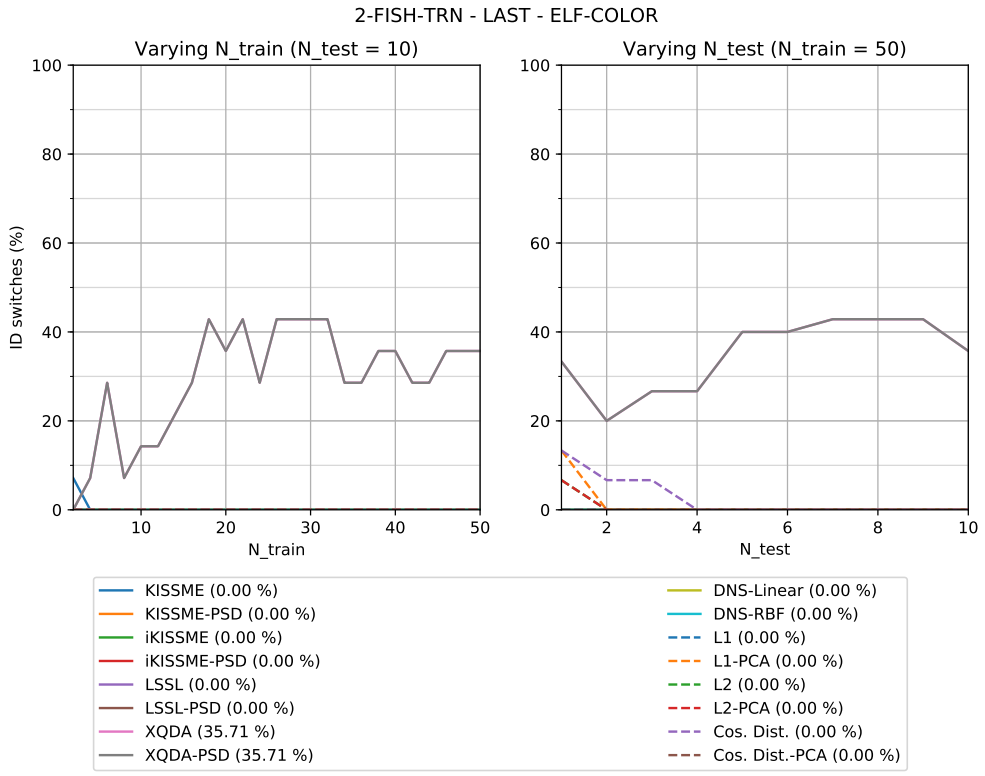


Figure B.1: Sequence results for 2-FISH-TRN with the ELF-COLOR and "Last Frame" feature representation

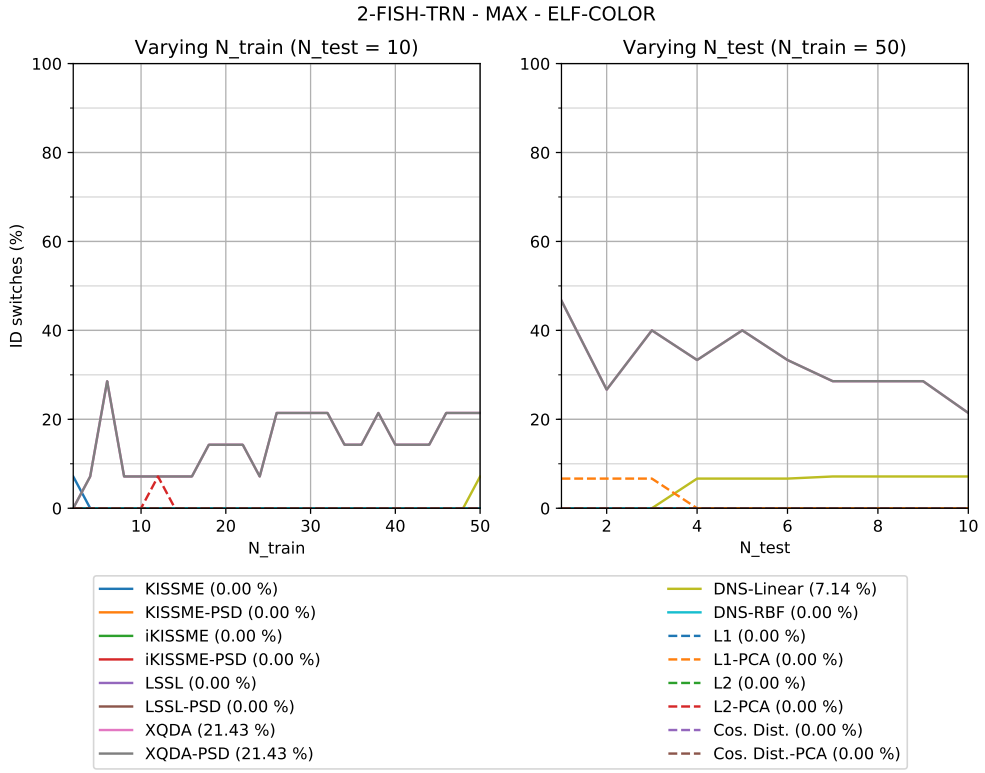


Figure B.2: Sequence results for 2-FISH-TRN with the ELF-COLOR and "Max" feature representation

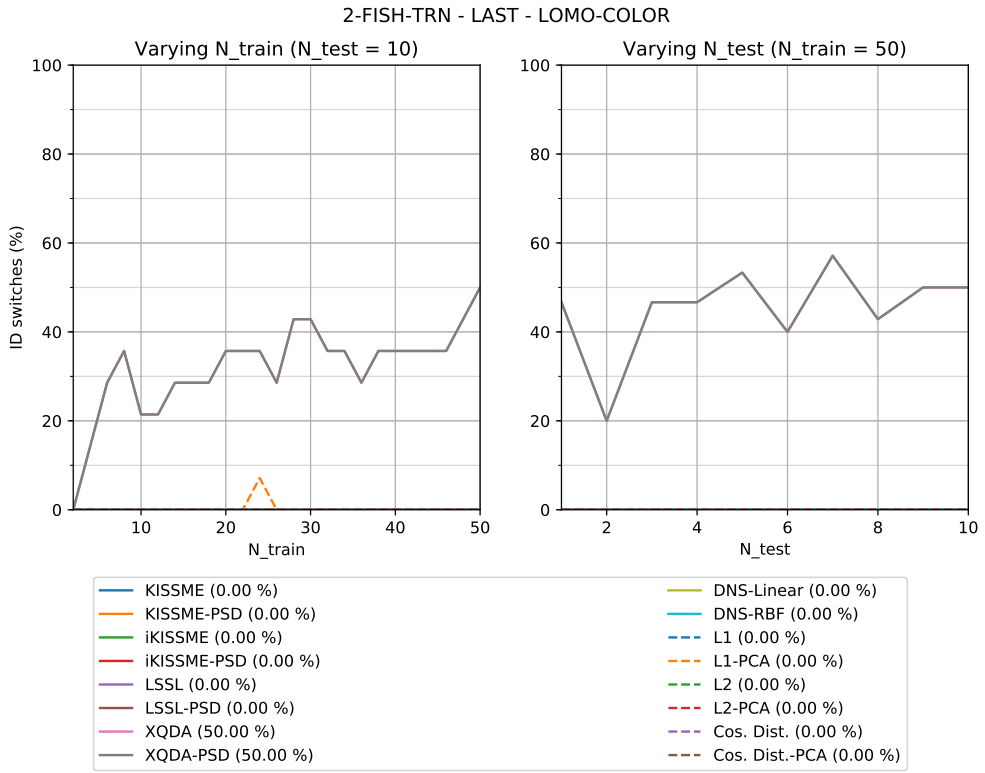


Figure B.3: Sequence results for 2-FISH-TRN with the LOMO-COLOR and "Last Frame" feature representation

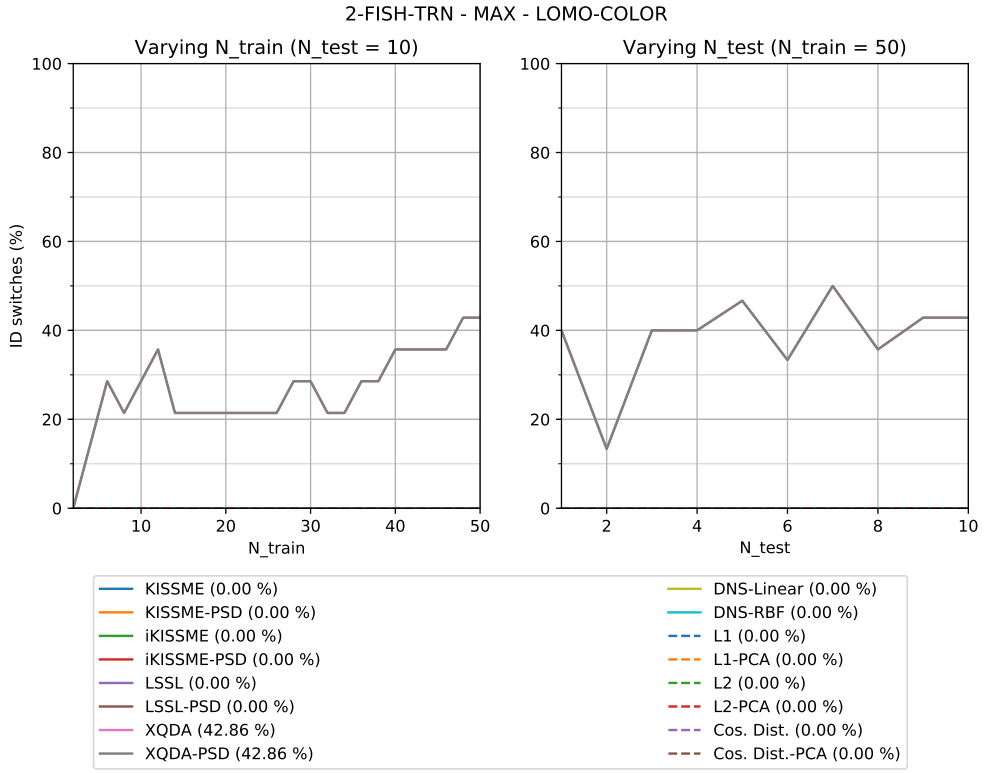


Figure B.4: Sequence results for 2-FISH-TRN with the LOMO-COLOR and "Max" feature representation

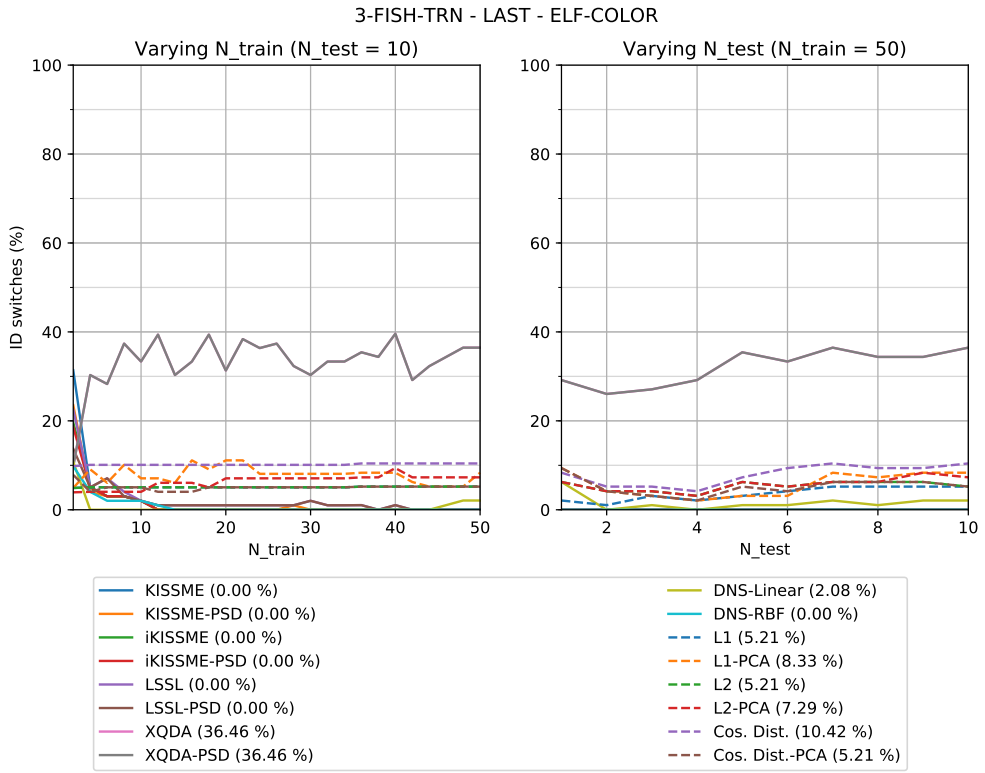


Figure B.5: Sequence results for 3-FISH-TRN with the ELF-COLOR and "Last Frame" feature representation

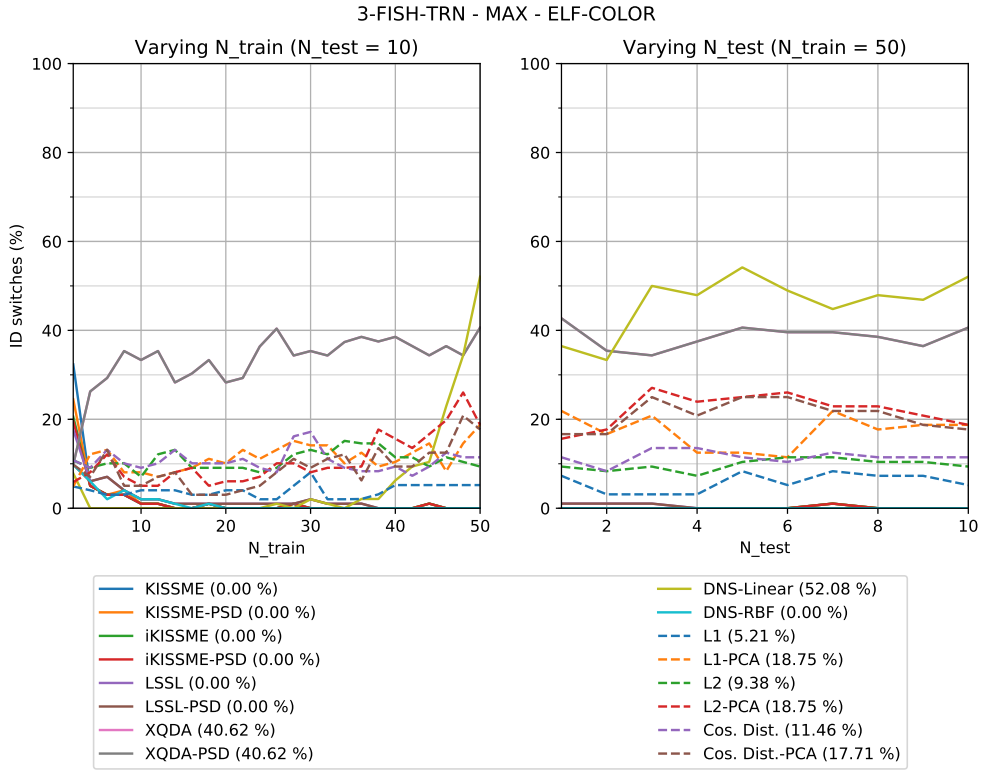


Figure B.6: Sequence results for 3-FISH-TRN with the ELF-COLOR and "Max" feature representation

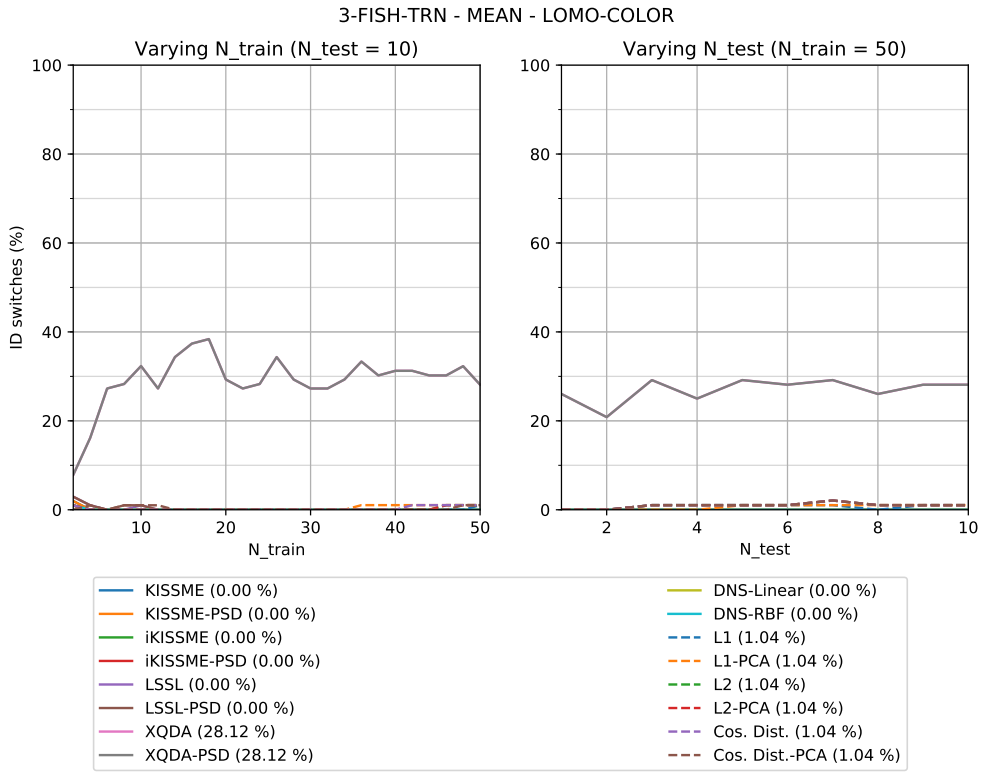


Figure B.7: Sequence results for 3-FISH-TRN with the LOMO-COLOR and "Mean" feature representation

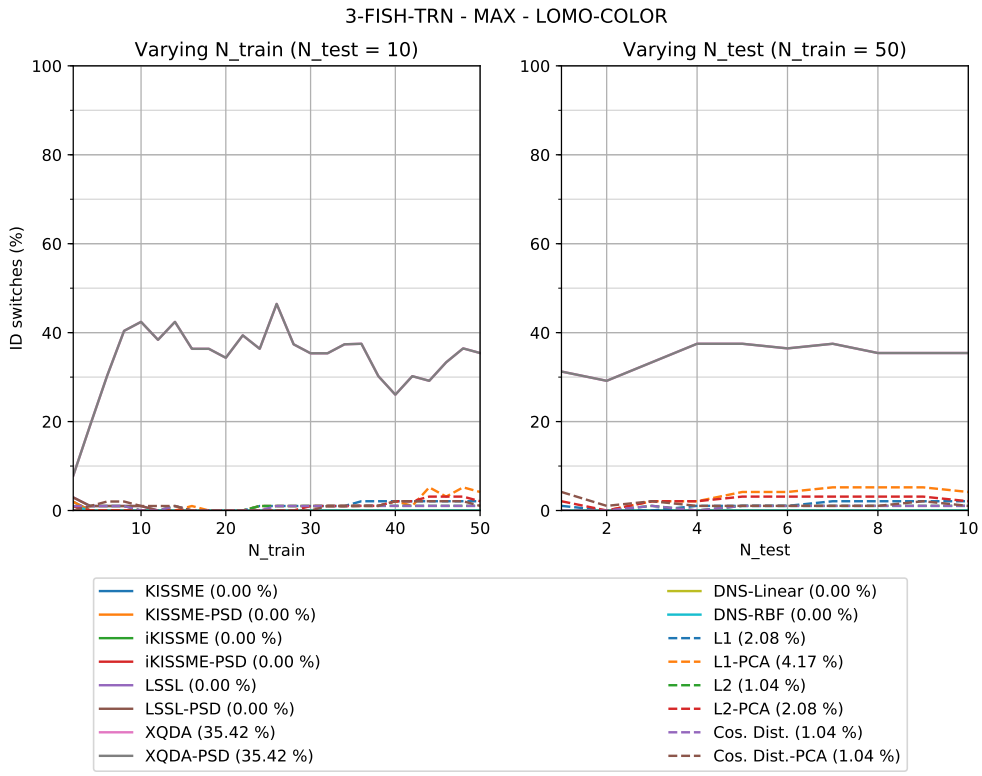


Figure B.8: Sequence results for 3-FISH-TRN with the LOMO-COLOR and "Max" feature representation