國立臺灣大學電機資訊學院電信工程學研究所

碩士論文

Graduate Institute of Communication Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

基於深度學習的物件追蹤算法與海豚偵測與識別

Deep Learning Based Algorithms for Object Tracking and Dolphin Detection and Identification

許宏瑋

Hung-Wei Hsu

指導教授：丁建均 教授

Advisor: Prof. Jian-Jiun Ding

中華民國 107 年 6 月

June, 2018

doi:10.6342/NTU201801036

國立臺灣大學（碩）博士學位論文
口試委員會審定書

基於深度學習的物件追蹤算法與海豚偵測與識別

Deep Learning Based Algorithms for Object Tracking and Dolphin Detection and Identification

本論文係 許宏瑋 君（R05942039）在國立臺灣大學電信工程學研究所完成之碩（博）士學位論文，於民國 107 年 06 月 19 日承下列考試委員審查通過及口試及格，特此證明

口試委員： 丁建均 （簽名）

（指導教授）

王鵬華

[illegible]

[illegible]

所　　長 吳宗霖 （簽名）

誌謝

首先謝謝媽媽、爸爸、哥哥，謝謝你們一直以來的鼓勵支持打氣敦促，你們是最棒的家人，謝謝你們。

感謝丁教授，不論在研究上、學業上、個性上或生活上的指導和溝通，感謝老師讓我有機會在碩士期間接下兩個計畫、出國和參加研討會，這些經驗讓我接觸不同文化和不同思考，開闊豐富自己的眼界，在心態上能夠更加包容，並在研究上有更多觀點和思維。

感謝已畢業的學長姊和實驗室的各位在研究、生涯上的建議和交流，感謝 Qualcomm 計畫中的郭博士在碩一時的指導，感謝耀仁學長在計劃時的建議，感謝奕承學長、周蓮香教授、侯雯學姊和海豚實驗室在海豚計畫上的共事和指導，也感謝奕承學長在生活、生涯、職場各方面的交流和建議。

感謝在 HTC DeepQ 部門實習期間遇到的各位和 mentors，在這段實習期間學習不論在研究、業界經驗都學習到許多。感謝佛學數位圖書館，感謝丁培峰先生，讓我在碩一兼職期間學習到有關資料庫和網站全端的各種知識和經驗。

感謝室友陳建欣、許祥奕、高偉哲、林清修，讓我有豐富的宿舍生活，見識各個領域不同的知識，和做研究的方法。

感謝肇輝，和你參加各式各樣的社群活動，遇見各式各樣值得學習的人，擴大自我的眼界，一同努力成長。

感謝榕潔，帶領我深入認識自己，認識這個世界，擁有突破自我的勇氣，理解和包容差異。

最後，感謝臺灣這片土地和土地之上豐富多元的社會及多采多姿的人們。

doi:10.6342/NTU201801036

摘要

本論文包含兩部分，第一部分有關 class-agnostic tracking。本論文提出兩個演算法，希望透過深度學習對於時序和影像資料的豐富特徵，解決目前 class-agnostic tracking 領域的問題。第二部分是將電腦視覺技術應用在環境保育領域，針對海豚資料的偵測及辨識。

在 trakcing 上，本論文中提出 FasterMDNet 和 RDisp。FasterMDNet 將 MD-Net 中時間複雜度極高的 online training 以根據 RNN 為基礎的模型更新策略取代，並以重複的 online training 和 back-propagation through time (BPTT) 來訓練。Faster-MDNet 比起 MDNet，時間節省大約十倍左右。RDisp 將已訓練好的物件偵測模型，加上 ConvRNN，建立追蹤模型。此外，由於 BPTT 在訓練 RDisp 的缺陷，本論文提出以兩階段片段訓練取代 BPTT 來訓練 RDisp。RDisp 在 GPU 上的執行速度大約是 25 fps，並能夠克服多種常見的影像變化。

在海豚偵測與辨識上，主要的兩個演算法是 Faster-RCNN 和 DenseNet。此外，在海豚名稱辨識上，單純的海浪背景讓訓練好的模型無法著重在海豚本身的細節特徵上。本論文提出結合基於深度學習和基於規則的 saliency 領域的演算法，來偵測海豚的區域，並把海面部分刪除，除去海面的影響。

關鍵字：物件追蹤，卷積遞歸神經網路，物件偵測，圖像分類，細粒度圖像分類，DenseNet，媽祖魚。

doi:10.6342/NTU201801036

Abstract

The first topic is class-agnostic visual tracking. The proposed algorithms attempt to tackle this problem via strong representative ability of temporal-spatial information in deep-learning techniques. The second topic is dolphin detection and identification.

The FasterMDNet and the RDisp are proposed. The FasterMDNet replaces computation-costly online training by an RNN model adaptation and trains RNN model adaptation by repeated online training via back-propagation through time (BPTT). The temporal cost is reduced to around 10 times faster than MD-Net with little sacrificed on accuracy. The RDisp incorporates pretrained detection model with ConvRNN cells. A two-stage clip training is proposed to replace BPTT in training to solve some defects of BPTT. The RDisp runs at 25 frames per second with consistency under multiple circumstances.

In the dolphin detection and identification, the trunks are the Faster-RCNN and the Densenet. In classification of dolphin names, interference of sea surfaces distracts the model from details of dolphins. The ensemble of deep-learning based and rule-based saliency detection algorithms with a soft gaussian threshold is proposed to create the dolphin mask to remove the pixels of sea surfaces.

Keywords: visual tracking, convolutional recurrent neural network, detection, classification, Taiwanese Humpback Dolphin, DenseNet, saliency.

doi:10.6342/NTU201801036

Contents

doi:10.6342/NTU201801036

doi:10.6342/NTU201801036

doi:10.6342/NTU201801036

List of Figures

doi:10.6342/NTU201801036

doi:10.6342/NTU201801036

List of Tables

doi:10.6342/NTU201801036

Chapter 1 Introduction

Two topics are included in this thesis. The first topic is about class-agnostic visual tracking. The proposed algorithms attempt to tackle problems in visual tracking via strong representative ability of temporal-spatial information in deep-learning techniques. The second topic is an integrated application on dolphin detection, segmentation, and recognition for marine conservation purposes.

Class-agnostic visual tracking is a fundamental and challenging topic in computer vision. First, the class-agnostic property leads to that class visual priors cannot be learned from the training dataset. This increases the difficulty and the system should adopt a versatile, complete, and robust visual features to represent all possible tracked objects and, in the meantime, reject all possible backgrounds. Second, there are a variety of changes for the tracked object among different video frames, including object motion, camera motion, occlusion, object deformation, illumination change, etc. Moreover, these changes are unpredictable in the long term. Therefore, the class-agnostic and long-term-change-agnostic properties of visual tracking magnify the difficulty in extracting or learning robust spatial-temporal features. In addition, the temporal cost is a vital component in the performance of a tracking algorithm. Hence, a good tracking algorithm should perform well on both accuracy and time-complexity. In the field of class-agnostic visual tracking, the two-stage system including a feature extraction block and a feature processing block is still the mainstream. Well-known deep-learning based tracking models, including the MDNet [1], the

doi:10.6342/NTU201801036

TCNN [2], etc, utilizes online training to process temporal changes which has a high computation cost. Other algorithms benefit from the versatile visual features extracted from the pre-trained CNN in different tasks.

Two tracking algorithms are proposed: the FasterMDNet and the RDisp. The proposed algorithms take the MDNet as the baseline. The FasterMDNet is trying to replace the high-computation-cost of the online training model by an RNN model adaptation. The simple idea behind is that the process of updating the model based on the current model, previous input frames, and predictions in repeated online training can be represented by a learnable function modeled as the RNN. The FasterMDNet is based on this concept. It replaces the online training with an RNN model and trains the RNN by repeated online training. The RDisp is based on the philosophy behind the first work but utilizes a more complicated model and state-of-the-art structures of object detection such as region proposal network, pre-trained one-stage detection model, and convolutional recurrent neural network. In addition, a two-stage clip training is proposed to replace back-propagation in training the RDisp due to some defects in back-propagation leading to a failure in training.

The second part of this thesis is about dolphin detection, dolphin identification, and dolphin image foreground extraction. Different from well-known detection and semantic segmentation datasets like VOC [3], COCO [4], etc where complete annotations are available, our dolphin dataset contains only parts of annotations, which makes the work even challenging.

Here, a system to process our dolphin dataset including detection, foreground extraction, and identification is proposed. First, to train the system of detection and recognition, the Faster-RCNN [5] is utilized. The Faster-RCNN works well on detection under large number annotations and satisfactorily on identification in the beginning. Yet, when the same model is tested on recently taken images, the results demonstrates a much poorer performance than the performance on the testing dataset. With the analysis of the model

doi:10.6342/NTU201801036

and the split of the dataset using gradient-based saliency maps [6], the results shows the sea surface background distracts the model. A further experiment on dataset split by dates approves the argument. Therefore, a saliency block is inserted into the pipeline and the system turns to 3 blocks: the detection block, the saliency block, and the classification block.

In the detection block, the base algorithm is the Faster-RCNN. Several preprocessing methods are utilized to enhance the dolphin information and remove irrelevant noisy background. In the saliency block, due to the lack of segmentation annotations, it is implausible to train a data-driven model based on dolphin mask groundtruths. Due to the simple background of the sea, we utilize saliency detection algorithms to segment the dolphin parts. Yet, in direct use of algorithms such as [7, 8, 9, 10] on the cropped patch, problems arise that the saliency lie on only parts of the dolphin since the whole dolphin accounts for most pixels in the cropped patch. A solution is to use twice the bounding box for crop and the results demonstrate qualitative results. In addition, an ensemble on four utilized algorithms is induced to increase the robustness of the produced saliency maps. In the classification block, the DenseNet121 [11] is utilized on the masked dolphin images for classification of names and on the original dolphin images for classification of angles and stages. In classification of stages, histogram of pixels and histograms of color channels as features and neural networks as classifiers are also attempted. In classification, the DenseNet121 demonstrates the highest accuracy among classification of three annotation types. Finally, an desktop application of the proposed system is built upon Python Tkinter.

doi:10.6342/NTU201801036

Chapter 2 Reviews of Tracking Techniques

In this chapter, basic knowledge of visual tracking is introduced. This chapter is organized as follows. Section 2.1 introduces background knowledge of visual tracking. Section 2.2 introduces the model formulation of visual tracking. Section 2.2.1 and 2.2.2 includes different settings of visual tracking. Section 2.3 introduces the specific tracking setting, class-agnostic RGB tracking, in this thesis. Section 2.4 introduces three well-known class-agnostic RGB tracking datasets.

2.1 Introduction

Visual Tracking is a fundamental, significant and challenging task in the field of computer vision. When human observes the world, certain objects are focused by human eyes and being kept in the line of sight. Tracking is similar to this process to focus on a certain object. In technologies nowadays, cameras act as human eyes for the machines and capture information as videos composed of frames sampled at a fixed time rate. In standard video analysis [12]. There are three primary steps: detection of interesting objects, tracking of these objects, and analysis of object tracks for higher semantic informations. The tracking algorithm correlates interesting objects in different frames for temporal consistency and combines informations in 2D static image domain into 3D temporal-spatio domain. Thus, tracking algorithms are pertinent in the video-related tasks [12].

The task of visual tracking can be basically interpreted as following steps: choosing

doi:10.6342/NTU201801036

the tracked target in the beginning and determining the location and size of the target in the following consecutive video frames. Visual tracking plays an important role in applications like robotic vision, surveillance, military monitoring, etc. In the applications of visual tracking, the trajectory, the appearance change or the motion of the target can be further analyzed to extract in-depth knowledge of the video.

2.2 Problem Formulation

In the scenario of object tracking, the interesting objects and the video frames are assumed to be given. The goal of object tracking is to search the location of the interesting objects in videos frames. Normally, the locations of the interesting objects are given for the first frame or the first few frames, and tracking algorithms will produce a location of each interesting object in each following frame.

Mathematically, suppose there are a set of videos frames $F = \{f_1, f_2, ..., f_N\}$ and a set of interesting objects $X = \{x_1, x_2, ..., x_K\}$ with ground-truth locations $Y = \{Y_1, Y_2, ..., Y_M\}$, $M < N$ where $Y_t = \{y_{0,t}, y_{1,t}, ..., y_{K,t}\}$ are the ground-truth locations of all interesting objects for frame f_t. The goal of a tracking algorithm is to predict the locations of interesting objects in frame $\{f_{M+1}, f_{M+2}, ..., f_N\}$. The predictions are notated as $\hat{Y} = \{\hat{Y}_{M+1}, \hat{Y}_{M+2}, ..., \hat{Y}_N\}$, where $\hat{Y}_t = \{\hat{y}_{0,t}, y_{1,t}, ..., y_{K,t}\}$ are the ground-truth locations of all interesting objects for frame f_i. Thus, a tracking algorithm can be generally modelled as a function g such that

$$\hat{Y} = g(F, X, Y). \tag{2.1}$$

Assume that interesting objects are independent. The problem can be simplified into the scenario of a single tracked object. The problem formulation is simplified as that given input video frames $F = \{f_1, f_2, ..., f_N\}$ and a interesting object x with ground-truth locations $\{y_1, y_2, ..., y_M\}$ in frame $\{f_1, f_2, ..., f_M\}$, a tracking algorithm searches

doi:10.6342/NTU201801036

predictions $\hat{Y} = \{\hat{y}_{M+1}, \hat{y}_{M+2}, ..., \hat{y}_N\}$ in frame$\{f_{M+1}, f_{M+2}, ..., f_{M+K}\}$. Equation 2.1 is simplified as,

$$\{\hat{y}_{M+1}, \hat{y}_{M+2}, ..., \hat{y}_N\} = g(f_1, f_2, ..., f_N, x, y_1, y_2, ..., y_M). \tag{2.2}$$

In addition, most applications require a real-time tracking system for shorter latency in relative response. For example, a short latency is required for a surveillance system when a suspicious person or a suspicious activity is captured to faster response to potential dangers. Hence, **a tracking algorithm is normally assumed to be causal** since a requirement for future frames will fundamentally increase the latency. Based on this, equation 2.2 can be simplified as,

$$\hat{y}_t = g(f_1, f_2, ..., f_M, ..., f_t, x, y_1, y_2, ..., y_M), M < t \leq N. \tag{2.3}$$

In most scenarios, the computational cost of processing all previous frames and the current frame in every time-step is very high. Most algorithms will store the information extracted from ground-truths, previous frames, and previous predictions in a set of information represented in different forms. The information can be previous t frames, object candidates, state variables, appearance models, etc. In the beginning of tracking, the initial information is extracted from initial frames and ground-truths. In the following time-step, the algorithm includes two steps: prediction based on previous information and current frame, and information update. Suppose the information from $\{y_1, y_2, ..., y_M\}$ and $\{f_1, f_2, ..., f_M, ..., f_t\}$ is notated as S_t. The algorithm formulation can be represented as

doi:10.6342/NTU201801036

follows,

$$S_M = o(f_1, f_2, ..., f_M, x, y_1, y_2, ..., y_M) \qquad \text{(initial information).} \tag{2.4}$$

$$\hat{y}_t = g(f_t, S_{t-1}) \qquad \text{(prediction).} \tag{2.5}$$

$$S_t = h(f_t, S_{t-1}) \qquad \text{(update).} \tag{2.6}$$

In equation 2.4, function $o(\cdot)$, function $g(\cdot)$, function $h(\cdot)$, and the representation of S are unknown. Thus, tracking algorithms differ from the following questions: how is the information S represented, how to extract initial information (function o), how to generate prediction (function g), and how to update information (function h).

2.2.1 Video Frame Channels: RGB, D, and RGB-D

For a video in our daily lives, the video frames are composed of color informations, or RGB channels, thanks to the cheap price and small volume of CMOS image sensors. In early years before 2010, due to the high price of RGB-D camera like the Swiss Ranger SR4000 and PMD Tech products which costed around USD$10000, it's rarely considered to use RGB-D channels in video applications. In November 2010, Microsoft introduced *Kinect for Xbox 360* at a price around USD$150 which provides color information and depth information for every pixel using structured light technique, which largely decreased the price of RGB-D sensor in order of magnitude and aroused the interests in utilization of RGB-D channels in different fields [13]. In Feburary 2016, the *LeapMotion* introduced Orion on their *Leap sensor* which is available to track every articulations of hand in first-person view using only depth information at a price at around USD$20 with a much smaller volume than *Kinect*.

Since depth information is illumination-invariant, the algorithm with the alleviation of depth can be more robust to changes in color domain. Due to affordable prices of RGB-D

doi:10.6342/NTU201801036

sensors and the illumination change immune property of depth information, utilization of depth information increased in computer vision fields including tracking. According to the channels of input video frames, tracking algorithms can be split into 3 genres: RGB tracking [14], Depth tracking [15, 16, 17, 18] and RGB-Depth tracking [19, 20, 21, 22, 23, 24, 25].

However, even though RGB-D cameras reach affordable prices after the release of *Kinect*, they are still hundred times of RGB cameras. Furthermore, RGB sensors are built-in devices in a wide range of commercial electronic products nowadays such as smart phones, notebooks. Hence, for applications toward public people, algorithms based on RGB channels require more inexpensive hardware and can reach more people compared with RGB-D cameras.

2.2.2 Class-Aware and Class-Agnostic Tracking

According the prior knowledge of the tracked objects, tracking can be split into two genres: class aware and class agnostic. If tracking algorithms track only interesting objects of specific classes, it's class-aware tracking since the algorithm is aware the class prior before tracking. For example, pedestrian tracking, face tracking, animal tracking, or football tracking are class-aware tracking. On the other hand, if tracked objects can be of any class, it's class-agnostic tracking. The tracked object in class-agnostic tracking are given by the location in the frame such as a bounding-box in the first frame of a video. Due to the absence of prior knowledge of tracked objects and potentially various classes of objects, class-agnostic tracking is much more challenging.

For simplification, tracking and visual tracking mentioned in following chapters refers to RGB class-agnostic tracking. Plus, existence of interesting objects inside video frames are assumed. The multi-object tracking requiring detection of appearance and disappear-

doi:10.6342/NTU201801036

ance of interesting objects [26, 27, 28, 29] is not in the range of this thesis.

2.3 Class-Agnostic RGB Tracking

Class-agnostic rgb tracking is a challenging task in computer vision due to versatile factors of changes of the aimed target and the quality of the video including illumination variations, scale changes, occlusions, motion blur, target deformations, background clutters, coexistence of similar objects, etc.

In the task of class-agnostic tracking, the target is provided by a bounding box in the first frame of the video and the tracker predicts the location and the size of the target in the following frames. The difficulties of visual tracking includes online target acquisition, limited time for adaptation, versatile video quality, etc. For example, in VOT 2016, each video was labeled with six different visual attributes: 1. occlusion, 2. illumination change, 3. motion change, 4. size change, 5. camera motion, and 6. unassigned.

To design a tracking algorithm, three major problems should be considered:

I. Target Acquisition

Target Acquisition means the tracking algorithm has to recognize the tracked target chosen by the user in order to track the same object in the following frames. Normally, the chosen target is provided as a rectangle or a quadrilateral enclosing the target in the first frame.

In KCF, the target is represented directly by RGB pixel values inside the rectangle. In C-COT, multiple features extracted from the patch including HoG, CNN features, etc are utilized. In MDNet and T-CNN, the target acquisition is done by training the convolutional neural network with multiple positive and negative patches, where whether a patch is positive or not is determined by IoU with the ground-truth rectangle.

doi:10.6342/NTU201801036

II. Target and Environment Change

Multiple changes of the target and environment will occur in different videos. The target changes includes (1) the relative motion of the target: changes of location or velocity of the target in consecutive video frames (2) change of the appearance: illumination change, size change, deformation like a person turning around or a diving person, color changes, etc. The environment changes can be more complicated like occlusion, motion of camera, coexistence of similar objects, etc. A robust tracking algorithm requires a good model update strategy to deal with versatile changes in a video.

III. Computation Cost

In addition, for a practical tracking algorithm, time resources are limited. For example, a surveillance system that detects potential dangerous activities requires a short latency for quick response. Hence, a good model update strategy and a good target acquisition strategy require small time complexity for real-time tracking.

2.4 Datasets

In this section, three notable datasets of class-agnostic rgb tracking are introduced. The qualities of sequences, annotation formats, annotations of visual attributes, evaluation metrics, and evaluation protocols are covered and summarized as a table for comparison.

2.4.1 Amsterdam Library of Oridnary Videos fo tracking (ALOV++)

ALOV++ [30] is a milestone dataset in the field of visual tracking which covers versatile scenarios with an enormous number of annotated videos.

For qualities of sequences, ALOV++ gathers **315** video sequences covering diverse circumstances involving illumination changes, occlusion, clutter, camera motion, low con-

doi:10.6342/NTU201801036

trast, specularities. Among 315 video fragments, 65 sequences are from PETS workshop [31], 250 sequences are newly collected from real-life videos on YouTube with 64 different types of targets. The sequences are divided into 13 aspects of difficulties. For the length of the videos, most videos are short videos with an average of 9.2 seconds and a maximum of 35 seconds. Ten videos have long durations between one and two minutes.

For annotation format, each video has a single target and is annotated by rectangular bounding-boxes along the main axes of flexible sizes every fifth frame. The ground truths of intermediate frames are acquired by interpolation. No visual attributes such as occlusion label are given for sequences or frames.

For evaluation protocol, ALOV++ utilizes one-pass evaluation (OPE). The first frame and the ground truth in the frist frame are given for initialization. Trackers produce bounding boxes for prediction in every following frames until the last frame of a video. When the target is lost by the tracker, no re-initialization is provided.

For evaluation metrics, ALOV++ summarizes metrics in previous literatures and presents seven metrics as Table 2.1. The notations follow [30]. Since ALOV++ is not adopted in the proposed algorithms in this thesis due to lack of evaluation toolkit, lack of re-initialization mechanism, and inconsideration of robustness, the details remain for interested readers to refer to section 2.3 in the original paper [30].

2.4.2 Object Tracking Benchmarks (OTB50/OTB100)

In CVPR 2013, OTB [32] is released with 50 fully annotated sequences. In 2015, OTB [33] is augmented with additional 50 sequences and summed as 100 sequences in total. In literatures, the dataset released in 2013 is called OTB50 or OTB2013 and the one released in 2015 is notated as OTB100 or OTB2015.

For qualities of sequences, the lengths of videos range from short term with dozens of

doi:10.6342/NTU201801036

Table 2.1: Evaluation Metrics of ALOV++ [30].

Name	Equation	Aim	Measure
F-score	$2 \cdot \frac{precision \cdot recall}{precision + recall}$	Accuracy	Threholded precision and recall
$F1$-score	$\frac{1}{N_{frames}} \sum_i 2 \cdot \frac{p^i \cdot r^i}{p^i \cdot r^i}$	Accuracy	Precision and recall
OTA	$1 - \frac{\sum_i (n^i_{fn} + n^i_{fp})}{\sum_i g^i}$	Accuracy	False positive and false negative
OTP	$\frac{1}{\lvert M_s \rvert} \sum_{i \in M_s} \frac{T^i \cap GT^i}{T^i \cup GT^i}$	Accuracy	Average overlap over matched frames
ATA	$\frac{1}{N_{frames}} \sum_i \lvert \frac{T^i \cap GT^i}{T^i \cup GT^i} \rvert$	Accuracy	Average overlap
Deviation	$1 - \frac{\sum_{i \in M_s} d(T^i, GT^i)}{\lvert M_s \rvert}$	Location	Centroid normalized distance
PBM	$\frac{1}{N_{frames}} \sum_i \left[1 - \frac{Distance(i)}{T_h(i)} \right]$	Location	Centroid L1-distance

frames to long term with thousands of frames. The targets of the videos include human body for 36 sequence, face or head for 26 sequences, and other versatile objects for remaining 38 sequences. Among 100 videos, 74 sequences are color videos, 26 sequences are gray-scale videos.

For annotation format, each video fragment has a single target and is annotated by rectangular bounding boxes along main axes for every frame. To be noticed, sequences Jogging and Skating2 has two annotated targets for each sequence. Yet, in each run, only the initial bounding box of a single target is provided. In addition, every sequences are labeled for eleven different visual attributes as Table 2.2. A sequence can have two or more labels. For example, sequence Girl has labels SV, OCC, IPR, OPR.

For evaluation protocol, OTB adopts three types of evaluations involving one-pass evaluation (OPE), temporal robustness evaluation (TRE), and spatial robustness evaluation (SRE). In OPE, a tracker is fed with the initial bounding box in the first frame and predict bounding boxes for every following frames until the end of a video. In TRE, the temporal robustness means how an initialization of a tracker performs in different time-

doi:10.6342/NTU201801036

steps. A video sequence is divided into 20 segments and a tracker is evaluated on each segment using OPE and the overall statistics of segments of the same video are tallied. In SRE, the spatial robustness means how an initialization of a tracker performs in noisy initial bounding box. In each evaluation in SRE, the initial bounding box is perturbed by either one of 4 center shifts, 4 corner shifts and 4 scale shifts. The shift amount is one of $\pm 10\%$ of target size along two main axes. The scale ratios include 0.8, 0.9, 1.1, 1.2 to the ground truth. In SRE, each tracker is evaluated 12 times where each run adopts a single perturbation method. In OTB2013, no re-initialization is provided when the target is lost. In OTB2015, re-initialization is provided directly on the next frame of the failed frame. The failure is defined when the overlap of the prediction and the ground-truth dropped to a threshold θ. When re-initialization is considered, two new protocols are proposed in OTB2015 as follows 1. One Pass Evaluation with Restart (OPER) 2. Spatially Robustness Evaluation with Restart (SRER).

In addition, in evaluation of OTB, a tracker should be on-line/causal, indicating that the current prediction should only depend on the initial bounding box, previous predictions, and current and previous frames. For the parameters, sequence-specific parameters tuned manually are not allowed. All parameters should be fixed or generated automatically for all sequences.

For evaluation metrics, OTB adopts success plot and precision plot. Figure 2.1 is an example of the results in OTB2013 [32]. For precision plot, x-axis is center location error in units of pixels, y-axis is precision denoting the percentage of frames with center location errors smaller than a specific threshold in x-axis. A representative precision score for each tracker is the precision value in threshold = 20 pixels. For success plot, x-axis is overlap score and y-axis is success rate. Overlap score is denoted as $S = \frac{|r_t \cap r_a|}{|r_t \cup r_a|}$ where r_t is set of pixels inside the ground truth bounding box, r_a is the set of pixels inside the prediction bounding box, and $|\cdot|$ denotes the number of elements inside a set. Success rate

doi:10.6342/NTU201801036

Table 2.2: List of Visual Attributes in OTB [32].

Attr	Description
IV	Illumination Variation - the illumination in the target region is significantly changed.
SV	Scale Variation - the ratio of the bounding boxes of the first frame and the current frame is out of the range $[1/t_s, t_s], t_s > 1$ (t_s=2).
OCC	Occlusion - the target is partially or fully occluded.
DEF	Deformation - non-rigid object deformation.
MB	Motion Blur - the target region is blurred due to the motion of target or camera.
FM	Fast Motion - the motion of the ground truth is larger than t_m pixels (t_m=20).
IPR	In-Plane Rotation - the target rotates in the image plane.
OPR	Out-of-Plane Rotation - the target rotates out of the image plane.
OV	Out-of-View - some portion of the target leaves the view.
BC	Background Clutters - the background near the target has the similar color or texture as the target.
LR	Low Resolution - the number of pixels inside the ground-truth bounding box is less than t_r (t_r=400).

is the percentage of frames with overlap score larger than the specific threshold in x-axis. For the case in considering restart, a new metric is the number of restart w.r.t the failure threshold.

For evaluation tool-kit, a code library for evaluation and a set of codes including most publicly available trackers in 2013 in total of 29 trackers is released. To add a new tracker, only the code with an interface under specification to produce bounding box output and fps is required.

2.4.3 Visual Object Tracking (VOT)

In 2013, the Visual Object Tracking (VOT) workshop [34] was organized in conjunction with ICCV2013. The goal of the dataset aims to collect a small number of videos with various real-life visual phenomena. A large pool of video sequences were collected and clustered based on their visual attributes. A subset of 16 sample videos were selected

doi:10.6342/NTU201801036

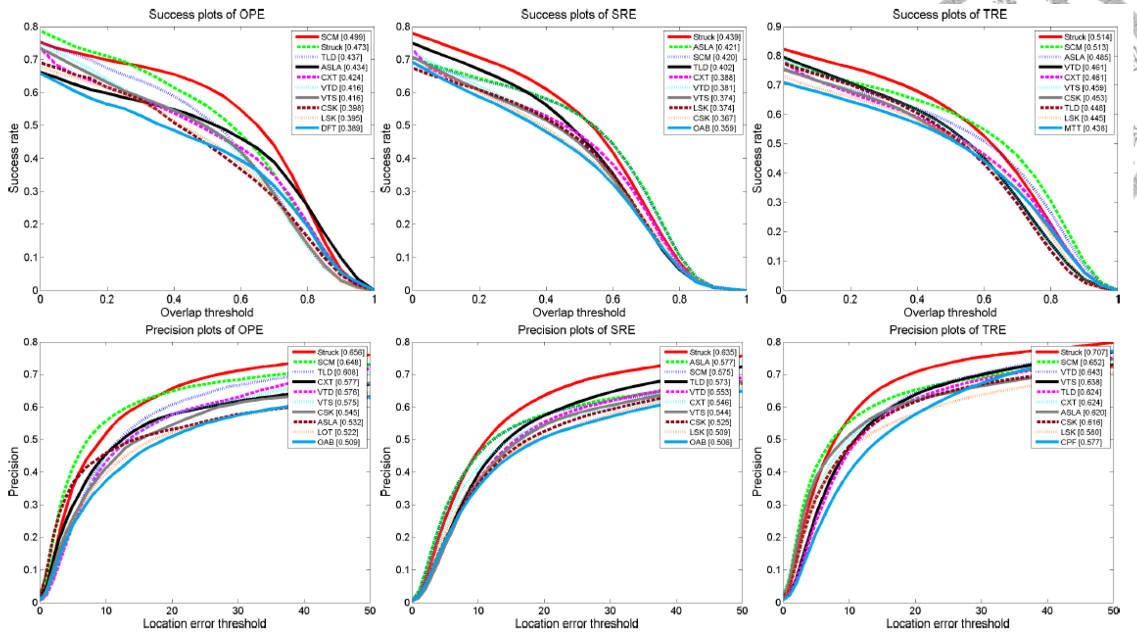


Figure 2.1: The success plot and precision plot of OTB2013 in evaluation of OPE, SRE, TRE [32].

such that different circumstances are still well represented under selection. In every year after 2013, VOT is organized in conjunction with ECCV or ICCV. Every year, the dataset or the evaluation tool-kit are updated to follow the development in tracking field.

For qualities of videos, the lengths of videos from VOT2013 to VOT2018 are short in an order of hundreds frames between 2 to 20 seconds. In 2018, a new long-term sub-challenge of VOT, VOT-LT2018, is organized. In this sub-challenge, videos are long-term in an order of thousands frames above one minute. Most targets of the videos in VOT-LT2018 are vehicles or pedestrians. For the number of videos, 16 videos are in VOT2013, 25 in VOT2014, 60 in VOT2015, 60 in VOT2016, 60 in VOT2017, and 35 in VOT-LT2018. VOT2016 dataset has the same sequences as VOT2015 yet annotations re-labeled. The sequences and annotations in VOT2017 dataset is the same as in VOT2016. VOT2018 removes sequences succeeded by most trackers in VOT2017 challenge and fills another more difficult sequences.

For annotation formats, each sequence has a single target and is annotated by quadrilaterals. To be noticed, the quadrilateral is not necessarily a rectangular bounding box

doi:10.6342/NTU201801036

except that in VOT2013 the annotations are rectangular bounding boxes along main axes. For visual attribute labels, all sequences are labelled per-frame by five visual attributes as follows 1. occlusion 2. illumination change 3. motion change 4. size change 5. camera motion. For each visual attribute in each frame, a zero/one is given to indicate whether the frame under a certain circumstance. The annotation rule is that more than 60% of the pixels of the target should be inside the quadrilateral.

For evaluation protocol, provided with the initial bounding box in the first frame, a tracker predicts bounding boxes in every following frames. When a tracker fails on frame t meaning the overlap of the ground truth and the prediction drops to zero, a re-initialization is conducted five frames later at frame $t + 5$. In addition, the re-initialization is a complete re-initialization, indicating the information from previous frames, ground-truths, and predictions should be discarded. The reason of five frame gap after the failure in re-initialization is to provide a burn-in period to render the tracker unbiased by the initialization polluted by an influential failure factor in the failed frame. For example, when a failure occurs on an occluded frame, a direct re-initialization in the next frame may be incorrect since the object enclosed by the ground-truth may be the object in front of the target instead of the target.

VOT includes three different experiment setting as follows:

- Experiment 1: The experiment tests a tracker initialized by **ground-truths** on **color videos**.
- Experiment 2: The experiment is the same as the first experiment excluding that a tracker is initialized by **noisy bounding-boxes**. The noisy bounding boxes are created by inducing a perturbation on the size and location of the ground-truth by a randomness sampled uniformly from $\pm 10\%$ interval of the ground truth size.
- Experiment 3: The experiment is the same as the first experiment with the color

doi:10.6342/NTU201801036

videos changed to **grayscale videos**.

All the experiments are executed by the evaluation code and each experiment is executed 15 times. The performance metric of a single experiment is averaged by the number of the repeat of the experiment, e.g. 15.

For evaluation metrics, VOT chose tow orthogonal measures: accuracy and robustness. For accuracy, the overlap of the ground truth and the prediction is adopted. The accuracy of a tracker on a sequence in an experiment is summed and averaged per frame and then over the number of repeated times of an experiment. For robustness, the number of failure is adopted. The robustness of a tracker of a sequence in an experiment is summed and averaged over the number of repeated times of an experiment. To evaluate all trackers proposed to the challenge, M. Kristan et al. [34] proposed a ranking-based methodology. After evaluating the accuracy and robustness of a tracker on a sequence, the rank of a measure of a tracker on a sequence is derived. After averaged over sequences, the ranking with respect to a measure metric is obtained. Given all measures the same weight, the ranking of a tracker on an experiments is acquired. Given all three experiments the same weights, the final ranking score is obtained. Since this methodology is based on ranking, it's a relative score with respect to all trackers proposed in that year.

For evaluation tool-kit, M. Kristan et al. [34] proposed a library written in MATLAB and supports trackers of different languages. In addition, the code or the code archive of the trackers are included in the released packages with setup interfaces.

doi:10.6342/NTU201801036

Chapter 3 Proposed Faster-MDNet

In this chapter, the proposed tracking algorithm Faster-MDNet is introduced. This chapter is organized as follows. In section 3.1, a brief introduction to the proposed Faster-MDNet is included. In section 3.2, related works are presented. In section 3.3, MD-Net is introduced. In section 3.4, the proposed model Faster-MDNet including model architecture, three-phase training, and prediction method is introduced. In section 3.5, the experiment results are presented. In section 3.6, this chapter ends up with a conclusion.

3.1 Introduction

Among tracking-by-detection trackers with convolutional neural network as the binary classifier, online learning is a common method to update the model. These trackers will online collect appropriate positive and negative samples and finetune the network whenever the model adaptation is required, e.g. prediction confidence is too low or a fixed time duration is passed. Yet, since neural network is designed to spend most time on training to learn the data distribution and predict relatively in a short time, the online learning procss will create a bottleneck on time complexity and render the tracking algorithm away from applications.

For example, one long-term update in MDNet [1] takes around 10-15 times the time of prediction in one frame. Even though the long-term update occurs every ten frames in the settings of MDNet, this finetuning process has at least doubled the overall temporal cost

doi:10.6342/NTU201801036

without consideration of the time of short-term update and collecting training samples.

In addition, the power of RNN has not been well developed in the area of visual tracking due to the two reasons mentioned above. Hence, we develop a model adaptation algorithm that utilizes recurrent neural network to learn how to update the model from online learning. Proposed RNN-based algorithm can be finetuned in testing to acquire the tracked target.

Furthermore, we reuse the candidate bounding-boxes for predictions to update the model, which removes the time and the memory of collecting extra finetuning samples. Our RNN-based method predicts and updates the model at the same time, thus reducing the temporal cost of model adaptation to nearly the same as prediction time. Proposed algorithm is a general idea that can be applied onto any trackers based on tracking-by-detection with neural network classifier. The overall algorithm can be treated as an addition of the RNN branch to the original tracker. The RNN branch can be designed individually without changing the prediction branch of the tracker.

3.2 Related Works

I. Visual Tracking Algorithms

Most class-agnostic rgb tracking algorithms can be divided into two major genres: the generative model and the discriminative model. The generative model attempts to describe the appearance of the targets and search for the best-fitting regions in frames [35, 36]. The discriminative model aims to transfer visual tracking into a problem of separation of foreground and background where the foreground is the target model.

In 2016, correlation filters [37] and CNNs [1, 37, 38] have gained a large attention in the field of visual tracking. Among the top-10 trackers of VOT2016, 4 trackers were derived from CNNs, 4 trackers were variations of correlation filters. Rank-1 tracker C-COT

doi:10.6342/NTU201801036

[37] is based on correlation filter and utilizes CNN features for strong representations. In CVPR2017, M. Danelljan et al. [37] proposed another correlation-filter-based trackers: ECO, which runs faster and more accurate than C-COT.

II. Tracking by Detection

In the field of object detection, the core idea behind some well performed detection algorithms is to predict the result by classification and regression of candidate bounding boxes. Tracking by detection adopts the idea and predict the tracking result by proposing candidate bounding-boxes and evaluating each candidate. The major difference between detection and tracking-by-detection [1, 37, 38] is that tracking by detection requires to update the model to adapt to the change of target appearance.

III. Recurrent Neural Network on Visual Tracking

RNN is a neural network modeling designed to process temporal-spatial information and has gained huge attentions in language processing field like language understanding, language generation, captioning, etc. Recently, [39, 40, 41] has used RNN to address temporal-spatial information in visual tracking. However, they focus on artificially generated sequences and synthesized data. [42] is a tracker which uses deep convolutional neural network and object detection framework [43] to extract image features and feed them into LSTM to predict the target. However, [42] does not finetune the network for the target object in the first frame of the video, which restricts the tracker to track only the saliency or pre-trained objects in one video rather than the object enclosed by the bounding-box of the first frame. Strictly speaking, [42] does not belong to visual tracking algorithms. In cases like tracking a ball when a group of people are playing basketball, [42] may not be appropriate.

The challenges of RNN on visual tracking lies primarily on two aspects:

doi:10.6342/NTU201801036

Target Acquisition

In visual tracking, the tracker requires the target acquisition step to obtain the tracked target. In most applications of CNN, one image has only one ground-truth information. However, in visual tracking, given a video frame, the prediction still depends on the ground-truth rectangle. Hence, a good data acquisition strategy is required to enable the RNN-based tracking algorithm to realize the tracking target.

Number of Training Videos and training time

The abundant amount of training data is a vital factor for a successful supervised learning algorithm. For visual tracking, to train an RNN-based algorithm to learn multiple visual variances of unexpected objects, videos of different visual attributes are necessarily required.

Yet, the storage requirements and the cost to collect abundant videos are challenge to conquered. In addition, even if such dataset is built, a tremendous amount of training time is required to train with such abundant videos. A model that can be trained more efficiently is important.

3.3 MDNet

MDNet is a tracker based on tracking-by-detection and VGG-M network [44]. In MDNet, VGG-M is first pre-trained on ImageNet dataset [45]. Then, the network is transferred to video domain by multi-domain learning. Before tracking, the last layer will be randomly initialized and finetuned on the first frame. In tracking, positive and negative samples are collected from frames with confident predictions. A short-term update is executed when the prediction score is too low, which takes around half the time of prediction in one frame. A long-term update is executed every ten frame, which is around 10-15 times the time of

doi:10.6342/NTU201801036

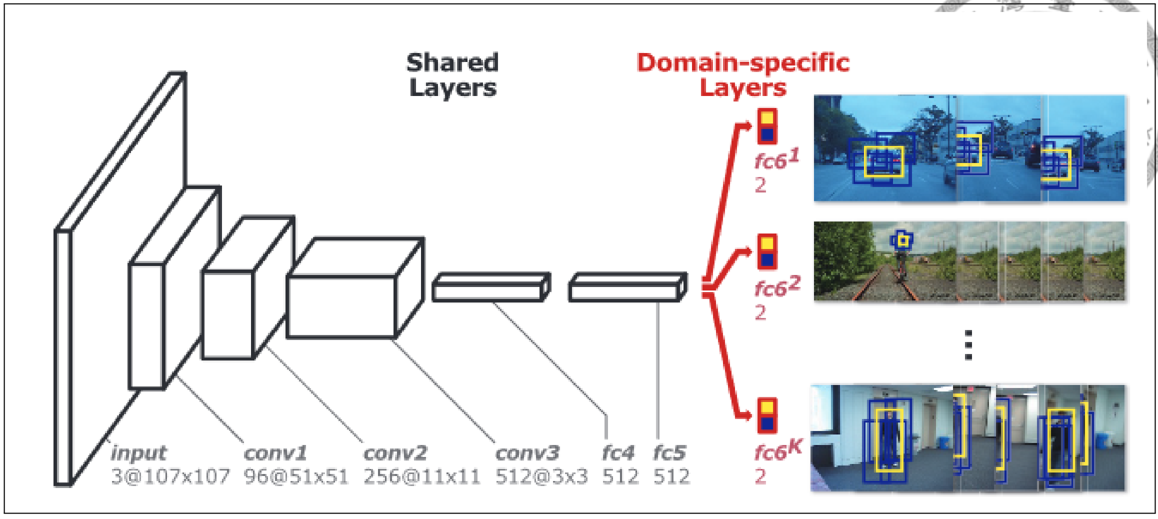


Figure 3.1: The architecture of Multi-Domain Network [1] consisting of shared layers of each domain branch and domain-specific layers. Yellow box and blue boxes are postive and negative samples for each domain.

prediction in one frame. The details can be referred to the paper of MDNet [1].

The overall temporal cost of MDNet in prediction including the time of (1) prediction (2) collecting finetuning samples (3) short-term finetuning (4) long-term finetuning. (1)(2) occur nearly in every frame. (3) occurs only when the confidence is two low. (4) occurs in every 10 frames. (1), (2), and (3) can be executed in a very short time, while (4) can be 10-15 times longer than others. The proposed algorithm completely replaces (4) and (2) and the additional temporal cost is smaller than that of (1).

3.4 Proposed Faster-MDNet

In this section, we explain by example of MDNet how we can aggregate a tracker with proposed RNN-based model adaptation.

As in Figure 3.2, the original MDNet has one branch to predict score for each proposal. The original MDNet is aggregated with a new branch which is a copy of the fully connected layers. The new branch and the original branch are notated as **prediction branch** and **model adaptation branch** respectively.

doi:10.6342/NTU201801036

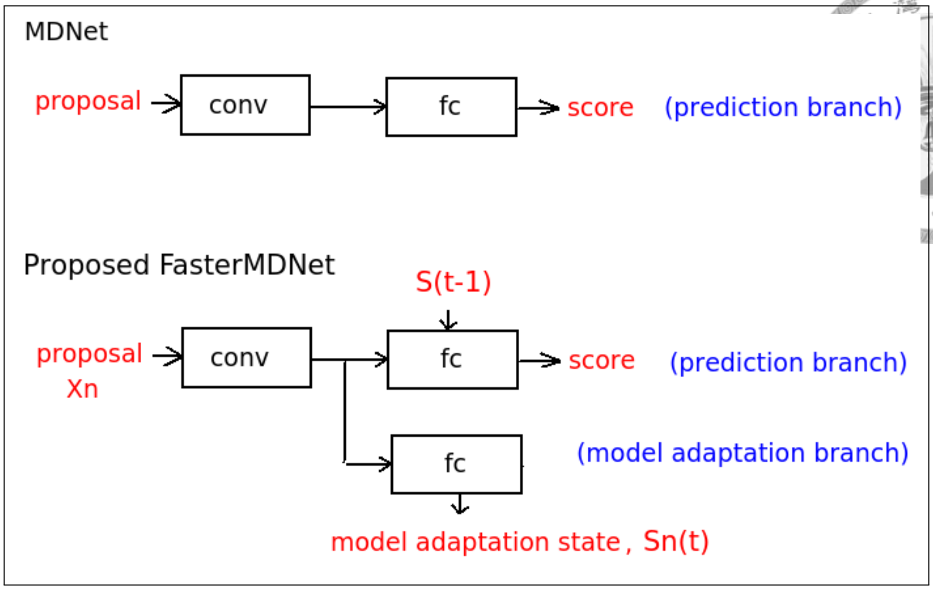


Figure 3.2: Aggregation of RNN model adaptation onto MDNet. The original MDNet has only one branch for prediction. Proposed Faster-MDNet has two branches: the prediction branch and the model adaptation branch. The model adaptation branch predicts the model adaptation states.

For each proposal, the prediction branch produces a score and the model adaptation branch produces a model adaptation state. The model adaptation state is the outputs of all fc layers in model adaptation branch.

As in Figure 3.3, since there are a bunch of proposals, a number of model adaptation states and scores will be produced. To emphasize the higher-scored proposals, the model adaptation states are weighted by their scores and taken the mean to produce the final model adaptation state.

$$S_i(t) = \frac{1}{N}\sum_{n=1}^{N} w_n S_{n,i}(t). \tag{3.1}$$

At timestep t, w_n is the score of n^{th} proposal, $S_{n,i}(t)$ is the output of n^{th} proposal at fc$_i$ layer, $S_i(t)$ is the final model adaptation state at fc$_i$ layer.

In prediction at timestep $t+1$, the model adaptation state is treated as the hidden state

doi:10.6342/NTU201801036

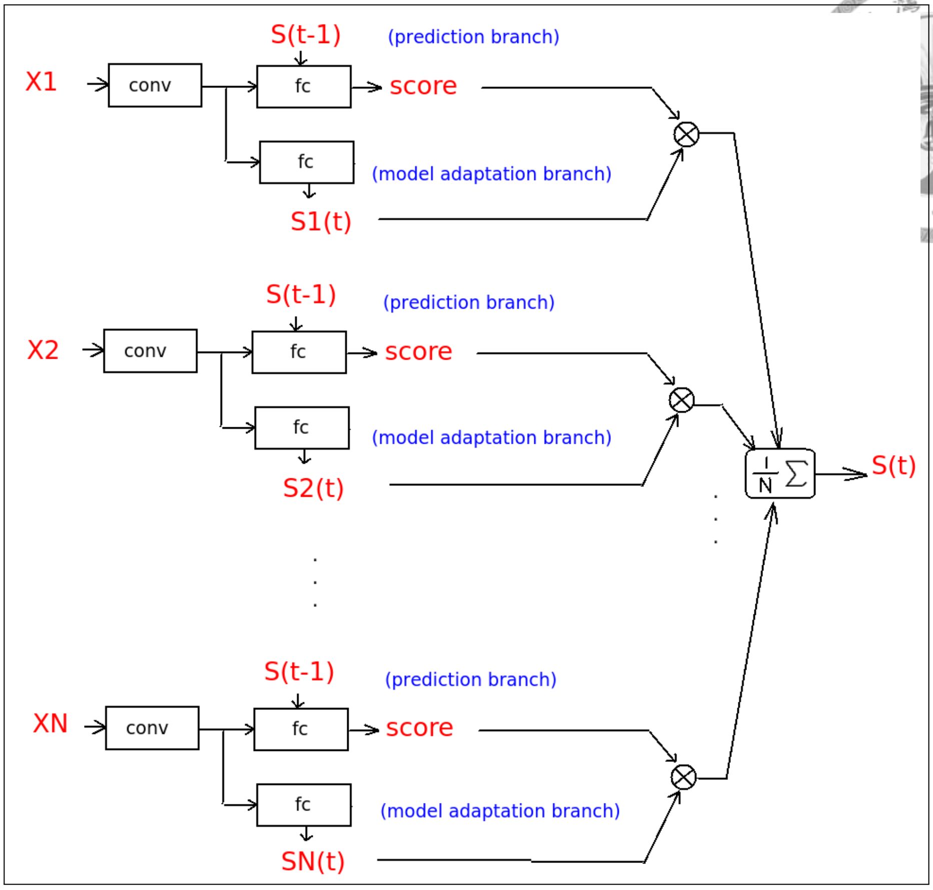


Figure 3.3: Generation of the final state from state of each proposal. X_i is the i^{th} proposal, $S_i(t)$ is the model adaptation state generated by i^{th} proposal at timestep t, $S(t)$ is the final model adaptation state at timestep t.

of RNN and added directly onto fc4, fc5, and fc6 in prediction branch. That is,

$$X_{n,4}(t) = \sigma(W_{3,pred}X_{n,3}(t) + S_4(t-1)). \tag{3.2}$$

$$X_{n,5}(t) = \sigma(W_{4,pred}X_{n,4}(t) + S_5(t-1)). \tag{3.3}$$

$$X_{n,6}(t) = \sigma(W_{5,pred}X_{n,5}(t) + S_6(t-1)). \tag{3.4}$$

where $X_{n,i}(t)$ is the features of proposal n at fc$_i$ layer in prediction branch at timestep t, $W_{i,pred}$ is the weights at fc$_i$ layer in prediction branch, σ is the nonlinearity.

In this paper, we treat the final model adaptation state as the mean of score-weighted model adaptation state of all proposals. Yet, the relation among the final state, the score and the state of each proposal can be more complicated as a function, e.g. a model of

doi:10.6342/NTU201801036

neural network. That is,

$$S_i(t) = f(S_{i,1}(t), S_{i,2}(t), ..., S_{i,N}(t), w_1, w_2, ..., w_N). \tag{3.5}$$

Plus, the RNN state can have more complicated relations with the prediction branch in the next step. For example, we can use LSTM cells to handle those states rather than direct summation. That is,

$$X_{n,i}(t) = g(S_i(t-1), X_{n,i-1}(t)). \tag{3.6}$$

However, the number of training videos would be an issue when we proceed toward more complicated model.

3.4.1 Three-Phase Training

With this aggregation, we can train the prediction branch and the model adaptation branch at the same time in the following way. We divide the overall training into three phases:

I. Pre-Training on ImageNet

In this phase, we pretrain the VGG-M network on multi-label classification of ImageNet dataset. This phase is to train the network by millions of images to empower the network the ability to extract high-order informations.

II. Multi-Domain Learning on Frames

In this phase, we train the VGG-M network by multi-domain learning as in MDNet. This phase is to transfer the model from multi-label classification to the domain of visual tracking.

doi:10.6342/NTU201801036

III. Model Adaptation Learning on Videos

In this phase, we first create the model adaptation branch by copying both the structure and the weights from fc layers of the network pretrained in phase 1 and phase 2 as mentioned in section III.B. Second, for each video, we randomly initialized the fc6 layer in prediction branch and prepare a zero state as our initial model adaptation state. Third, we finetune the prediction branch by extracting positive and negative samples from the first frame. Finally, we unroll our RNN by max-numstep $M (M = 40$ in our experiment), and in each training batch, we extracts positive and negative proposals from consecutive M frames as our training data to train the unrolled RNN. The final model adaptation state will be fed into next training batch as the initial state adaptation state.

3.4.2 Prediction

In predictoin, we randomly initialize the last layer in prediction branch and prepare a zero-state as the initial model adaptation state. Then we finetune the prediction branch on the first frame. In the following frames, the model adaptation is completed primarily by the model adaptation branch. Thus, the time complexity can be largely decreased to nearly the prediction time.

3.5 Experiments

3.5.1 Implementation Details

The experiment is implemented in Python2.7 and TensorFlow r1.0.0 on Ubuntu 14.04 LTS, Intel Core i7-6700K @ 4.00GHzx8, NVIDIA GeForce GTX TITAN X.

The ImageNet-pretrained VGG-M network is downloaded from caffe-model zoo. We trained the network on VOT 2013, 2014 and 2015 and tested on OTB-30 dataset [32]. All

doi:10.6342/NTU201801036

Algorithm 1 Model Adaptation Learning in Faster-MDNet

```
Require:
    Pretrained CNN filters $\{w_1, w_2, ..., w_5\}$.
    $M$: $\{M_{conv}, M_{pred}, M_{adapt}\}$, the overall model which consists of shared convolution
    layers, fc layers in prediction branch, and fc layers in model adaptation branch.
    $V$: the set of videos.
    $L_c$: the maximum length of a training clip.

 1: $M_{conv} \leftarrow \{w_1, w_2, w_3\}$                ▷ set filters in conv layers by pretrained weights
 2: $M_{pred} \leftarrow \{w_4, w_5\}$               ▷ set filters in prediction branch by pretrained weights
 3: $M_{adapt} \leftarrow \{w_4, w_5\}$      ▷ set filters in model adaptation branch by pretrained weights
 4: for all $v \in V$ do
 5:     $L_v \leftarrow$ number of frames in $v$
 6:     $\{x_t, y_t | 0 \leq t < L_v - 1\} \leftarrow v$                ▷ x: frame image, y: groundtruth
 7:     $i \leftarrow 0$
 8:     $S \leftarrow \{0\}$                                                        ▷ RNN state
 9:
10:     Randomly initialize $w_6$ in prediction branch
11:     Finetune $M_{pred}$ by $\{x_0, y_0\}$
12:     $M_{unrolled} \leftarrow$ unroll $M$ for $L_c$ steps
13:     while $i + L_c \leq L_v$ do
14:         $X_{i,L_c} \leftarrow \{x_t | i \leq t < i + L_c\}$
15:         $Y_{i,L_c} \leftarrow \{y_t | i \leq t < i + L_c\}$
16:         $S_{next}, gradients \leftarrow forward\_backward(M_{unrolled}, S, X_{i,L_c}, Y_{i,L_c})$
17:         update $M_{adapt}$ by $gradients$
18:         $S \leftarrow S_{next}$
19:         $i \leftarrow i + L_c$
20:     end while
21: end for
```

Algorithm 2 Prediction in Faster-MDNet

```
Require:
    $M$: the overall model with shared convolution layers, prediction branch, and model
    adaptation branch
    $\{x_0, x_1, x_2, ..., x_{N-1}\}$: a sequence of frames
    $y$: the groundtruth of the first frame

 1: Randomly initialize $w_6$ in prediction branch
 2: Finetune prediction branch by $\{x_0, y\}$
 3: $S, \hat{y} \leftarrow forward(M, x_0, \emptyset)$                              ▷ initial state
 4: $Y = \{\}$
 5: for all $t = 1, 2, ..., N - 1$ do
 6:     $S_{next}, \hat{y} \leftarrow forward(M, x_t, S)$
 7:     $Y \leftarrow Y \cup \{\hat{y}\}$
 8:     $S \leftarrow S_{next}$
 9: end for
10: return Y
```

doi:10.6342/NTU201801036

the settings of phase2 training are the same as in MDNet. In phase3 training, we trained for 10 epochs. To prevent gradient explosion, we clipped the gradient by 1. The overall training takes around one week.

3.5.2 short-term finetuning

Due to the limitation of training data, we preserve the short-term finetuning of MDNet which takes only half the time of prediction. The additional time of short-term finetuning is negligible compared with the time of the replaced long-term finetuning.

3.5.3 Results

Elapsed Time

Table 3.1: The average elapsed time on Faster-MDNet and MDNet.

Task at one frame	elapsed time (second)
Faster-MDNet: predict scores of all proposals and generate the final model adaptation state	~0.8
Faster-MDNet: predict scores of all proposals and generate the final model adaptation state + short-term finetuning	~1.3
MDNet (prediction and long-term finetuning)	~11.9

We compared MDNet and Faster-MDNet. To reduce the influence of implementation language, we re-implement MDNet on python and tensorflow. Table 3.1 shows the result of elapsed time.

Accuracy

Table 3.2 shows the average overlap scores on OTB-30 dataset [32] with one recent tracker ROLO [42], the basis of ROLO: YOLO [43], MDNet [1] and proposed Faster-MDNet. The average overlap score of a sequence is the intersection over overlap between

doi:10.6342/NTU201801036

prediction and groundtruth averaged by frames. Suppose that for a sequene with frames $\{f_1, f_2, ..., f_N\}$ and groundtruths $y_1, y_2, ..., y_N$, a tracker create predictions $\hat{y}_2, \hat{y}_3, ..., \hat{y}_N$. The average overlap score S is defined as follows.

$$S = \frac{1}{N-1} \sum_{n=2}^{N} \frac{y_n \cap \hat{y}_n}{y_n \cup \hat{y}_n}. \tag{3.7}$$

Table 3.2: Summary of Average Overlap Scores (AOS) results.

Sequence	ROLO	YOLO+SORT	MDNet	Faster-MDNet
Human2	0.545	0.636	0.728	0.680
Human9	0.352	0.193	0.383	0.095
Gym	0.599	0.460	0.563	0.500
Human8	0.364	0.416	0.638	0.407
Skater	0.618	0.283	0.651	0.553
SUV	0.627	0.455	0.684	0.781
BlurBody	0.519	0.337	0.690	0.041
CarScale	0.565	0.627	0.498	0.380
Dancer2	0.627	0.201	0.746	0.614
BlurCar1	0.537	0.082	0.738	0.02
Dog	0.429	0.241	0.457	0.359
Jump	0.547	0.208	0.082	0.270
Singer2	0.588	0.400	0.676	0.562
Woman	0.649	0.358	0.781	0.643
David3	0.622	0.224	0.708	0.590
Dancer	0.755	0.551	0.725	0.683
Human7	0.456	0.291	0.596	0.701
Bird1	0.362	0.048	0.403	0.333
Car4	0.768	0.690	0.708	0.028
CarDark	0.674	0.211	0.825	0.790
Couple	0.464	0.204	0.477	0.565
Diving	0.659	0.166	0.363	0.194
Human3	0.568	0.386	0.551	0.038
Skating1	0.572	0.443	0.551	0.485
Human6	0.532	0.376	0.497	0.380
Singer1	0.653	0.332	0.404	0.794
Skater2	0.652	0.532	0.585	0.122
Walking2	0.595	0.362	0.702	0.222
BlurCar3	0.539	0.191	0.801	0.787
Girl2	0.517	0.337	0.708	0.706

doi:10.6342/NTU201801036

3.6 Conclusions

An model adaptation method based on recurrent neural network on tracking-by-detection tracker with neural network classifiers is proposed to extremely decrease the time of online finetuning and deliver real-time computational cost with little sacrifice of performance. The method can be generalized to any tracker based on tracking-by-detection with neural network classifier and can be further developed with more complicated recurrent network architectures or more complicated proposal state merging policy. An experiment on MDNet and simple recurrent network settings, Faster-MDNet, is conducted and shows a huge decrease of computational cost on prediction time with small drop on performance.

doi:10.6342/NTU201801036

Chapter 4 Proposed RDisp: Recurrent Detection is Powerful

A visual tracking algorithm RDISP based on detection models pretrained on large datasets and convolutional recurrent neural network (ConvRNN) cells is proposed. Our model is composed of two branches. The initial state extraction branch is responsible for extracting the initial state of ConvRNN cells and the prediction branch is responsible for predicting bounding-boxes based on the current image and previous states. The structure of proposed model is modified from detection model with insertion of ConvRNN cells, thus benefiting from the low computational cost of region proposal networks and reaching a extremely fast speed. In training the proposed model, conventional back-propagation through time does not work due to inefficient utilization of video data. A two stage clip training is proposed to replace back-propagation through time. The experiment shows the proposed tracker can handle multiple circumstances and reduces the time complexity to around 25 fps.

4.1 Introduction

Class-agnostic visual tracking is a fundamental and challenging task in the field of computer vision. The challenges include versatile visual changes in video frames such as illumination change, occlusion, deformation and lack of learnable prior knowledge from

doi:10.6342/NTU201801036

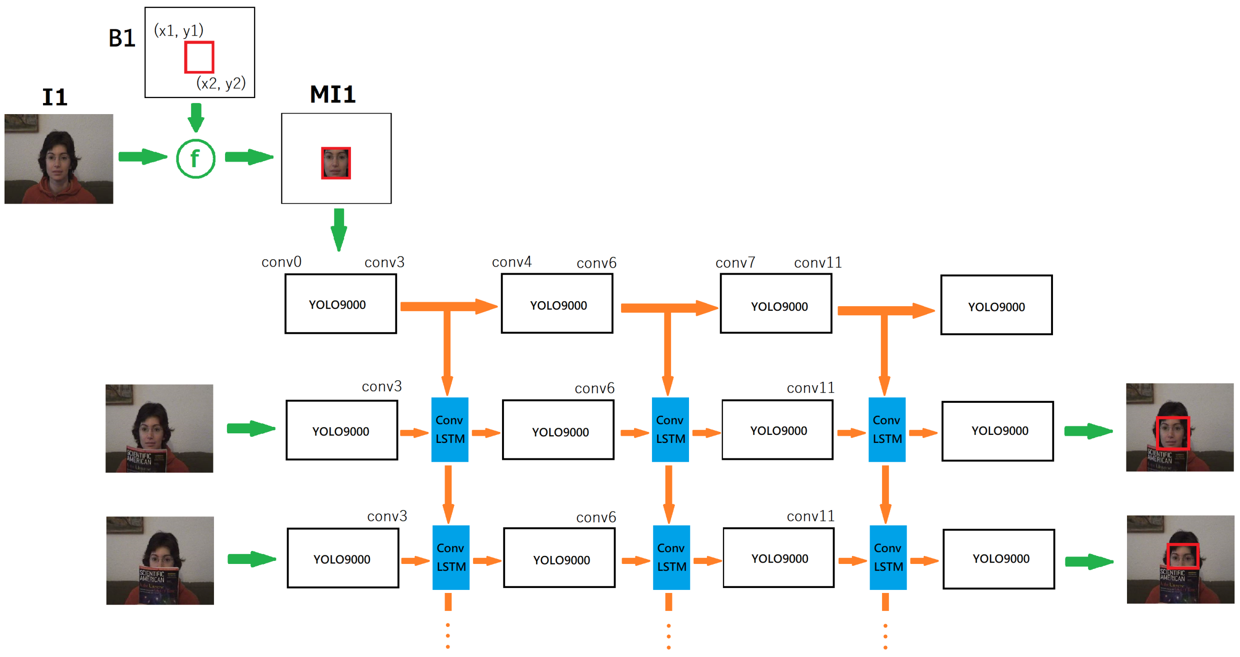


Figure 4.1: The structure of the proposed model including the initial state extraction branch and the prediction branch. The initial state extraction branch is fed with the masked image to generate the initial state. The prediction branch predicts bounding boxes based on the image and previous states. (The frame images are from OTB100 FaceOcc1 sequence).

large datasets in class-agnostic settings.

Based on tracking-by-detection technique and a pre-trained image classification network on ImageNet dataset as the classifier, the MD-Net [1] has presented impressive accuracy on class-agnostic tracking dataset. Yet, the problem of high time complexity of online training to adapt the classifier to the time-varying appearance model of the target has long been an obstacle toward real-world applications.

Recently, convolutional recurrent neural network cells have been widely adopted in video-related tasks. The advantage of ConvRNN cells are that the input and the state are blobs preserved as three dimensional size which are more representative for image-related information. Thus, ConvRNN cells are suitable choices for temporal-spatial information representation in video frames.

In addition, even though the idea behind tracking-by-detection is adopted from detection field, trackers based on tracking-by-detection techniques only adopt the idea of taking

doi:10.6342/NTU201801036

proposals and classifying proposals but not yet attempted to transfer pre-trained detection model to tracking field to benefit from fast speed due to sharing features of region proposal networks and abundant information from detection datasets.

A method incorporating ConvRNN cells into pretrained one-stage detection models to form an end-to-end trainable tracking model with pretrained parameters is proposed. In detection fields. To train such end-to-end tracking model, a training using back-propagation through time, it has failed in decreasing training loss due to several reasons. To overcome these obstacles, a two stage clip training is proposed to replace the original back-propagation through time. The two stage clip training split the overall training into two stages. The first stage focus on how to extract appropriate recurrent neural network states and predict based on these states. The second stage focus on how to update the states given new video frames and predictions.

4.2 Related Works

The Faster-MDNet [46] endeavoured to decrease the time complexity of the MDNet by aggregating the original classifier with a model adaptation branch and predicting based on the model adaptation state collected and merged by a function fed with all model adaptation states generated by each proposal. Yet, the proposals in the Faster-MDNet are generated by selective search, thus propagating the error of selective search through time. In addition, each proposal is cropped and resized into the input size of the classifier, so the recurrent neural network state contains only the appearance model of the target. The trajectory of the predicted bounding-boxes is not encoded in the state for rejection of similar objects in different locations.

Other works based on the MD-Net focus on the classifier model structure and does not handle the problem of time complexity of online training.

doi:10.6342/NTU201801036

For detection model, there are primarily two categories, one-stage structure and two-stage structure. The representation of two stage structure is Faster-RCNN [5]. In two stage structure, the input image is fed into a region proposal network to generate region proposals. Then these region proposals are fed into a convolutional neural network to classify the class label, the bounding-box regression terms, and additional information (e.g. mask). The representations of one stage structure are SSD [47], YOLO [43], and YOLO9000 [48]. In one stage structure, the image is fed into a region proposal network directly and predicts everything (e.g. object score, regression terms, and class labels) in the output of the network.

Since the model based on tracking-by-detection classify the proposals into target or non-target, one-stage detection model is adopted for the pretrained detection model in this work.

4.3 Methods

4.3.1 Transfer Learning from Detection

In recent years of detection field, the computational cost has reached an impressive milestone thanks to the sharing features in fully convolutional network and the problem settings by anchor boxes.

Compared with detection, the major difference of the information in tracking is the information involving similar object rejection and the detailed appearance model of the target, both derived temporally and on-line. If this two temporal information can be properly merged into detection model, tracking algorithms can benefit from pretrained detection models to tremendously decrease the computational complexity by the network structure and learn the prior knowledge of detection dataset encoded in pretrained detection models.

doi:10.6342/NTU201801036

The idea behind the proposed end-to-end trainable model is that the detailed appearance model and the similar object rejection information in tracking can be modelled as residual to the appearance model in detection. Based on the prior knowledge learned from large datasets of static images, the structure and sharing features, the tracking algorithms can focus on temporal information for similar object rejection and detailed appearance models.

4.3.2 Injection of Temporal Information by Convolutional Recurrent Neural Network Cells

The proposed end-to-end trainable model is an incorporation of convolutional recurrent neural network cells (ConvRNN) into one-stage detection model. ConvRNN is the modification of RNN cell such that the input and the state are three dimensional blobs which are more representative for image-related information. In ConvRNN, the multiplication with parameter matrix is replaced by convolution layer. For example, LSTM cells can be described as follows.

$$i_t = \sigma(W_i * [X_t h_{t-1}] + b_i). \tag{4.1}$$

$$o_t = \sigma(W_o * [X_t h_{t-1}] + b_o). \tag{4.2}$$

$$f_t = \sigma(W_f * [X_t h_{t-1}] + b_f). \tag{4.3}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \sigma(W_g * [X_t h_{t-1}] + b_g). \tag{4.4}$$

$$h_t = o_t \odot tanh(c_t). \tag{4.5}$$

where σ is the activation function (e.g. ReLU), X_t, h_t, c_t, i_t, o_t, f_t are the input, the hidden state, the cell state, the input gate, the output gate, the forget gate of timestep t respectively,

doi:10.6342/NTU201801036

W and b are corresponding weight and bias of each gate, and $\odot$ is the Hadamard product.

For ConvLSTM [49], the only difference is to replace multiplication with matrix with convolution with filters. Thus the equations of ConvLSTM are as follows.

$$i_t = \sigma(conv_i([X_t h_{t-1}])). \tag{4.6}$$

$$o_t = \sigma(conv_h([X_t h_{t-1}])). \tag{4.7}$$

$$f_t = \sigma(conv_f([X_t h_{t-1}])). \tag{4.8}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \sigma(conv_g([X_t h_{t-1}])). \tag{4.9}$$

$$h_t = o_t \odot tanh(c_t). \tag{4.10}$$

In this work, ConvRNN cells are adopted to inject temporal information into detection models. Suppose that there is a one-stage detection model with N layers and that the output of nth layer is directly the input of n+1th layer where 0<=n<N. The injection of temporal information is implemented as that instead of directly feeding output of n^th layer into $n + 1^{th}$ layer, it is extracted as an input to an ConvRNN cell together as well as a state to ConvRNN cell as the temporal information and the output of ConvRNN cell is then fed into $n + 1^{th}$ layer.

4.3.3 Proposed Model

Inspired by [50], The proposed end-to-end trainable model can be split into two branches as Figure 1. The initial image and groundtruth bounding box is fed into an initial state extraction branch to extract information as the initial state to ConvRNN cells. In each of following timesteps, the frame image and previous states are fed into a modified detection model stacked with ConvRNN cells to produce outputs and states.

doi:10.6342/NTU201801036

In the beginning, when the initial image I_1 and the target bounding box B_1 is provided, an emphasis function f (e.g. a mask function) is applied on I_1 to emphasize the pixels inside target area B_1. The output of emphasis function is notated as MI_1 as follows.

$$MI_1(x) = f(x; I_1, B_1) = \begin{cases} I_1(x), & \text{if } x \in B_1. \\ 0, & \text{if } x \notin B_1. \end{cases} \tag{4.11}$$

Then MI_1 is fed into an initial state extraction branch to extract the i^{th} layer output as the initial state S_1 for ConvRNN cells.

In time-step t, the current image It and the previous state S_{t-1} are fed into a prediction branch. The prediction branch is a modified detection model with stacks of ConvRNN cells as metioned in III.B. The input of the ConvRNN cell is the output of ith layer in the original original detection structure. Suppose the original detection has the structure

$$\begin{cases} out_i = F_i(input). \\ input_{i+1} = out_i. \end{cases} \tag{4.12}$$

where F_i, $input_i$, out_i are the function, the input and the output of i^{th} layer. After the insertion of ConvRNN cell at the ith layer, the structure becomes as follows.

$$\begin{cases} out_i = F_i(input) \\ cout = ConvRNN(out_i, S_{i,t-1}). \\ input_{i+1} = cout_i. \end{cases} \tag{4.13}$$

The output features of the overall model are decoded for regression terms and passed through non-maximum suppression to produce the predicted bounding-box of the target in timestep t. If there are more than one detected bounding-boxes, the one with the highest score is selected as the prediction. If there are no detected bounding-boxes, the proposal with the highest score is selected as prediction.

doi:10.6342/NTU201801036

The initial state extraction branch and the prediction branch has the same structure as the adopted detection model except that the prediction branch has insertions of ConvRNN cells. The initial states are extracted from the output of n^{th} layer in the initial state extraction branch where a ConvRNN cell is inserted after n^{th} layer and before $n + 1^{th}$ layer in the prediction branch. In addition, the initial state extraction branch and the prediction branch do not share parameters.

4.3.4 Multi Layer Injection

ConvRNN cells can be inserted into a detection model in multiple layers. Since deeper layer contains more semantic information and shallower layer contains more location information, multi-layer insertion of ConvRNN cells can inject abundant information into detection model. Yet, when ConvRNN cells are inserted in too many layers, the training would take more time to complete and the computational cost in evaluation would increaes.

In our experiments, ConvRNN cells are inserted into one layer, three layers, and four layers. The insertion of more layers are limited by memory requirements. The performances show that the insertion of three layers has the best performance under a trade-off between the degree of difficulty in training and performance.

4.3.5 Two Stage Clip Training

In the beginning, back-propagation through time was utilized as an attemption to train the unrolled network. For a video sequence of N frames, the first frame and the initial bounding box are fed into the initial state extraction branch to extract the initial state. Then in each next run, the prediction branch is unrolled for m (m>1) frames. m video frames together with a state is fed into the unrolled network for training.

doi:10.6342/NTU201801036

Yet, when back-propagation through time was utilized, the loss did not drop down through around a week of training on GPU. These failures can be attributed to three reasons.

First, the initial state extraction branch is trained only once in the first clip batch of a sequence. However, a sequence contains dozens or even hundreds of clip batches. Thus, the initial state extract branch is not properly trained, leading to effortless training on prediction branch. Second, prediction at time-step t requires temporal information generated from previous frames and correct predictions. In the beginning of training, branches at lower time-steps make incorrect predictions and noisy states, causing noisy and vain gradients in higher time-steps. The noisy gradient may destroy the training in lower time-step, summing to zero training in prediction branch.

Due to these reasons, two stage clip training is proposed to replace back-propagation through time. In the first stage, each video is split into clips with only two frames I_1 and I_2 with B_1 as the groundtruth bounding-box in I_1. I_1 and B_1 are fed into the initial state extraction branch to produce the initial state S_1. S_1 and I_2 are fed into the prediction branch to produce predictions. The loss is computed and the gradients are back-propagated through both branches. In the second stage, each video is split into clips of random lengths in range [a, b] (e.g., a=2, b=20). For each clip, the first image and the groundtruth bounding box in the first frame are fed into the initial state extraction branch to produce the initial state. Then the initial state and the remaining video frames are fed into the unrolled model. The loss is computed and the loss is back-propagated through only prediction branches. The parameters of the initial state extraction br anch are fixed in the second stage.

The goal of the first stage is to train the initial state extraction branch to extract information from the first frame and to train the prediction branch to predict based on extracted state. The goal of the second stage is to train the prediction branch to update the state given the new frames and adapt to the updated states.

doi:10.6342/NTU201801036

Algorithm 3 The First Stage of Two Stage Clip Training

```
Require:
    $M$: the unrolled model of initial state extraction branch and one time step of prediction
    branch.
    $V$: the set of training videos including video frames and groundtruths

 1: $C_b \leftarrow \{\}$
 2: for all $v \in V$ do
 3:     $x_i$: the $i^{th}$ frame
 4:     $y_i$: the groundtruth of the $i^{th}$ frame
 5:     $L_v$: the number of frames in $v$
 6:
 7:     $C \leftarrow \{(x_t, y_t, x_{t+1}, y_{t+1}) | t = 2n, n \in [0, \lfloor \frac{L_v}{2} \rfloor])\}$
 8:     $C_b \leftarrow C_b \cup C$
 9: end for
10: $C_b \leftarrow$ shuffle($C_b$)
11:
12: for all $batch \in C_b$ do
13:     $train(M, batch)$
14: end for
```

4.4 Experiments

4.4.1 Implementation Details

The experiment is implemented in Python3.5 and Tensorflow r1.3.0 on Ubuntu 14.04 LTS, CPU Intel Core i7-6700K and a single GPU NVIDIA GeForce GTX TITAN X.

Since the unrolled model cannot fit entirely into a single GPU, the training is implemented as follows. In the forward propagation, in each run only a branch model (either initial state extraction branch or prediction branch of a specific time step) is allocated on GPU. After given the input and previous state, the network computes all the variables. These generated variables are moved from GPU memory to main memory. (The move of variables from GPU memory to main memory can be implemented simply by putting the tensors in the first parameter of Session.run in Tensorflow.) In next run, the previous state, the input and the model are again allocated on GPU. In back propagation, for the

doi:10.6342/NTU201801036

Algorithm 4 The Second Stage of Two Stage Clip Training

```
Require:
    $M_{init}$: the initial state extraction branch
    $M_{pred}$: the prediction branch
    $V$: the set of training videos including video frames and groundtruths

1: $C \leftarrow \{\}$
2: for all $v \in V$ do
3:     $x_i$: the $i^{th}$ frame
4:     $y_i$: the groundtruth of the $i^{th}$ frame
5:     $L_v$: the number of frames in $v$
6:
7:     $i \leftarrow 0$
8:     while $i < L_v - 1$ do
9:         $L_c \leftarrow$ draw a random integer from [2, 20]
10:        if $i + L_c \leq L_v$ then
11:            $L_c \leftarrow L_v - i$
12:        end if
13:        $C \leftarrow C \cup \{(x_t, y_t) | i \leq t < i + L_c\}$
14:        $i \leftarrow i + L_c$
15:    end while
16: end for
17:
18: $C_b \leftarrow \{\}$
19: for $n \in [2, 20]$ do
20:     $C_n \leftarrow \{c | length(c) = n, c \in C\}$
21:     $C_b \leftarrow C_b \cup create\_batches(C_n)$
22: end for
23: $C_b \leftarrow$ shuffle($C_b$)
24:
25: for all $batch \in C_b$ do
26:     $L_b$: the length of $batch$
27:     $M_{pred,L_b}$: the unrolled version of $M_{pred}$ for length $L_b$
28:
29:     $X_t, Y_t, X_{t+1}, Y_{t+1}, ..., X_{t+L_b-1}, Y_{t+L_b-1} \leftarrow batch$
30:
31:     $S_{init} \leftarrow$ extract_initial_state($M_{init}$, $X_t$, $Y_t$)
32:     train($M_{pred,L_b}$; $S_{init}$, $batch - \{X_t, Y_t\}$)
33: end for
```

doi:10.6342/NTU201801036

final timestep, the loss can be directly computed together with forward propagation on a single GPU run. For each next run of back propagation, the branch is allocated on GPU and the previous stored variables are loaded into GPU. Given the gradient of states of higher steps, GPU calculated the gradients correspondingly. After the calculation of gradients in a timestep is completed, the variables are dropped and the gradients of parameters are moved from GPU memory to main memory. After the gradients of all time-steps are calculated, the gradients with respect to the same parameter are averaged and applied onto parameters by an optimizer.

The pretrained detection model is the YOLO9000 [48]. The ConvRNN cells are the ConvLSTM [49]. The insertion of ConvLSTM cells occurs in the 8th, the 16th and the 28th layer which are all the last two layers before pooling layers. The experiments metioned in III.D has the insertion of one ConvLSTM cell at the 28th layer, the insertion of four ConvLSTM cells at the 2nd, the 8th, the 16th and the 28th layer. In experiments, an insertion into the 8th and the 2nd layer has been attempted. Yet, since the lower layer has larger blob size, insertion into the 8th and the 2nd layer requires too large memory for state blobs, resulting in out-of-memory error in a batch size of only two.

For the training time, the training of the first stage and the second stage in two stage clip training takes around one week respectively. For training dataset, the network is trained on OTB2015 [32, 33] and tested on VOT2016 [51]. The proposed algorithm is compared with the MDNet [1], the SRDCF [52], the C-COT [53], the KCF [54], and the DSST [**DSST**].

4.4.2 Results

Figure 2 and Figure 3 are the prediction results on a girl sequence and a basketball player sequence respectively. From the figure, the proposed tracker can successfully track

doi:10.6342/NTU201801036

the targets under versatile situations involving occlusion, image blurring, existence of similar objects and target deformation.

Table 4.1: The comparance of computational cost of the RDisp and other algorithms.

Tasks	Elapsed Time
one frame prediction of the MD-Net [1]	$\sim$ 1 second
long-term finetuning of the MD-Net [1]	$\sim$10 seconds
short-term finetuning of the MD-Net [1]	$\sim$0.5 second
one frame prediction of the Faster-MDNet [46]	$\sim$0.8 second
one frame prediction of the C-COT [53]	$\sim$1 second
one frame prediction of the SRDCF [52]	$\sim$1 second
one frame prediction of the DSST [**DSST**]	$\sim$0.15 second
one frame prediction of the KCF [54]	$\sim$0.09 second
one frame prediction of the RDisp	$\sim$**0.04 second**

Table 1 is the comparance of prediction time with the MD-Net, the Faster-MDNet, C-COT, SRDCF, KCF, DSST, and the proposed model. The testing time in one timestep in the proposed model is around 20 times faster than one testing time step in the Faster-MDNet, 25 times faster than prediction in MD-Net, 12 times faster than short-term fine-tuning in MD-Net, 250 times faster than long-term finetuning in MD-Net.

Figure 4 is the comparison of the proposed model with other tracking algorithms on expected overlap with respect to different sequence length. From the figure, it leaves room for the proposed trackers to improve in accuracy and robustness.

4.4.3 Conclusions

A visual tracking algorithm based on the ConvLSTM and the YOLO9000 and a two-stage clip training to replace backpropagation through time to train the proposed model are proposed. An experiment with training in OTB2015 and testing in VOT2016 shows consistence of tracker under different circumstances such as existence of similar objects, occlusions, inside-object deformation and shows a low computational cost of around 25

doi:10.6342/NTU201801036

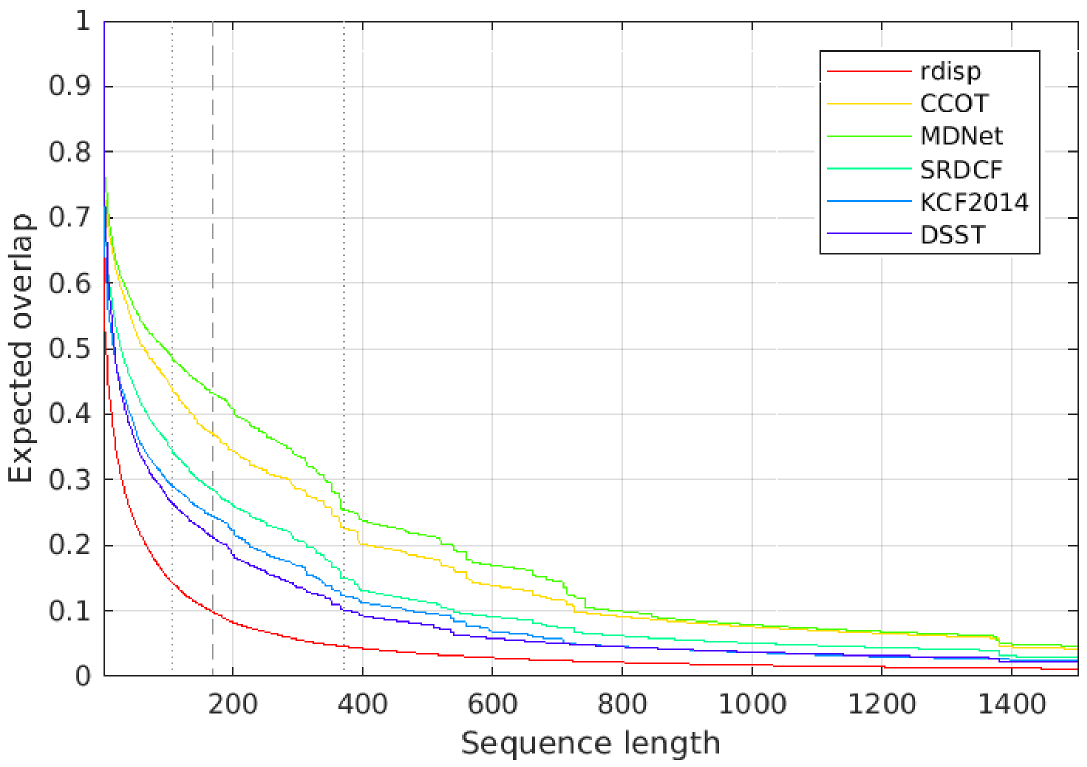
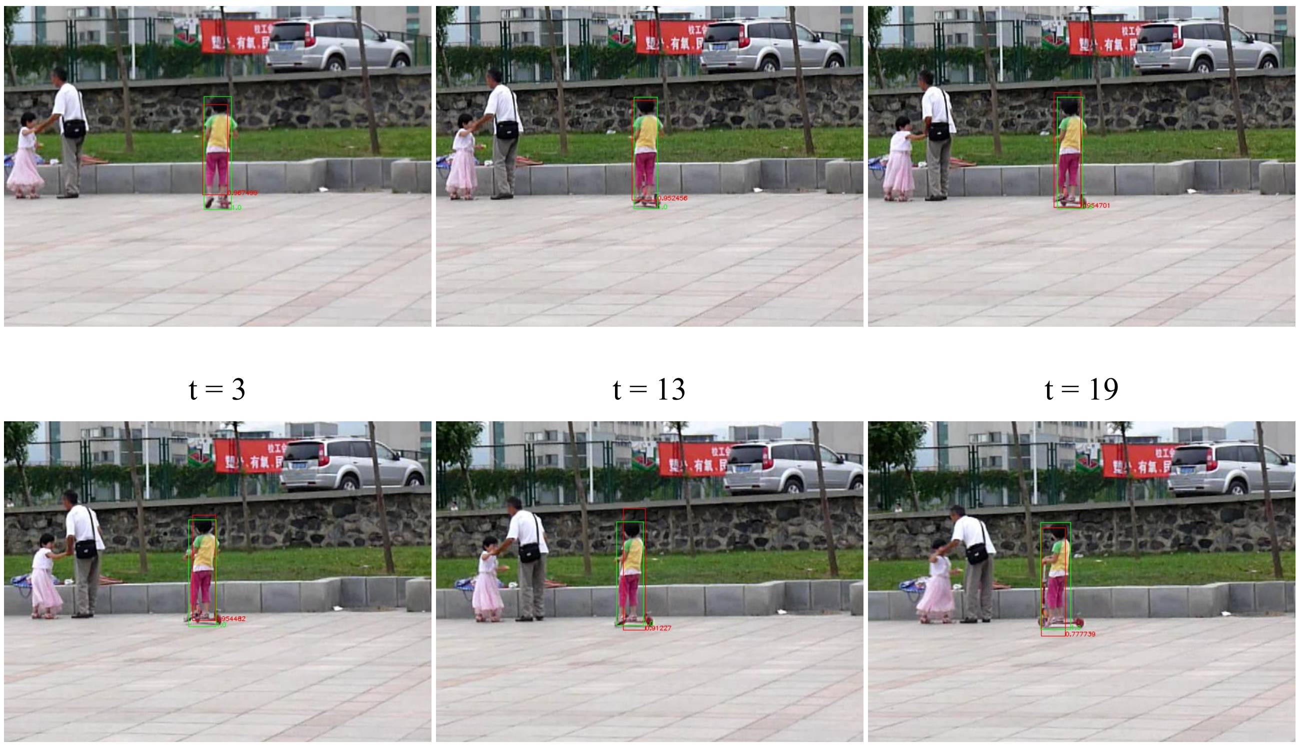
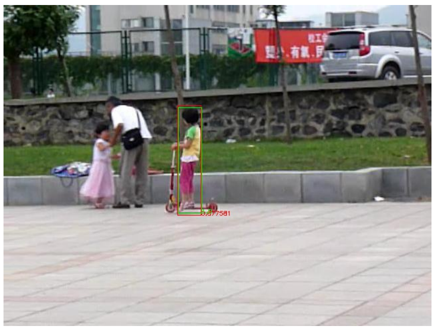


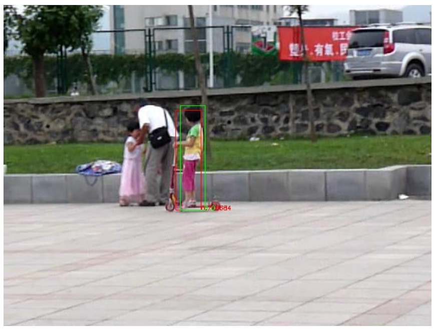
Figure 4.2: Comparances of expected overalp with respect to sequence length of RDisp with other algorithms on VOT2016 dataset.

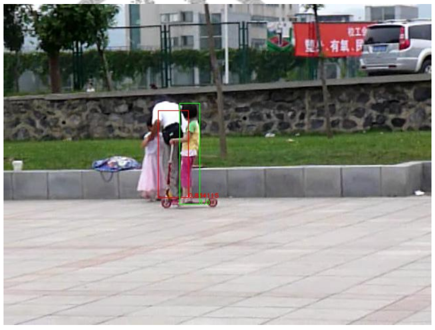
fps benefiting from shared features and anchor boxes in one-stage detection model.

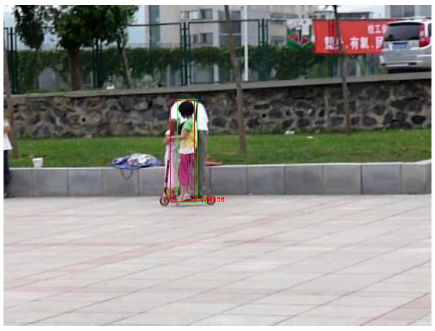


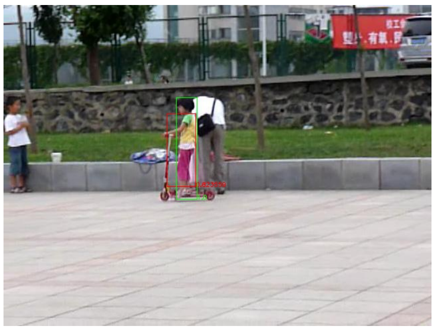
doi:10.6342/NTU201801036

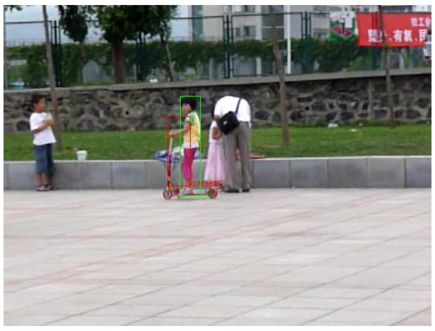

t = 43

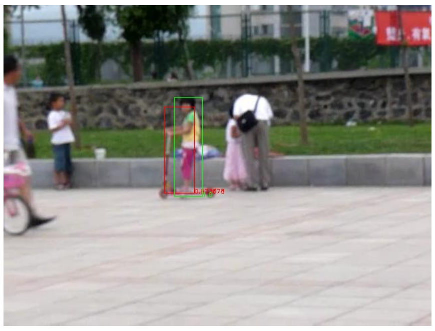

t = 49

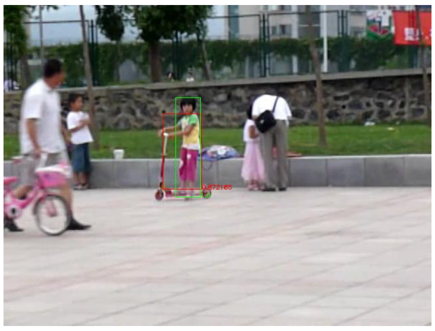

t = 55

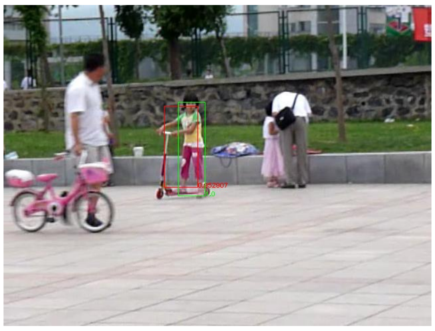

t = 61

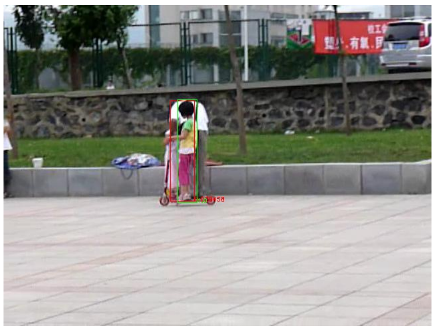

t = 67

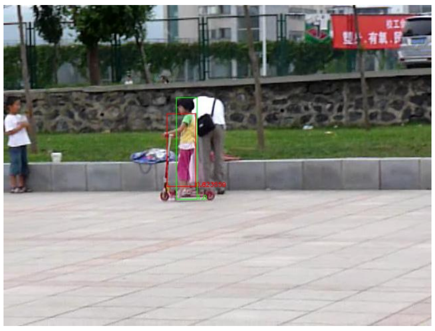

t = 73

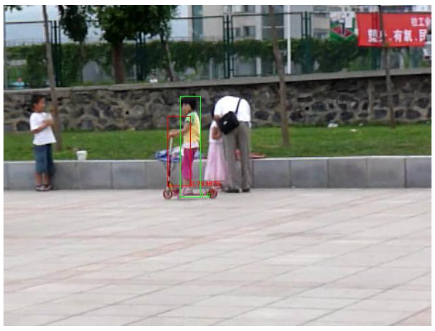

t = 79


t = 85

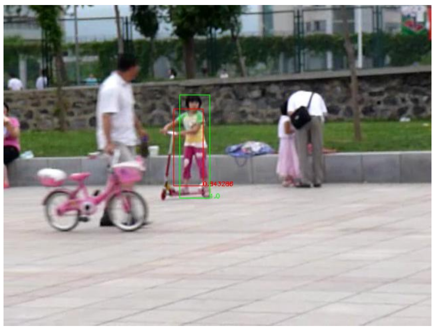

t = 91


t = 61

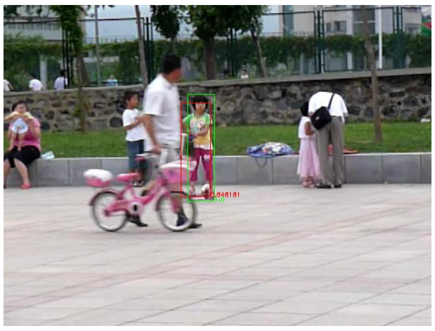

t = 67


t = 73

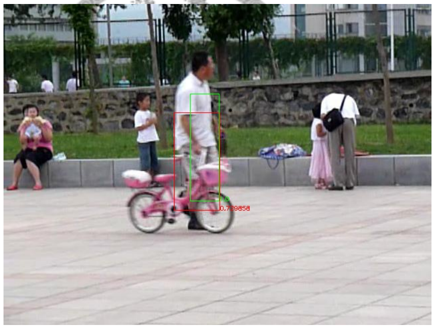
doi:10.6342/NTU201801036



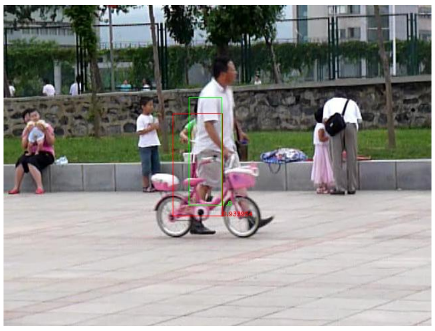
t = 96



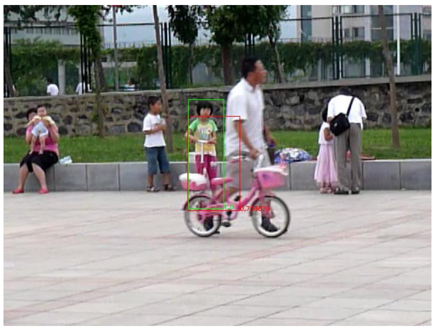
t = 103



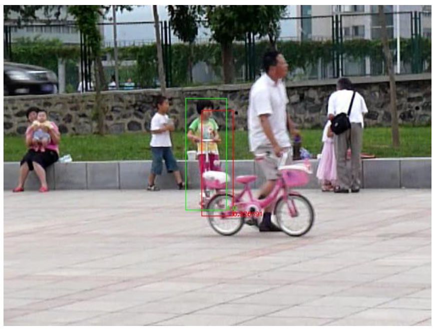
t = 109



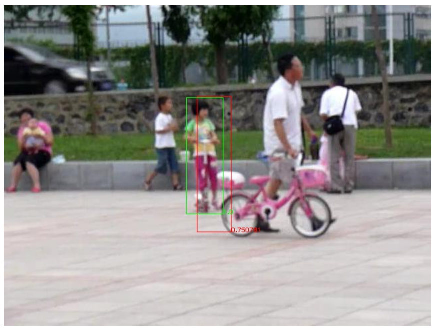
t = 115



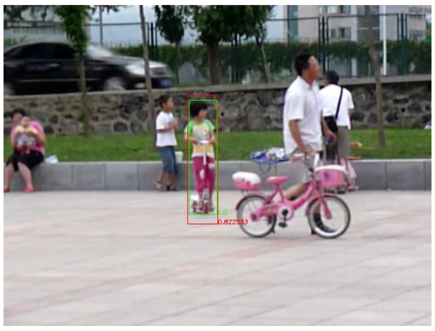
t = 121



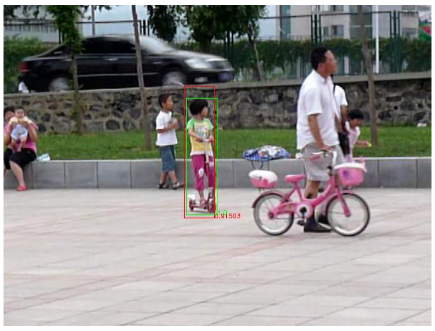
t = 127



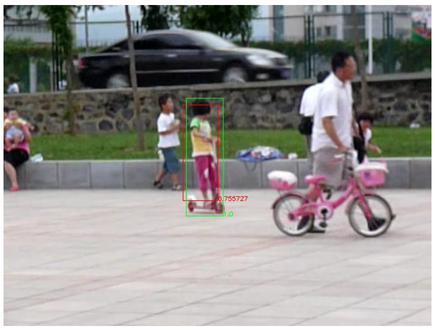
t = 133



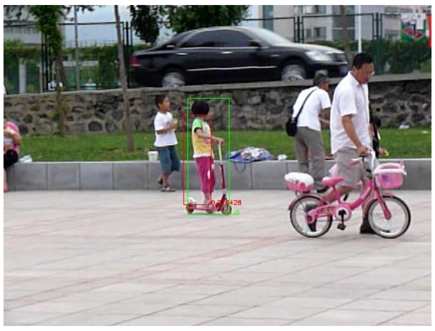
t = 139



t = 145



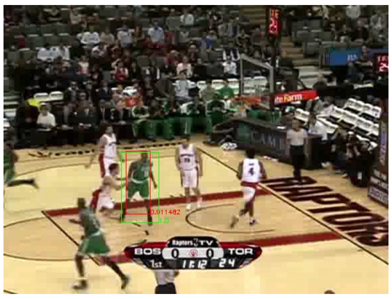
t = 151



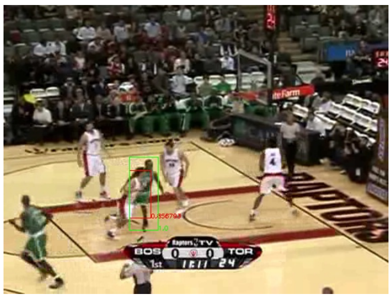
t = 157

Table 4.2: Results of RDisp on OTB2015 girl. (green box: groundtruth, red box: prediction).

doi:10.6342/NTU201801036

t = 1

t = 4

t = 7

t = 10

t = 13

t = 16

t = 19

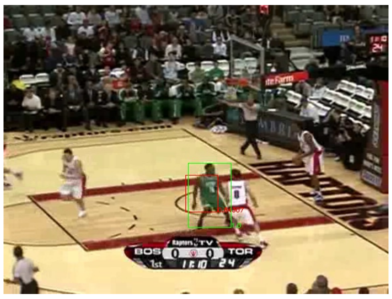
t = 22

t = 25

t = 28

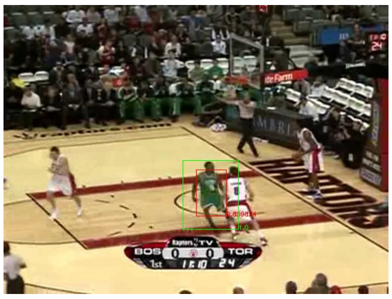
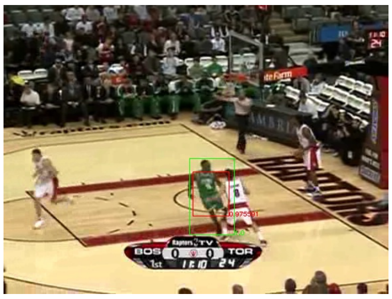
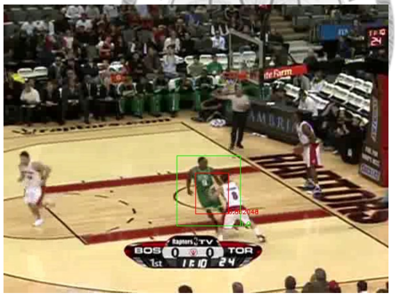
t = 31

t = 34

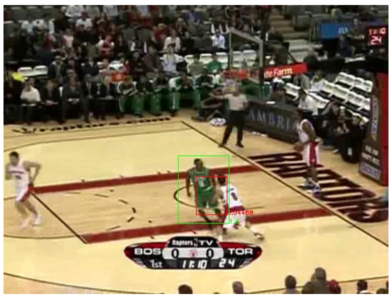
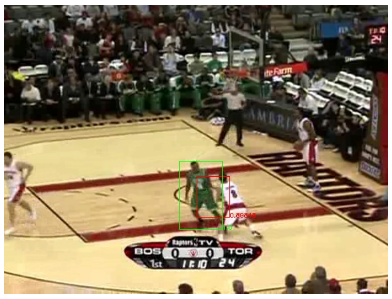
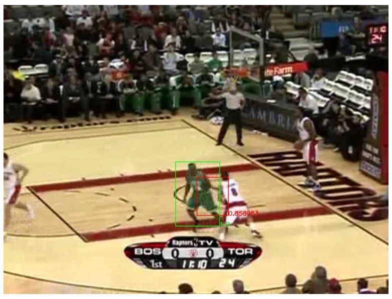


doi:10.6342/NTU201801036

t = 37 t = 41 t = 43

t = 46 t = 50 t = 53

t = 56 t = 58 t = 59

Table 4.3: Results of RDisp on VOT2016 basketball. (green box: groundtruth, red box: prediction).

doi:10.6342/NTU201801036

Chapter 5 Proposed Dolphin Detection

In this chapter, works on dolphin detection are introduce. This chapter is organized as follows. In section 5.1, the motivations on dolphin detection and identification are presented. In section 5.2, the provided dolphin datasets on detection and identification are introduced. In section 5.3, the overall system architecture of dolphin detection and identification is presented. In section 5.4, the works on dolphin detection are introduced.

5.1 Motivations

In the field of marine conservation, dolphin images are rare, precious, and vital resources. To obtain an image of dolphin, a group of team stays on the sea for weeks, hand-holds heavy and huge high-resolution cameras, and wait for appearance of dolphins to take shots to capture images of dolphins. After the raw data is obtained, it further requires tremendous human resources to post-process the images such as cropping dolphin patches, identifying information of individual dolphins to form a database for future research.

In recent years in the fields of computer vision, many detection and classification algorithms with high accuracy under the setting of general classes have been proposed, which benefits from release large image dataset such as MS COCO [4], ImageNet [45] and deep convolutional neural networks.

This works aims to alleviate dearly-won post-processing of dolphin images by automatic detection and identification based on recent developments in computer vision, which

doi:10.6342/NTU201801036

amends inefficient devotion into repeated works and helps the researchers focus on more innovative contributions.

5.2 Dolphin Datasets

The original dolphin dataset includes a set of original raw images captured by cameras and a set of cropped dolphin patches. Each type of annotations is given for a subset of the dolphin patch set. The annotation types consist of the names, the stages, and the angles of the dolphin relative to the photographer. For example, let the set of patches be S. Then, the set of patches annotated with the angle S_{angle} is a subset of S. That is, $S_{angle} \subset S$.

The details of the classes in each type of annotations are listed in Table 5.1. The details of the names of dolphins are sealed due to confidentiality.

Table 5.1: The classes in each dolphin annotation type.

Annotation Type	Classes
stage	Moltted Stage(少年), Speckled Stage (青年), Spotted Adult Stage (壯年), Unspotted Adult Stage (老年)
angle	+x, -x, +y, -y
name	-

In addition, the detection datasets includes two datasets. The first dataset includes raw images with at most one dolphin patch in each image. The second dataset includes raw images with all dolphin patches in each image. To be simplified, the first dataset and the second dataset are called ”single-dolphin-detection-dataset” and ”multi-dolphins-detection-dataset” respectively in the following sections.

doi:10.6342/NTU201801036

5.3 System Architecture

This project aims to utilize automatic detection and classification algorithms to alleviate the heavy burden of post-processing on dolphin images to help researchers focus on innovative works. The overall system architecture includes a detection block followed by a classification block. The dolphin block is responsible for the detection of each dolphin patch. The classification block is responsible for classifying each dolphin patch for different annotation types.

There are two reasons for this architecture. First, in the future there may be additional annotation types in classification. Using two block architecture, the detection model does not require fine-tuning and the performance will not be altered due to the addition of new annotation types. Second, the provided dataset contains patches with incomplete annotations. The meaning of "incomplete" is that a patch may be annotated with no labels, only part of labels, or all labels. Suppose S is the set of all patches, S_{name}, S_{stage}, and S_{angle} are the subsets of patches annotated with name labels, stage labels, and angle labels respectively, then

$$S - S_{stage} - S_{name} - S_{angle} \neq \emptyset. \tag{5.1}$$

$$S_a \cup S_b - S_a \cap S_b \neq \emptyset, S_a, S_b \in [S_{stage}, S_{name}, S_{angle}]. \tag{5.2}$$

Due to the incompleteness of the annotation, if the system model incorporates both detection and classification (e.g. a yolo9000 model [48] with multiple branches in the last layers for multiple annotation types), in back-propagation of the classification model, the branches related to the lost annotations would not be updated, thus creating an imbalance on the classification model among annotation types. If the annotations are complete for every patches, the model may learn versatile features to represent all annotation types.

doi:10.6342/NTU201801036

However, since the annotations are incomplete, an imbalance among datasets would potentially render the model preferable toward a specific dataset and downgrades the performances of detection or classification of unpreferable annotation types. Therefore, due to the reasons mentioned above, the overall architecture is chosen as a detection block followed by a classification block.

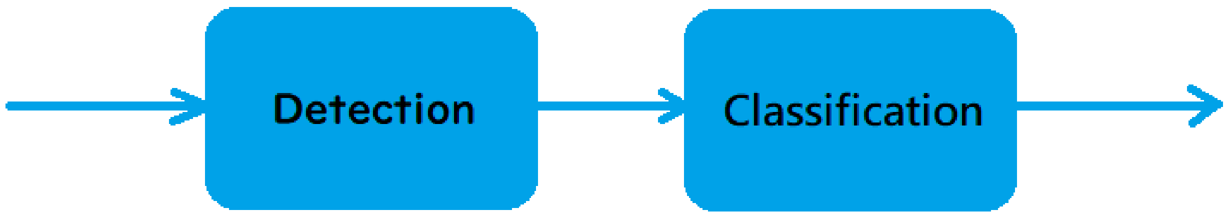


Figure 5.1: The two block system architecture.

5.4 Detection

5.4.1 Patch Matching

The detection dataset includes raw images and cropped dolphin patches. To train a detection model, the bounding-box with respect to each dolphin patch on the raw image is required. Hence, the dolphin patch should be matched on the original image to search the corresponding bounding-box.

OpenCV [55] is a cross-platform, open-source computer vision library under BSD license including a bunch of computer vision algorithms. OpenCV provides a set of template matching algorithms which are adopted in this work to match a dolphin patch on a raw image. The template matching function in OpenCV is fed with the patch and the image. The output of the function is a score map with the same size as the input image. The value at pixel (x, y) in the score map is the match score between the input patch and the patch starting at the pixel with the same of the input patch. Suppose the input patch is

doi:10.6342/NTU201801036

S with size (h_w, w_s) and the input image is I with size (h, w). The function is as follows.

$$O = match(S, I). \tag{5.3}$$

$$O(r, c) = match\_score(S, I\_p(r, c)). \tag{5.4}$$

$$I_p(r, c) = I(r : r + h_s, c : c + w_s), 0 \leq r < h, 0 \leq c < w. \tag{5.5}$$

OpenCV provides six methods to calculate the match score of two patch of the same size. The follows introduces the mathematical formulation of these six methods. R is the match score of the input patch T and the input image I.

1. TM_SQDIFF

$$R(x, y) = \sum_{x',y'} (T(x', y') - I(x + x', y + y'))^2. \tag{5.6}$$

2. TM_SQDIFF_NORMED

$$R(x, y) = \frac{\sum_{x',y'} (T(x', y') - I(x + x', y + y'))^2}{\sqrt{\sum_{x',y'} T(x', y')^2 \cdot \sum_{x',y'} I(x + x', y + y')^2}}. \tag{5.7}$$

3. TM_CCORR

$$R(x, y) = \sum_{x',y'} (T(x', y') \cdot I(x + x', y + y'))^2. \tag{5.8}$$

4. TM_CCORR_NORMED

$$R(x, y) = \frac{\sum_{x',y'} (T(x', y') \cdot I(x + x', y + y'))^2}{\sqrt{\sum_{x',y'} T(x', y')^2 \cdot \sum_{x',y'} I(x + x', y + y')^2}}. \tag{5.9}$$

doi:10.6342/NTU201801036

5. TM_CCOEFF

$$R(x,y) = \sum_{x',y'} (T'(x',y') \cdot I'(x+x',y+y'))^2. \tag{5.10}$$

$$\begin{cases} T'(x,y) = T(x,y) - \frac{1}{wh} \sum_{x'',y''} T(x'',y''). \\ I'(x+x',y+y') = I(x+x',y+y') - \frac{1}{wh} \sum_{x'',y''} I(x+x'',y+y''). \end{cases} \tag{5.11}$$

6. TM_CCOEFF_NORMED

$$R(x,y) = \frac{\sum_{x',y'} (T'(x',y') \cdot I'(x+x',y+y'))^2}{\sqrt{\sum_{x',y'} T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x',y+y')^2}}. \tag{5.12}$$

TM_SQDIFF is the sum of square difference of pixels. TM_CCORR is the correlation of pixels. TM_CCOEFF is the correlation of unbiased pixels. TM_SQDIFF_NORMED, TM_CCORR_NORMED, and TM_CCOEFF_NORMED are corresponding normalized versions.

Since the image and the cropped patch are both encoded by JPEG and JPEG is a lossy compression, errors would be introduced on encoded images. Due to the error caused by JPEG, difference based methods are not considered. To alleviate the effects of potential bias and illumination variation caused by lossy compression, TM_COEFF_NORMED is adopted.

In the single-dolphin-detection-dataset, the resulting groundtruth bounding-box is given as the one starting at the pixel with the highest match score with the same size of the patch. That is, suppose the input patch size is (w_s, h_s), the output match score map is R, and the

doi:10.6342/NTU201801036

output bounding-box is $B = (x_1, y_1, x_2, y_2), 0 \leq x_1, x_2 < w_s, 0 \leq y_1, y_2, < h_s$. Then,

$$B = (\hat{x}, \hat{y}, \hat{x} + w_s, \hat{y} + h_s). \tag{5.13}$$

$$\hat{x}, \hat{y} = argmax_{x,y} R(x, y). \tag{5.14}$$

In the multi-dolphins-detection-dataset, to ensure the property of the groundtruth, a policy based on the match score is set to drop images including potentially incorrect or damaged patches. Suppose an image I contains several dolphin patches $\{S_1, S_2, ..., S_N\}$. In template match of a patch $S_i, i \in [1, N]$ with the image, if the score map contains more than one pixels or no pixel with score > 0.999, the image is dropped. In our experiments, less than one percent of images in the dataset are dropped.

5.4.2 Detection

After the patch matching is completed, the datasets contains images and bounding-boxes of dolphin patches. In detection, two stage model is chosen for two reasons. The first reason is that the detection accuracy of two stage model has relatively better performances than one-stage model due to the second stage refinement. The second reason is that in the future the detection model and the classification model would be merged into a single end-to-end model for accuracy and complexity considerations. In addition, there may be additional annotation types in classification. Under two stage model, it is simpler to only fine-tune the second stage to add new annotation types instead of re-training the whole model in one stage model. Two stage model would be easier to add more labels for future research. The second reason is the primary reason of the choice of two stage model. To be concrete, FRCN [5] is adopted as the detection model.

In the beginning, single-dolphin-detection-dataset is utilized to train a FRCN model. The final model performs with man average precision 0.883 on testing dataset. The pre-

doi:10.6342/NTU201801036

cision and recall curve is as Figure 5.2.

In single-dolphin-detection-dataset, some dolphin are not enclosed by groundtruth boxes and treated as background. In the setting of training FRCN, there would be problems in training region proposal network in single-dolphin-detection-dataset. Since the negative samples are sampled from anchor boxes whose IoU with the groundtruth boxes is smaller than a threshold (e.g. 0.3), it's possible to sample the anchor boxes on dolphins without groundtruth bounding-box as the negative samples. As for training the second stage classifier, since the negative samples are sampled from anchor boxes surrounding the groundtruth bounding boxes with IoU in a range [a, b] (i.e. a = 0.1, b=0.5), it has less probability to sample other dolphin patches as negative samples.

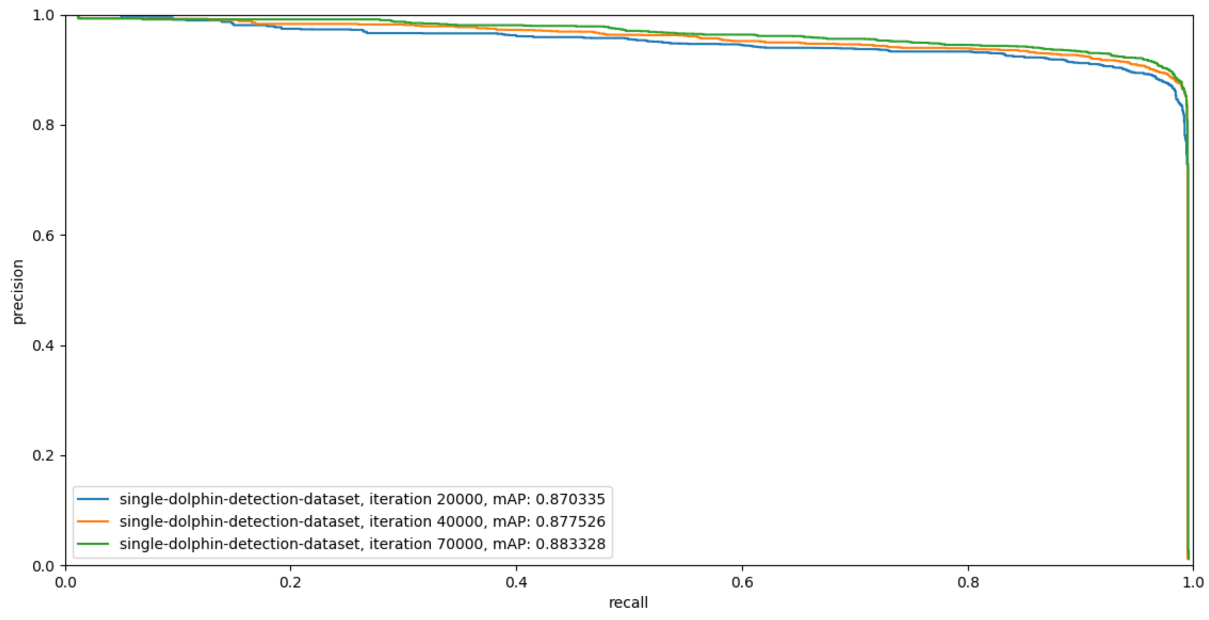


Figure 5.2: The precision and recall curve of models trained on single-dolphin-detection-dataset with iteration 20000, 40000, and 70000.

Due to these reasons, the multi-dolphins-detection-dataset is utilized to train another FRCN model. On the testing dataset of multi-dolphins-detection-dataset, the model trained on single-dolphin-detection-dataset and the model trained on multi-dolphins-detection-dataset has mean average precision 0.71.0 and 0.888 respectively. The precision and re-

doi:10.6342/NTU201801036

call curve of both models on testing dataset of multi-dolphin-detection-dataset is as Figure 5.3. From the figure, it can be shown that with additions with complete dolphin patches on all images, the precision and recall both increase evidently with a large gap.

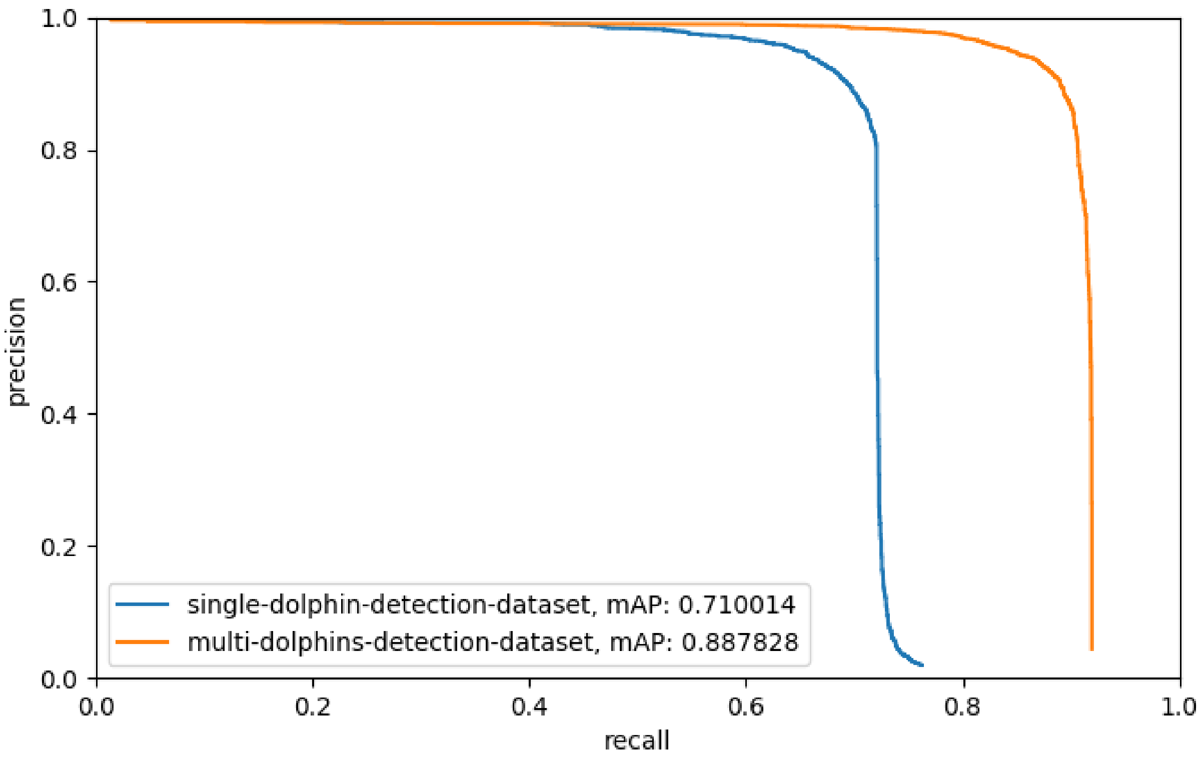


Figure 5.3: The precision and recall curve of models trained on single-dolphin-detection-dataset and multi-dolphins-detection-dataset.

doi:10.6342/NTU201801036

Chapter 6 Proposed Dolphin Identification

In this chapter, the works on dolphin identification is introduced. This chapter is organized as follows. In section 6.1, the outline of proposed dolphin identification is illustrated. In section 6.2, the based classification model in the dolphin identification work, the DenseNet121, is introduced. In section 6.3, the works on classification of angles and stages are presented. In section 6.4, the works on classification of names are presented. In section 6.5, the results of classification of names on masked images is presented. Finally in section 6.6, the desktop application on our proposed dolphin detection and identification system is introduced.

6.1 Introduction

In this section, the outline of the proposed dolphin identification is illustrated. In this work, dolphin identification involves three classification tasks: classification of stages, angles and names.

In the classification of stages, since the judgement of growing stages is based on colors of dolphin skin and the cropped patch has a tight bounding-box on the dolphin, methods based histogram of colors are utilized. In addition, convolutional neural network based on low-level and high-level features, the DenseNeTt121 [11], is also utilized. In total,

doi:10.6342/NTU201801036

neural network with histogram of pixels, neural network with three histograms of each RGB channel, and the DenseNet121 are utilized. With simple histogram features with neural network, testing accuracy can reach higher than 91%. The DenseNet121 reaches the highest testing accuracy around 98%. For the classification of angles, the DenseNet121 is adopted and the testing accuracy reaches around 96%.

For classification of names, in the beginning, the DenseNet121 is utilized and the accuracy on testing dataset can reach around 80%. However, when the photographs newly captured with the same hardwares are tested on this model, it performs poorly close to accuracy with random predictions. To observe why the model performs outstandingly on testing dataset yet poorly on captured images, a technique in [6] is utilized. This method calculate the saliency of an image based on the gradient of the ground-truth score with respect to image pixels. The result shows that the image has saliency not only on the dolphin but also on the sea surface. It's deduced that the model may classify based on the simple rough color of the sea background instead of the details of dolphins since many images under the same name are taken at the same date in consecutive frames. A direct split of training and testing data by ratio would create similar distribution on training and testing for the sea surface but not details of dolphins.

To verify the deduction, an experiment on splitting training and testing data by different dates is executed and demonstrates overfitting with training accuracy above 95% and testing accuracy close to random predictions under the same settings of training model and optimizer. Hence, to solve the problem, a new block to detect the dolphin mask to remove the pixels of sea surface is proposed.

Due to the lack of the dolphin mask and the simplicity of sea background, the saliency detection algorithms are utilized to detect the mask. In direct implementation of existing saliency detection algorithms on our dolphin patches, problems are found that the detection results lie on only parts of the dolphin instead of the whole dolphin. With further

doi:10.6342/NTU201801036

experiments, it shows that the reasons are the mismatches of the settings of the saliency detection and our goals. With this insight, a simple solution of enlarging the cropping bounding-box demonstrates an extreme result. In addition, to complement the drawbacks of different saliency algorithms, it's proposed to use the ensemble of different saliency algorithms with different bases to produce the final saliency mask. [TODO: Quantitative and qualitative results of saliency detection are included.] [TODO: Different ensemble methods]

Finally, the masked image are utilized to training the DenseNet121 model and performs around 85.53% testing accuracy. The overall structure is as Figure 6.2. In addtion, an GUI application based on our proposed system architecture is built upon Python Tkinter.

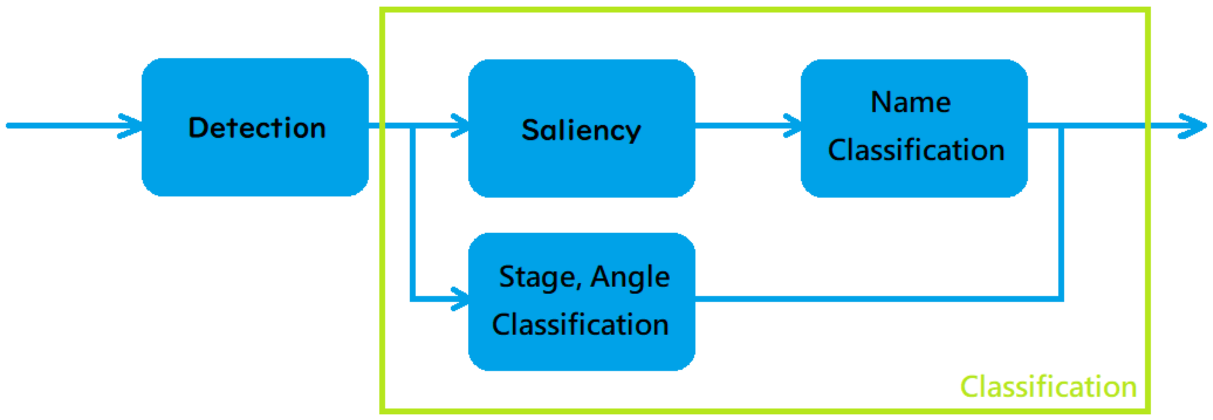


Figure 6.1: The block system architecture with details of classification block.

6.2 DenseNet

In 2016, Kaiming He et al. proposed the ResNet [56]. In the ResNet, it's demonstrated that by adding the input of a convolution layer to the output, the gradient vanishment problem, an obstacle of building deep neural networks, would be solved since the gradient can flow through skip connections.

doi:10.6342/NTU201801036

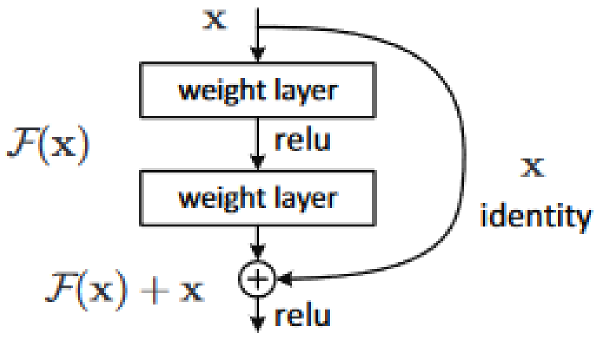

x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 6.2: The skip connection in ResNet [56] .

Akin to the ResNet, Gao Huang et al. [11] proposed to replace summation with concatenation. Suppose the feature in the l^{th} convolutional layer is x_l, then in ResNet, the output feature is the summation as follows.

$$x_l = Conv(x_{l-1}) + x_{l-1}. \tag{6.1}$$

In the Densenet, the output feature is the concatenation as follows.

$$x_l = [Conv(x_{l-1}), x_{l-1}]. \tag{6.2}$$

Thus, not only the gradient vanishment problem can be solved, the data flow is also not impeded by the summation. In the ResNet, convolutional layers finds the residual of the input features and update the input features; in the DenseNet, convolutional layers explores new features and concatenates both features. The output features of the DenseNet contains both low-level and locational features and high-level and semantic features. One problem of concatenation is that the number of channels will always increase, creating redundant channels and high spatial cost. Gao Huang et al. [11] proposed to use transition layer, a 1x1 convolution layer, to reduce the channel number and render a more compact model.

For the performance results of the DenseNet in [11], it reports lower error rates with less parameters on multiple dataset compared with several state-of-the-art algorithms in-

doi:10.6342/NTU201801036

cluding ResNet. In addition, Gao Huang et al. [11] proposed four model structures, DenseNet-121, DenseNet-169, Densenet-201, and DenseNet264 with different depths. Yet, due to the classes in our classification not as many as 1000 classes in the ImageNet [45], the prevention of overfitting, and the cosidieration of memory efficiency, the model with the least depth DenseNet121 is adopted in our classification block.

6.3 Classification of stages and angles

6.3.1 Observations

According to expert knowledge, the growing stages are judged by the color of dolphin skins. For the dolphin in our dataset, in the birth, the color of the dolphin will be purely black. When the dolphin grows from child to the elder, the dolphin skin would turn from black to pink white and then purely white. Since the dolphin comprise most pixels in the cropped patch, the color of the dolphin skin can be represented by the probability distribution of colors of pixels in the cropped patch.

6.3.2 Methods and experiments

Two features are utilized to calculate the probability distribution of colors of pixels in the cropped patch to represent the color of dolphin skin. The first feature is the histogram of pixels. The second features are three histogram of each RGB channel.

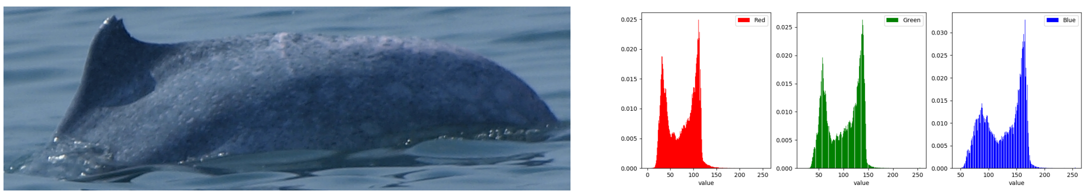


Figure 6.3: The example of the color histogram of a moltted-stage dolphin.

doi:10.6342/NTU201801036

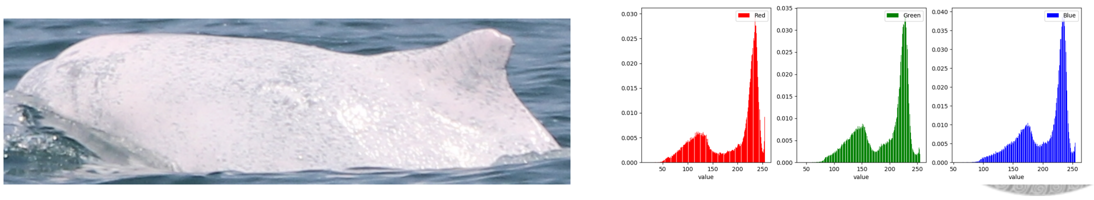


Figure 6.4: The example of the color histogram of a spotted-stage dolphin.

Histogram of pixels

Suppose $I(x, y) = [R(x, y), G(x, y), B(x, y)]$ is the pixel of image I at location (x, y) composed of three channels $R(x, y)$, $G(x, y)$, and $B(x, y)$ and the values in $R(x, y)$, $G(x, y)$, and $B(x, y)$ are integers in [0, 255]. Then the histogram of pixel is as equation 6.3.

$$H_{pixel}(r, g, b; I) = \frac{card(\{p = (r, g, b) \in I\})}{card(I)} \tag{6.3}$$

where I is the set of image pixels and $card(\cdot)$ is the cardinality of a set.

Yet, since the value of each red, blue and green channel is an integer in [0, 255], histogram values have $256^3 \approx 2 \times 10^7$ possibilities which requires too large spatial complexity. Thus, the pixel value is further quantized by 16 as equation 6.4. The histogram values of quantized pixel are reduced to $(\frac{256}{3})^3 = 4096$ possibilities.

$$H_{quantized,pixel}(qr, qg, qb; I) = \frac{card(\{p|p = (r, g, b) \in I, (\lfloor \frac{r}{16} \rfloor, \lfloor \frac{g}{16} \rfloor, \lfloor \frac{b}{16} \rfloor) = (qr, qg, qb)\})}{card(I)} \tag{6.4}$$

Histograms of color channels

For the histograms of color channels, RGB color channel is utilized. For each of red, green, and blue channel, an histogram is calculated. The concatenation of three histograms

doi:10.6342/NTU201801036

are the final output feature as equation 6.5.

$$H_{red}(r; I) = \frac{card(R(p) = r, p \in I)}{card(I)} \tag{6.5}$$

$$H_{green}(g; I) = \frac{card(G(p) = g, p \in I)}{card(I)} \tag{6.6}$$

$$H_{blue}(b; I) = \frac{card(B(p) = b, p \in I)}{card(I)} \tag{6.7}$$

$$H_{color}(I) = [H_{red}(I), H_{green}(I), H_{blue}(I)] \tag{6.8}$$

where $R(\cdot)$, $G(\cdot)$, and $B(\cdot)$ are the red, green and blue channel of a pixel.

Experiments

In experiments, three methods are utilized. First, histogram of pixel feature followed by neural network is carried out. Second, histograms of color channels followed by neural network is carried out. Third, the raw patch input with DenseNet121 is carried out. The structure of the neural network is as Table 6.2 and 6.1. As in Table 6.3, DenseNet121 demonstrates the best performance around 98.15%.

Table 6.1: The structure of neural network with histograms of colors input.

Layers	Output Size
Input Feature	768
Dense, 768 x 256 | BatchNorm | LeakyReLU	256
Dense, 256 x 56 | BatchNorm | LeakyReLU	56
Dense, 56 x 16 | BatchNorm | LeakyReLU	16
Dense, 16 x 56 | BatchNorm | LeakyReLU	56
Dense, 56 x 16 | BatchNorm | LeakyReLU	16
Dense, 16 x 4 | BatchNorm | LeakyReLU	4
Softmax	4

doi:10.6342/NTU201801036

Table 6.2: The structure of neural network with histogram of pixel input.

Layers	Output Size
Input Feature	4096
Dense, 4096 x 1024 | BatchNorm | LeakyReLU	1024
Dense, 1024 x 256 | BatchNorm | LeakyReLU	256
Dense, 256 x 56 | BatchNorm | LeakyReLU	56
Dense, 56 x 16 | BatchNorm | LeakyReLU	16
Dense, 16 x 56 | BatchNorm | LeakyReLU	56
Dense, 56 x 16 | BatchNorm | LeakyReLU	16
Dense, 16 x 4 | BatchNorm | LeakyReLU	4
Softmax	4

Table 6.3: The results of the classification of stages.

Methods	Accuracy
Histograms of colors + Neural Network	0.8209
Histogram of pixels + Neural Network	0.9129
Image Input + DenseNet121	**0.9815**

6.3.3 Classification of angles

For the classification of angles, there are four types of angles, +x, -x, +y, and -y according to the direction of the head as in Figure 6.6 and 6.5. Following the experiences in classification of stages, DenseNet121 is again adopted for classification of angles. The results shows an extreme testing accuracy 95.73%.

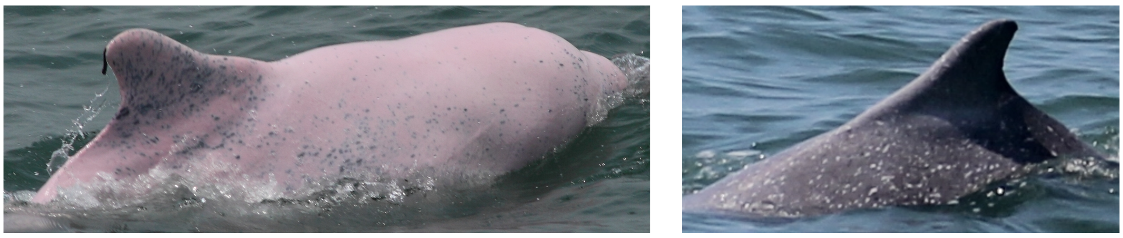


Figure 6.5: The examples of dolphin patches in +x and -x angles.

doi:10.6342/NTU201801036

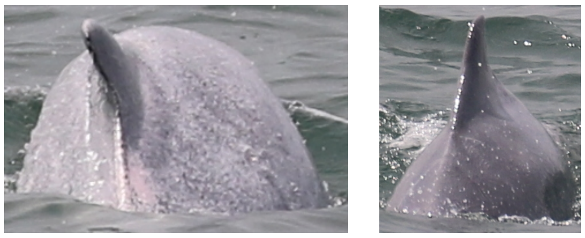


Figure 6.6: The examples of dolphin patches in +y and -y angles.

6.4 Classification of names

6.4.1 DenseNet121

In classification of names, in the beginning, DenseNet121 is utilized. The dolphin patches in the dataset are randomly split into training and testing dataset with training data ratio 0.85. With data augmentation of rotations and color jittering, DenseNet121 is trained with 45 epochs under Adam optimizer with learning rate 1e-3. The performance on testing dataset shows 79.8% accuracy.

After the training on training dataset and the evaluation on testing dataset is completed, the model is further tested on recently captured patches. However, the performance is near that of random predictions. The reason may be a mismatch of the distribution of the tested dataset and the dataset for training and evaluation.

6.4.2 Gradient-based saliency map

To further explore the reason why the model performs differently on newly captured patched and testing dataset , a technique in a technique in [6] is utilized. This method calculates the saliency of an image based on the gradient of the ground-truth score with respect to image pixels.

Suppose $I(i,j)$ is the pixel of an input image I composed of three RGB channels $R(i,j)$, $G(i,j)$, and $B(i,j)$, the output score of DenseNet121 is S and the groundtruth of

doi:10.6342/NTU201801036

the input image is y. Then this method produce the gradients of the output score with respect to all image pixels as the maximum of the gradient of output score of the groundtruth with respect to the three channels as equation 6.9.

$$G(i,j) = \max(\left|\frac{\partial S(y)}{\partial R(i,j)}\right|, \left|\frac{\partial S(y)}{\partial G(i,j)}\right|, \left|\frac{\partial S(y)}{\partial B(i,j)}\right|) \tag{6.9}$$

The output G is the gradient-based saliency map. If a pixel has a higher value in G, the variation of this pixel value creates larger variation to the score of the ground-truth. Thus, this pixel has larger influence to the output score and comprise the saliency of the image focused by the trained neural network.

Figure 6.7 is gradient-based saliency map of the patches in the dataset on the trained model. In the gradient-based saliency map, the image has saliency not only on the dolphin but also on the sea surfaces. Hence, the model may focus on the simple background of sea surfaces instead of the dolphin. It's deduced that the model may classify based on the simple rough color of the sea background instead of the details of dolphins since many images under the same name are taken at the same date in consecutive frames. A direct split of training and testing data by ratio would create similar distribution on training and testing dataset for the sea surface co-occuring with details of dolphins. Since the sea surfaces are much easier to be fit than subtle details of dolphins, the model fit the sea surfaces for predictions of classification instead of details of dolphins. Therefore, when newly captured images are tested, the model focus on the difference of sea surfaces, leading to a poor performance in comparison with groundtruth dolphin labels annotated by dolphin experts focusing on dolphin details.

To verify this argument, a further experiment is carried out. We intend to split the dataset such that the training dataset and the testing dataset has similar distributions of dolphin details but different distributions of sea surfaces. If the model trained using this

doi:10.6342/NTU201801036

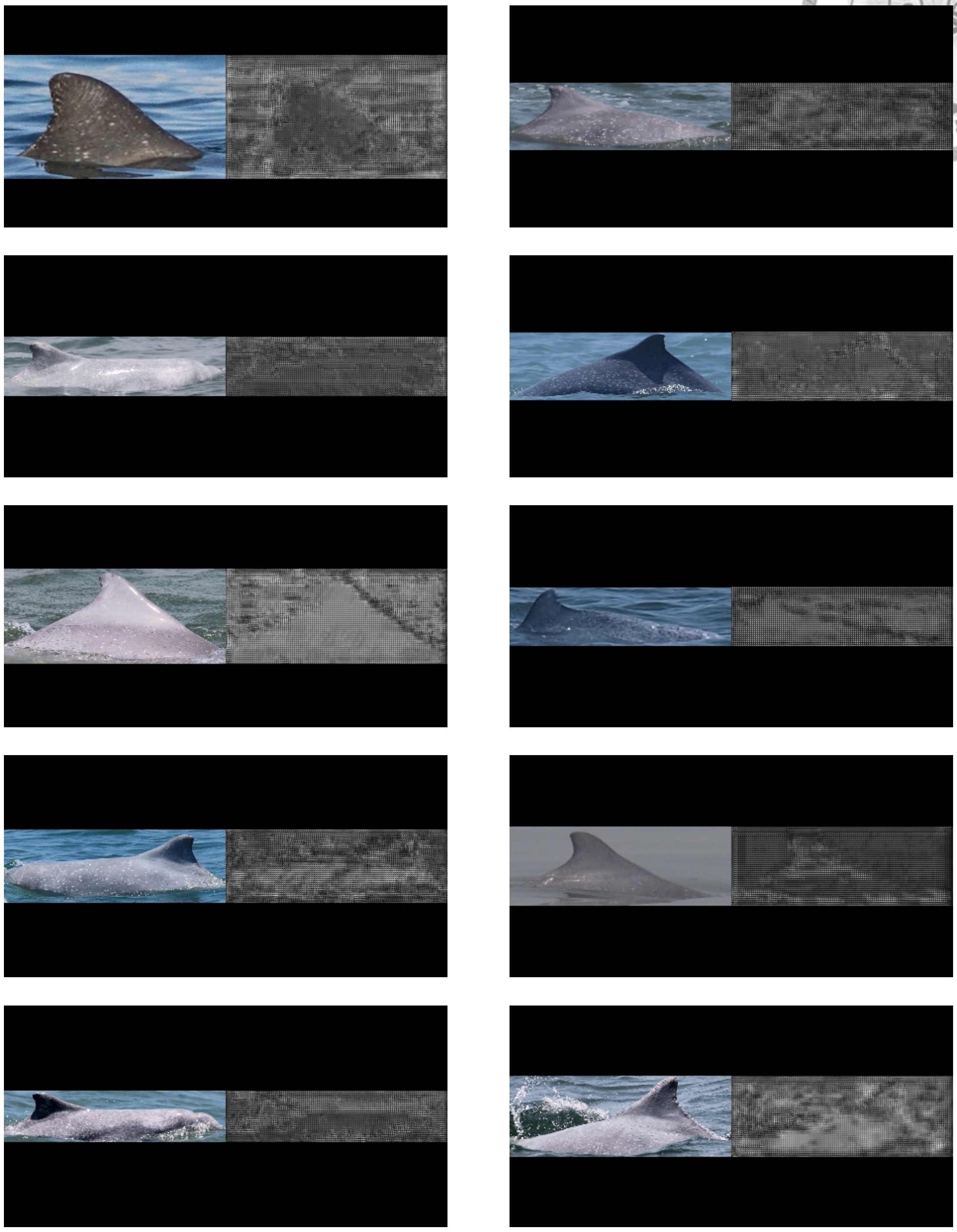


Figure 6.7: The gradient-based saliency map of the dolphin patches on the initial name-classification model. The whiter the pixel, the higher value of the gradient.

Table 6.4: The results of the experiments on dataset split by dates.

Dataset	Training Accuracy	Testing Accuracy
Random split dataset with train, test ratio 85:15	97.6%	79.8%
Split dataset by dates	**98.0%**	**4.6%**

doi:10.6342/NTU201801036

training dataset has similar performance on the testing dataset as the previous randomly split dataset. It means the model trained under randomly split dataset does not focus on the sea surfaces for the classification since the model performs equivalently whether the distribution of the sea surfaces in the training dataset is similar to the distribution of sea surfaces in the testing dataset. On the other hand, if the trained model overfits much more severely when the distributions of sea surfaces are different in training and testing dataset than when the distributions of sea surfaces are similar in training and testing dataset, it means the model fits the sea surfaces for classification so that the performance on the testing dataset with different distributions of sea surfaces is poor. Since the sea surfaces of images taken at different dates have explicitly different appearance, if images of the same labels have different dates in training and testing dataset, the distribution of sea surfaces are different in training and testing dataset. Thus, the dataset is split into to training and testing dataset on the rule that all the patches taken at the same date will be put in only one of the training dataset or the testing dataset. The date of every patch in the training dataset was different from the date of any patch in the testing dataset. Thus, the distributions of the sea surfaces are different in the training dataset and the testing dataset.

Under the same settings of training, if the trained model has comparative testing accuracy to the training accuracy, the model does not focus on the sea surface for classification but the details of the dolphin itself. On the contrary, if the trained model has extremely poor performance, the model focus on the sea surfaces since the model fits on the distribution of sea surfaces in training and fails on testing dataset which has different distribution of sea surface from the training dataset but the same distribution of dolphins as the training dataset. The result is as Table 6.4. The testing accuracy is 4.6%. Hence, the initially trained model actually classifies based on the explicit appearance of sea surfaces instead of the details of dolphins.

doi:10.6342/NTU201801036

6.4.3 Saliency mask on dolphin

To overcome the interference of sea surfaces, an intuitive solution is to multiply the image with a mask on the dolphin to remove the pixels of the sea surfaces. Based on the idea, first related works on semantic segmentation are surveyed. However, most general class semantic segmentation algorithms requires a set of images with mask groundtruths for training. Yet, in the beginning, the provided dataset does not include dolphin masks. In addition, annotations of groundtruth masks are labor-intensive and cannot be produced in a short time. Therefore, semantic segmentation algorithms are not considered.

With further observations into the dataset, an interesting characteristic of our dolphin patches invoke a new idea for the solution. Since the dolphin always appears on the sea, most of the background of the dolphin patch are simple sea surfaces. Thus, based on the simple background assumption, the dolphins in cropped patches are the saliency. Hence, saliency detection algorithms are resorted to mask out the dolphin.

In the beginning, four saliency detection algorithms the NLDF [7], the SODM [8], the GBVS [9], and the Saliency-HDCT [10] are utilized. The first two algorithms [7, 8] are proposed in CVPR 2017 and are based on deep-learning models. The last two algorithms [9, 10] are well-known rule-based algorithms. Figure 6.9, 6.10, 6.11, and 6.12 demonstrate the saliency maps computed by these four algorithms on the patches in Figure 6.8. From the figure, the deep-learning based algorithms can predict results with sharp edges and high confidences, yet lose some parts of dolphins. The rule-based algorithms predict more blurred edges and sometimes lose some parts of dolphins, too. Either of the four algorithms can produces saliency on all cases. A common failure among four algorithms is that in some cases the results contain only parts of the dolphins. This common failure on four algorithms based on different mathematical methods invokes an idea toward the inappropriateness of the data or a mismatch of our goal with the setting of saliency detection

doi:10.6342/NTU201801036

instead of an interior problem of these saliency algorithms.

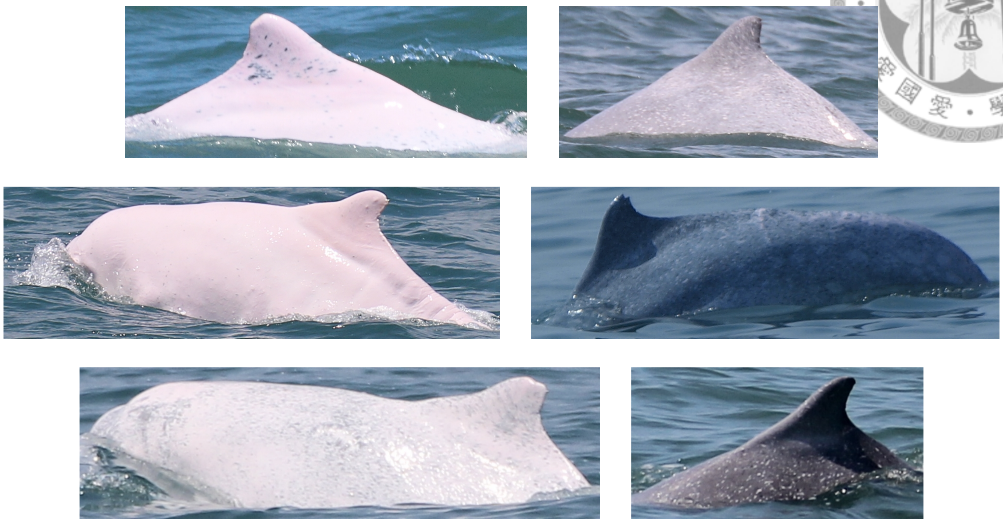


Figure 6.8: The original example dolphin patches on saliency algorithms.

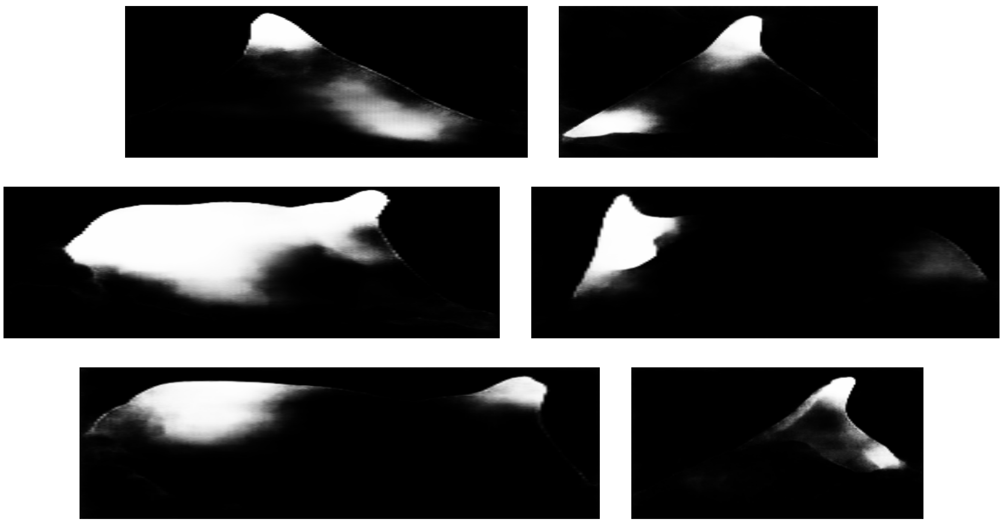


Figure 6.9: The saliency maps of example dolphin patches computed by the NLDF.

Since the cropped patch has a tight bounding-box on the dolphin, the dolphin itself comprise most pixels in a patch. Thus, in some cases, the saliency in the patch lies on parts of dolphin instead of the whole dolphin. Addition of more sea background can solve the problem. Hence, the solution is that in calculation of saliency, the patch is first cropped by twice the original bounding box to include more sea surface pixels. Suppose the original patch is cropped from (x_1, y_1, x_2, y_2) from an image with size (w, h). The enlarged

doi:10.6342/NTU201801036

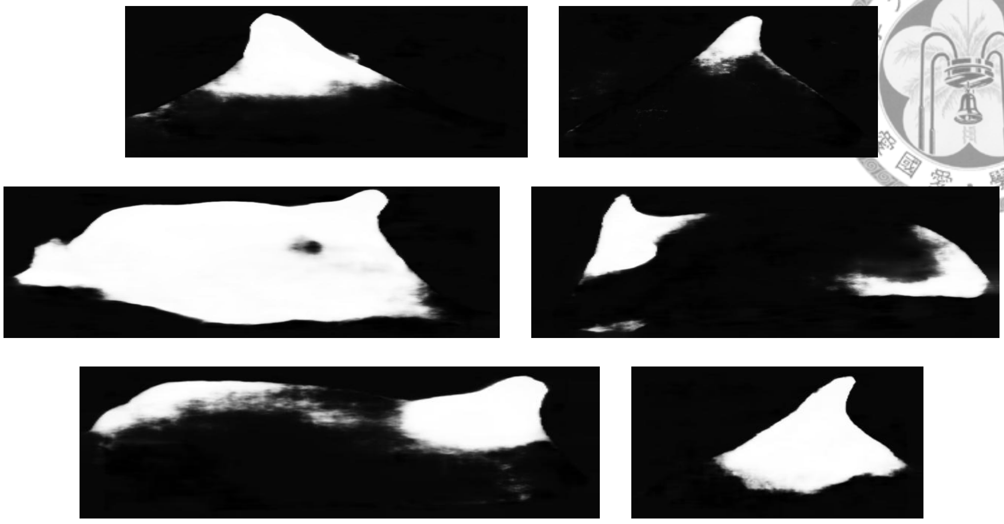


Figure 6.10: The saliency maps of example dolphin patches computed by the SODM.

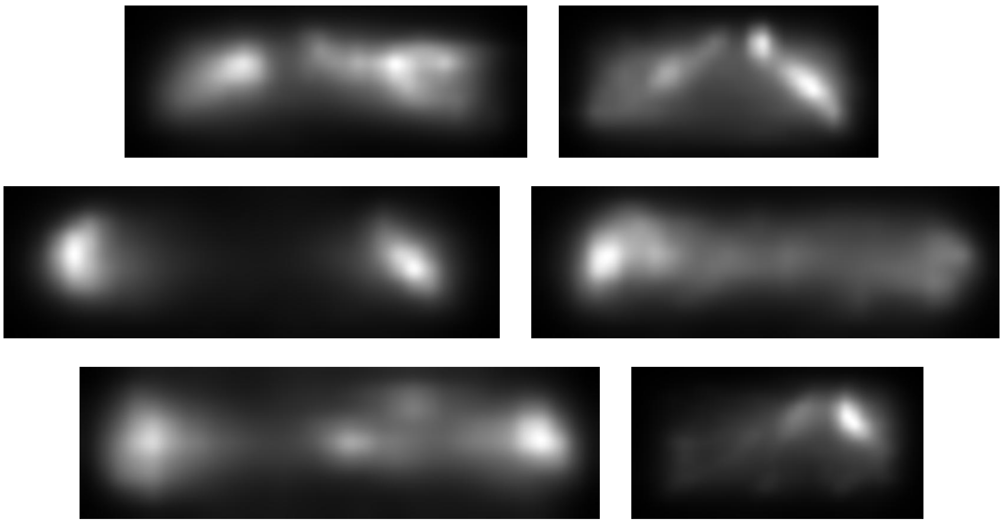


Figure 6.11: The saliency maps of example dolphin patches computed by the GBVS.

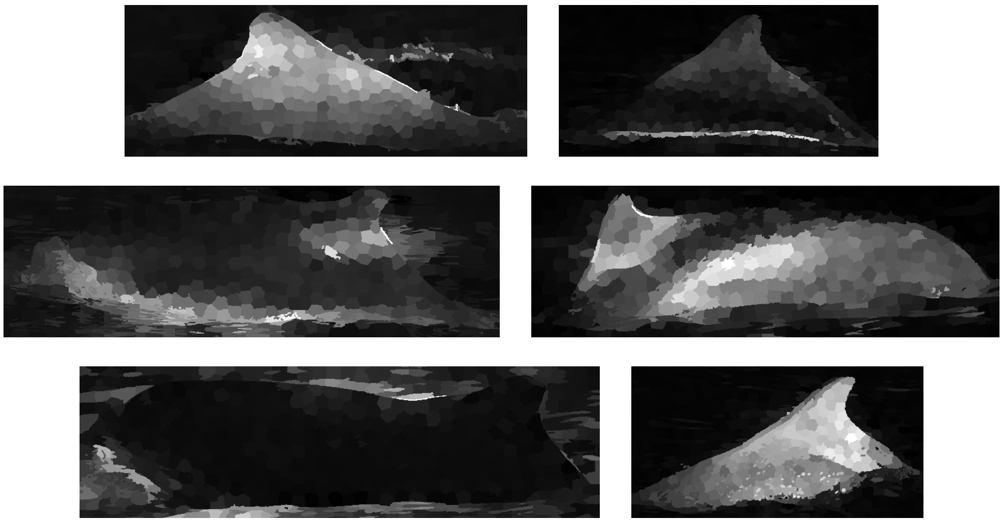


Figure 6.12: The saliency maps of example dolphin patches computed by the Saliency-HDCT.

doi:10.6342/NTU201801036

bounding box $(x_{1,twice}, y_{1,twice}, x_{2,twice}, y_{2,twice})$is computed as equations below.

$$w_p = x_2 - x_1. \tag{6.10}$$

$$h_p = y_2 - y_1. \tag{6.11}$$

$$x_{1,twice} = \max(0, x_1 - \left\lfloor \frac{w_p}{2} \right\rfloor). \tag{6.12}$$

$$y_{1,twice} = \max(0, y_1 - \left\lfloor \frac{h_p}{2} \right\rfloor). \tag{6.13}$$

$$x_{2,twice} = \min(w - 1, x_2 + \left\lfloor \frac{w_p}{2} \right\rfloor). \tag{6.14}$$

$$y_{2,twice} = \min(h - 1, y_2 + \left\lfloor \frac{h_p}{2} \right\rfloor). \tag{6.15}$$

After the saliency map of the patch of the enlarged bounding box is computed, the mask is produced by cropping the area corresponding to the original bounding box. Figure 6.14, 6.15, 6.16, and 6.17 are example results of saliency maps of patches in Figure 6.13 computed under twice enlarged bounding box in four algorithms. It demonstrates satisfactory qualitative results.

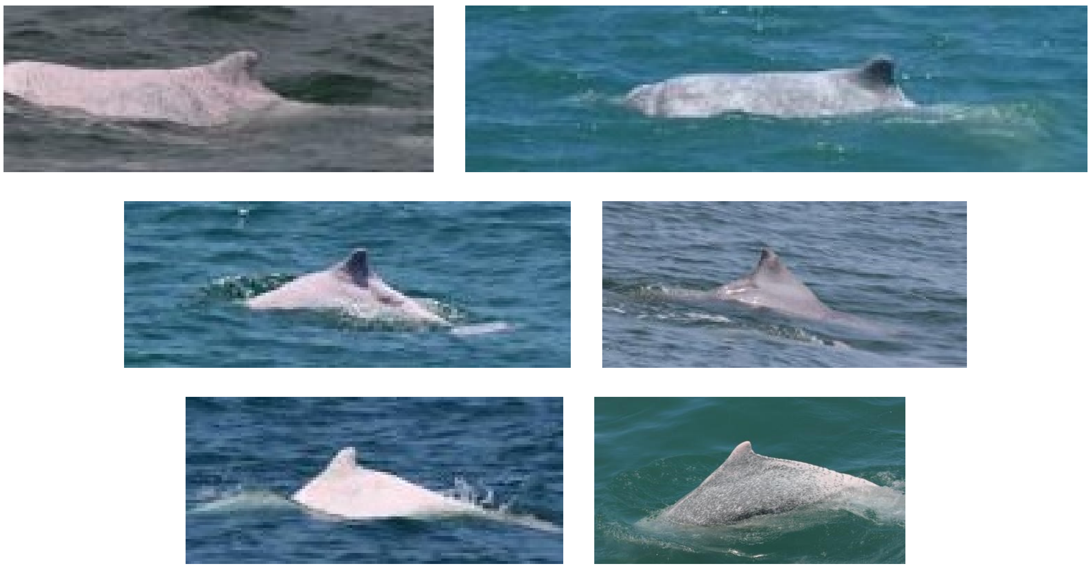


Figure 6.13: The original example dolphin patches of enlarged bounding boxes on saliency algorithms.

In addition, an ensemble on these four algorithms produce the final saliency map. An

doi:10.6342/NTU201801036

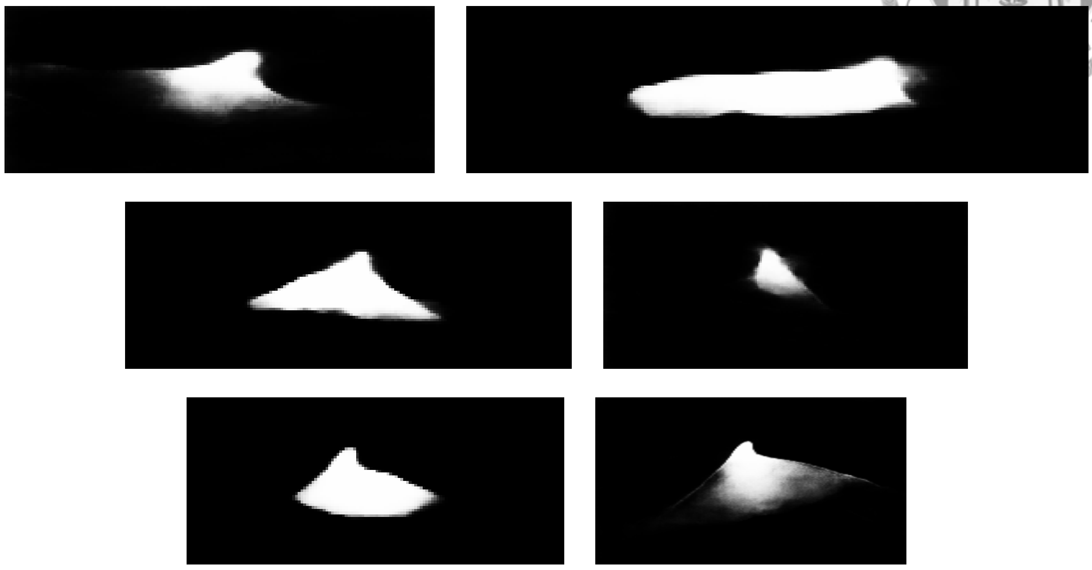


Figure 6.14: The saliency maps of example dolphin patches of enlarged bounding boxes computed by the NLDF.

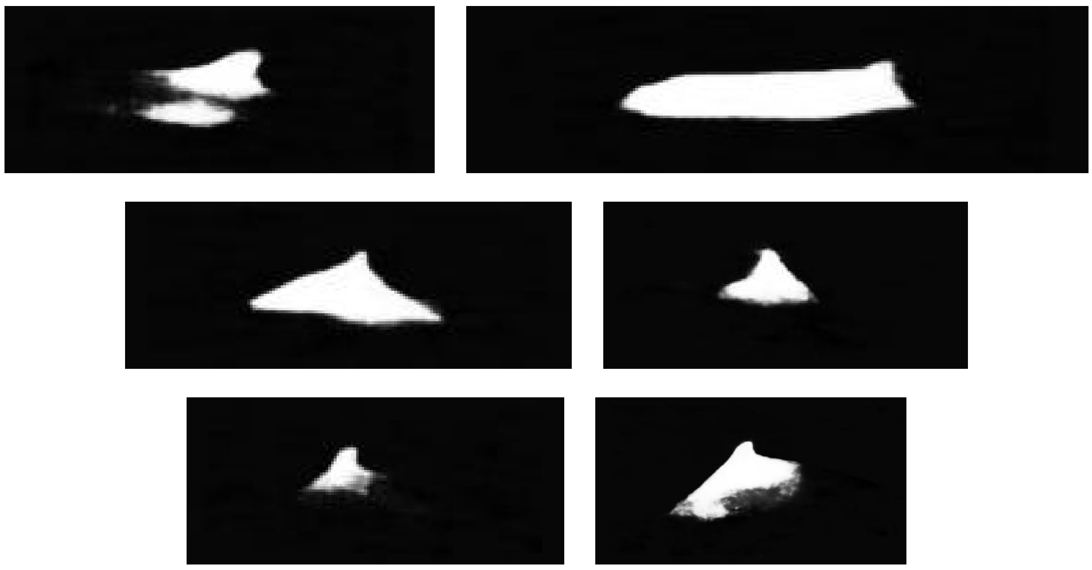


Figure 6.15: The saliency maps of example dolphin patches of enlarged bounding boxes computed by the SODM.

doi:10.6342/NTU201801036

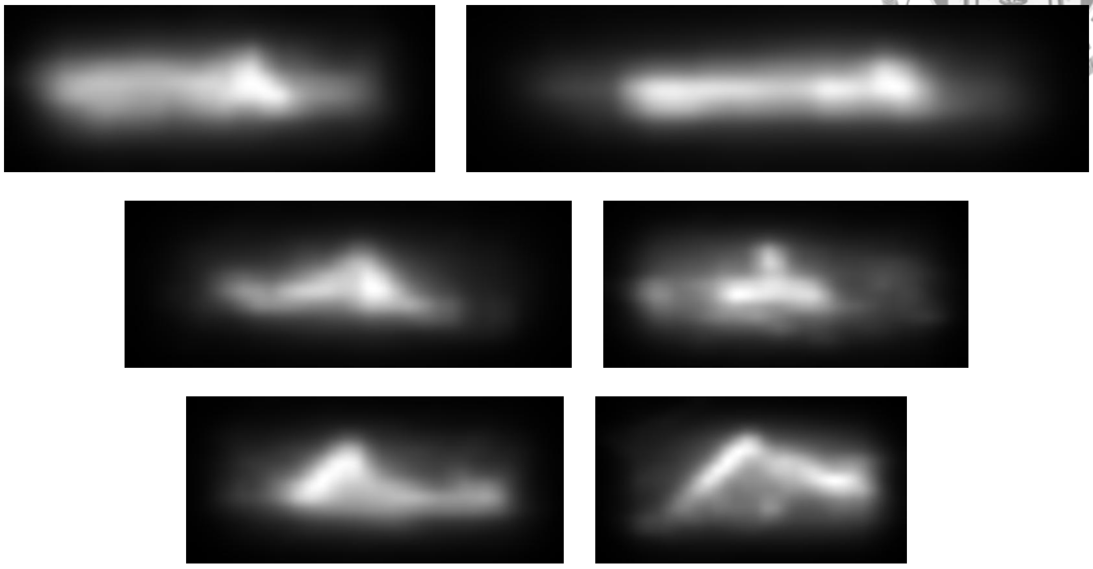


Figure 6.16: The saliency maps of example dolphin patches of enlarged bounding boxes computed by the GBVS.

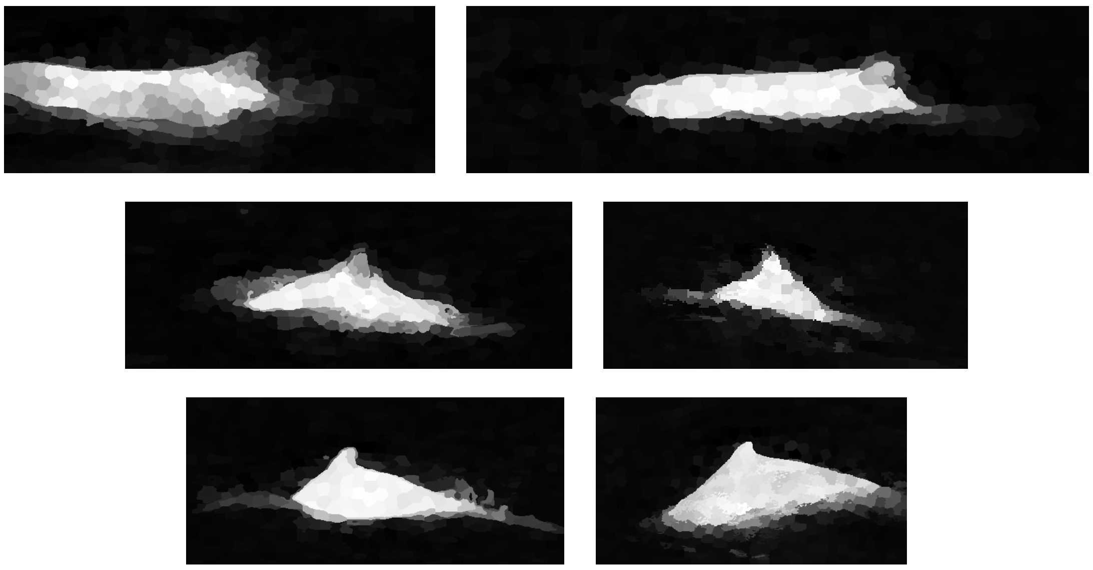


Figure 6.17: The saliency maps of example dolphin patches of enlarged bounding boxes computed by the Saliency-HDCT.

doi:10.6342/NTU201801036

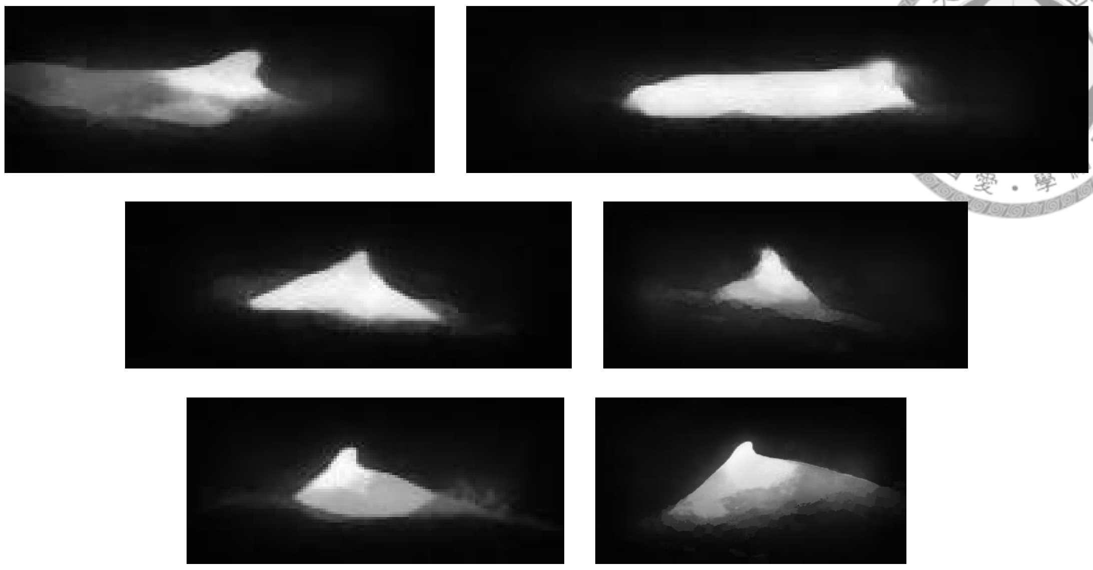


Figure 6.18: The ensemble of saliency maps of example dolphin patches of enlarged bounding boxes.

average of saliency maps of all algorithms is utilized as the ensemble method. After the saliency is produce, an soft threshold method as equation 6.16 is adopted to produce a sharper mask with floating value in [0, 1]. Suppose the computed saliency s have values in range [0, 1] and the mask is m. The soft threshold method is as follows.

$$m(i,j) = \begin{cases} s(i,j), \text{if } s(i,j) > 0.3 \\ (s(i,j)+0.5)e^{-10(s(i,j)-0.3)^2}, \text{if } 0.1 < s(i,j) \leq 0.3 \\ 0, \text{if } s(i,j) \leq 0.1 \end{cases} \tag{6.16}$$

Finally, the original patch is multiplied with mask to produce the masked patch. Then, these masked patches form a dataset to train classification models. Figure 6.18 and 6.19 are the example results of the ensemble saliency maps and the masked images. Figure 6.20 is the details of the architecture of the saliency block.

doi:10.6342/NTU201801036

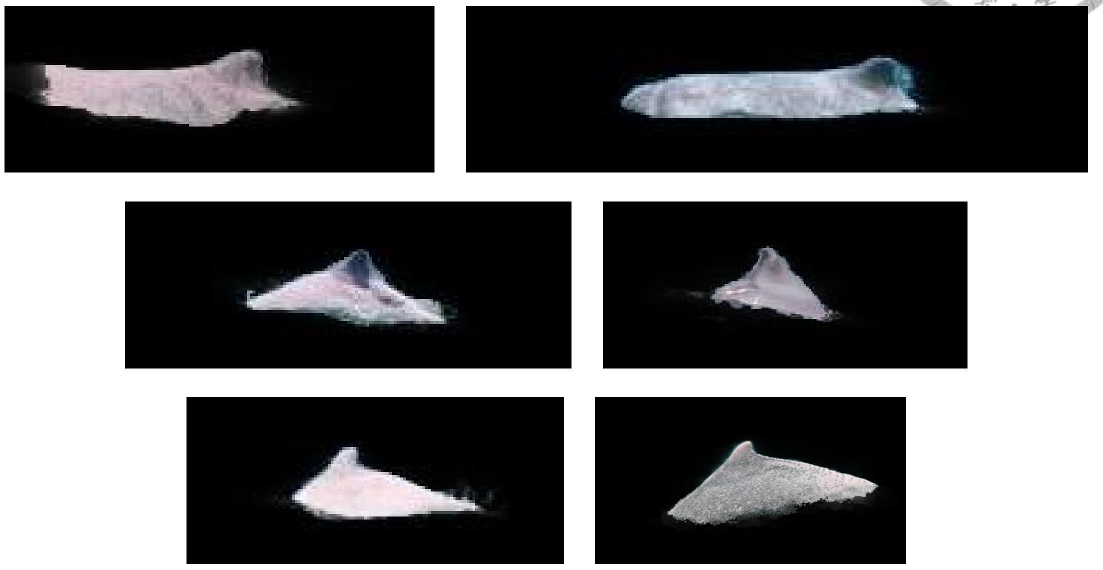


Figure 6.19: The masked images of example dolphin patches.

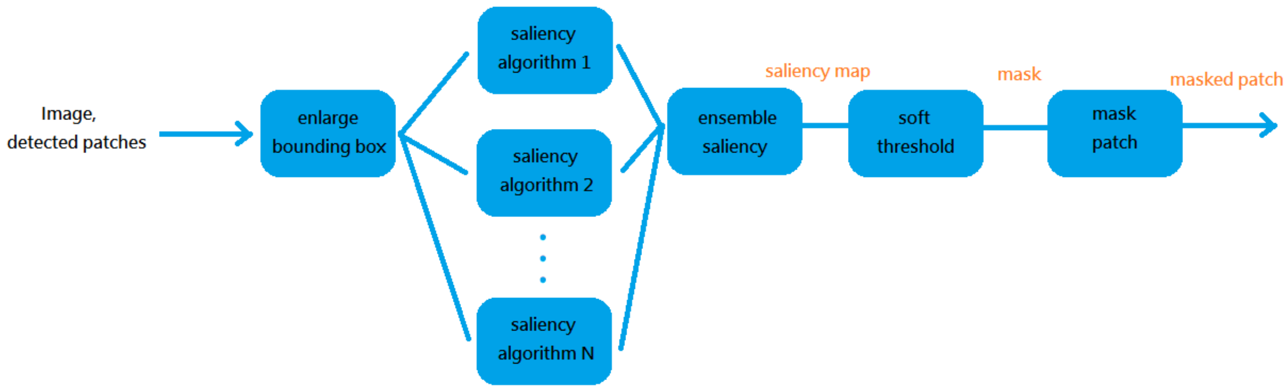


Figure 6.20: The details of the architecture of the saliency block.

doi:10.6342/NTU201801036

6.4.4 Saliency results

Around two thousand additional patches are requested for evaluation. Table 6.5 is the comparison of each algorithm with the ground-truth. The method ”ensemble” is the saliency after ensemble of four algorithms [7, 8, 9, 10]. The method ”mask” is the saliency after ensemble and soft-threshold. The recall, precision and F1-score are defined as follows. Suppose an algorithm produces a saliency map S for an input patch I and the groundtruth of I is y. The value of S is a floating number in [0, 1] and the value of y is a binary integer in $\{0,1\}$. First, a binary mask S_b of S is calculated by a threshold τ (i.e. $\tau = 0.8$). Then S_b is compared with y for recall, precision, and F1-score.

$$S_b(x;I) = \begin{cases} 1, \text{if } S(x;I) > 0.8 \\ 0, \text{if } S(x;I) \leq 0.8 \end{cases} \tag{6.17}$$

$$TP(I) = card(\{x|S_b(x;I) = 1, y(x;I) = 1\}) \tag{6.18}$$

$$FP(I) = card(\{x|S_b(x;I) = 1, y(x;I) = 0\}) \tag{6.19}$$

$$FN(I) = card(\{x|S_b(x;I) = 0, y(x;I) = 1\}) \tag{6.20}$$

$$recall = \frac{\sum_I TP(I)}{\sum_I TP(I) + FN(I)} \tag{6.21}$$

$$precision = \frac{\sum_I TP(I)}{\sum_I TP(I) + FP(I)} \tag{6.22}$$

$$F_1 = 2 \cdot \frac{recall \cdot precision}{recall + precision} \tag{6.23}$$

6.5 Results on masked images

The DenseNet121 model trained on the masked dolphin dataset has performance around 85.53% testing accuracy. Figure 6.21 are examples of gradient-based saliency maps of

doi:10.6342/NTU201801036

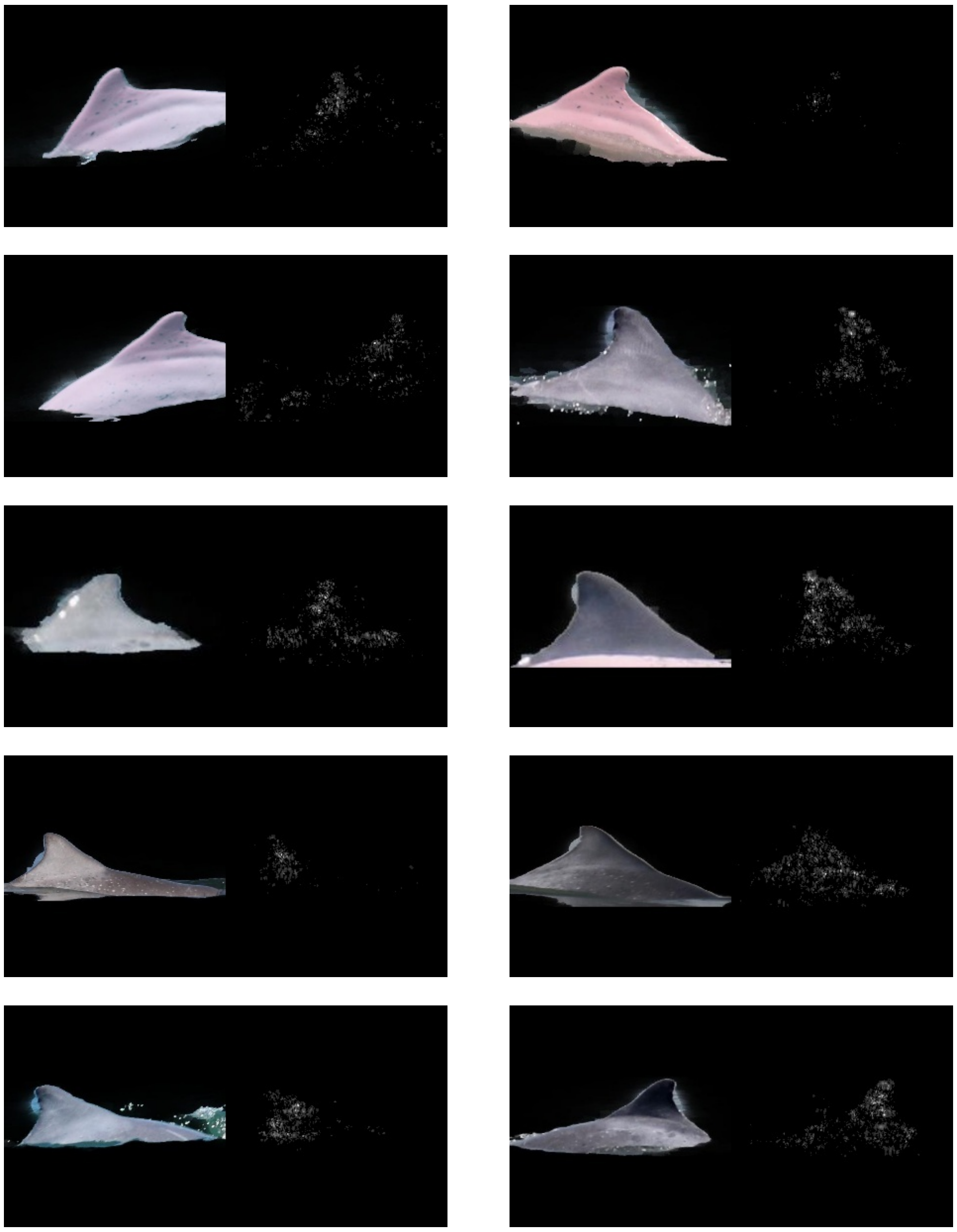


Figure 6.21: The gradient-based saliency map of the dolphin patches on the DenseNet121 trained on masked dolphin dataset. The whiter the pixel, the higher value of the gradient.

doi:10.6342/NTU201801036

Table 6.5: The statistics of saliency results. The method "ensemble" is the saliency after ensemble of four algorithms [7, 8, 9, 10]. The method "mask" is the saliency after ensemble and soft-threshold.

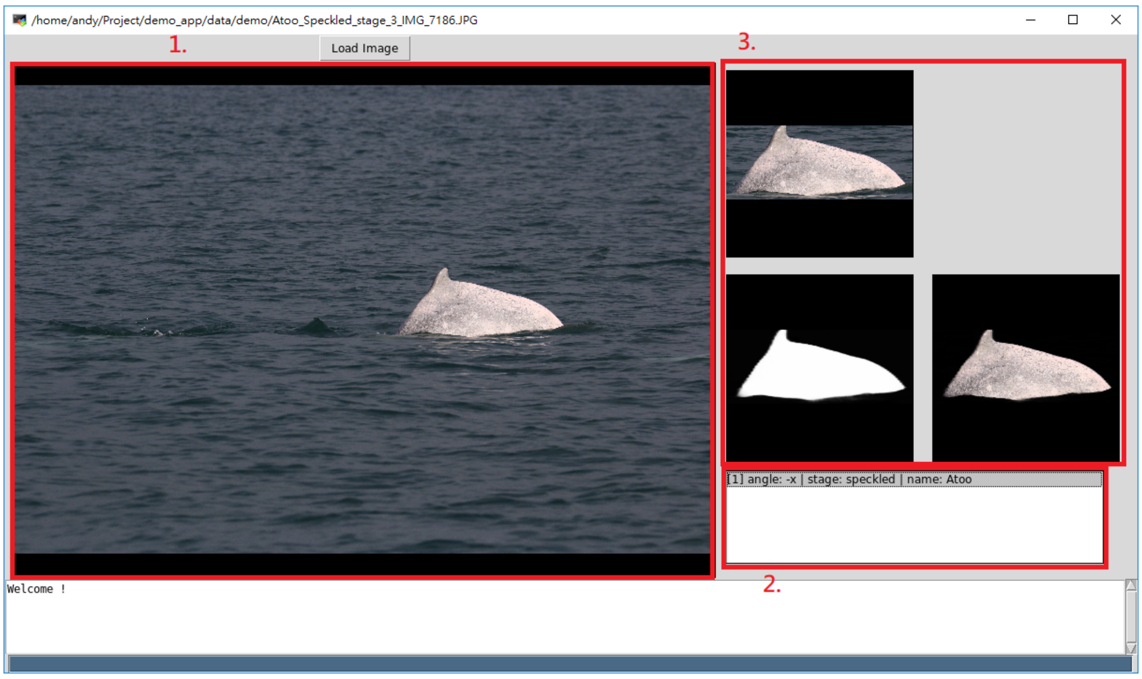
Bounding-box size	Methods	Recall	Precision	F1-score
original	NLDF [7]	0.662283	0.660265	0.661273
original	SODM [8]	0.997361	0.382603	0.553049
original	GBVS [9]	0.999694	0.385531	0.556463
original	Saliency-HDCT [10]	0.997758	0.384238	0.554816
original	ensemble	0.999992	0.382886	0.553748
original	mask	0.739775	0.850178	0.791143
twice size	NLDF [7]	0.962076	0.683463	0.799183
twice size	SODM [8]	0.998800	0.383006	0.553691
twice size	GBVS [9]	1.000000	0.382877	0.553740
twice size	Saliency-HDCT [10]	0.998016	0.385311	0.555974
twice size	ensemble	1.000000	0.382876	0.553739
twice size	mask	0.979391	0.689390	0.809192

masked images on the DenseNet121 trained on the masked dolphin dataset. From Figure 6.21, the gradient-based saliency lie on the dolphin and especially on the fins. Hence, the model trained on the masked dolphin dataset concentrates on details of dolphins for classification rather than on irrelevant information to dolphins such as sea surfaces.

6.6 Demo Application

In addition, a desktop application is built upon Python Tkinter package. The GUI of the application is as Figure 6.22. It includes a section displaying the input image, a list showing the classification results of the detected dolphins, and a section displaying the patch, the saliency and the masked patch of the selected dolphin.

doi:10.6342/NTU201801036


/home/andy/Project/demo_app/data/demo/Atoo_Speckled_stage_3_IMG_7186.JPG
1.
Load Image
3.
2.

[1] angle: -x | stage: speckled | name: Atoo

Welcome !

Figure 6.22: The GUI of the dolphin application. Section 1 is the input image. Section 2 is the list of detected dolphins and classification results. Section 3 is the patch, the saliency and the masked patch of the selected dolphin.

doi:10.6342/NTU201801036

Chapter 7 Conclusions and Future Works

7.1 Conclusions

A class-agnostic visual tracking algorithm Faster-MDNet [46] is proposed to replace the on-line training in MD-Net [1] which is a bottleneck of the computational cost in prediction with a RNN based model adaptation strategy. In the MD-Net, the information of the targets are raw patches and corresponding labels and the model update is executed through back-propagation training on the convolutional neural network model. In the Faster-MDNet, the information of the targets are encoded as a feature and the model update is excuted through injection of the feature into the prediction branch in convolutional neural network. The experimental results demonstrates the proposed Faster-MDNet runs around 10 times faster than the MD-Net with little sacrifice on accuracy.

A class-agnostic visual tracking algorithm RDisp is proposed. The RDisp transfer pretrained detection model into tracking field and incorporate pretrained detection model with convolustional LSTM cells to build the tracking model. In addition, a two-stage clip training is proposed to train the RDisp to solve the problem of memory limit and loss decreasing in utilization of back-propagation through time on training the RDisp. The experimental results demonstrates the RDisp runs at 25 frames per second higher than the prediction time of state-of-the-art algorithms. Qualitative results demonstrates consistency of the algorithm under versatile circumstances of target appearance changes.

doi:10.6342/NTU201801036

A system of dolphin detection and identification is proposed to alleviate the tremendous human resources on dolphin image capturing and post-processing. The dolphin detection is based on the Faster-RCNN detection model [5]. The dolphin identification is based on the DenseNet classification model [11]. In addition, a hybrid saliency method based on four saliency algorithms [7, 8, 9, 10] is proposed to remove the interference of sea surfaces in classification of dolphin names. A desktop application of the overall dolphin detection and identification system is built upon python TKinter for user-friendly utilization of our system.

7.2 Future Works

In the proposed Faster-MDNet, the recurrent neural network model state is generated by a direct copy of the fully connected layers and the model update is by a direct summation of the features with the state. A more complicated model that considers long-term information and more detailed appearance representations such as LSTM or skip connections can be attempted for higher accuracy and shorter computational time. Furthermore, in the Faster-MDNet, the ensemble of states of different patches are a score-weighted summation average. A smarter ensemble method can be attempted for better utilization of the information in states of different patches. In addition, the region proposal strategy in the Faster-MDNet is the selective search [57]. A smarter region proposal method such as region proposal network [5] can be attempted for faster execution and more compact enclosure of the proposals with the targets.

In the proposed RDisp, the initial state extraction method is to input a masked image on the target to the pretrained detection model and extract output features of specific layers as initial states. The masking method is to multiply the frame with a mask where the pixels inside the groundtruth bounding-box have value 1 and the pixels outside the

doi:10.6342/NTU201801036

groundtruth bounding-box have value 0. A more complicated masking method can be attempted to leave more background knowledge for better exclusion of background objects. Futhermore, the memory limit problem is solved by two-stage clip training and communications of model features between CPU and GPU memory during training in the RDisp. A more memory-efficient model can be attempted to solve the problem of memory limit fundamentally.

In the proposed dolphin detection and identification model, an end-to-end model combining detection and classification model can be attempted, which requires a training strategy under incompletely annotated data. Futhermore, the saliency detection and the classification can be combined for higher accuracy. Finally, a faster model which can run without GPU can be beneficial for marine conservationists who are not familiar with the setup of environments on GPU. Hence, a faster model that can run real-time on CPU is also a direction of future works in dolphin detection and identification.

doi:10.6342/NTU201801036

References

Chapter 1. Introduction

[1] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," *CoRR*, vol. abs/1510.07945, 2015. arXiv: 1510.07945. [Online]. Available: http://arxiv.org/abs/1510.07945.

[2] H. Nam, M. Baek, and B. Han, "Modeling and propagating cnns in a tree structure for visual tracking," *CoRR*, vol. abs/1608.07242, 2016. arXiv: 1608.07242. [Online]. Available: http://arxiv.org/abs/1608.07242.

[3] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[4] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, Springer, 2014, pp. 740–755.

[5] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds., Curran Associates, Inc., 2015, pp. 91–99. [Online]. Available: http:

doi:10.6342/NTU201801036

//papers.nips.cc/paper/5638-faster-r-cnn-towards-real-time-object-detection-with-region-proposal-networks.pdf.

[6] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

[7] Z. Luo, A. Mishra, A. Achkar, J. Eichel, S. Li, and P.-M. Jodoin, "Non-local deep features for salient object detection," in *IEEE CVPR*, 2017.

[8] Q. Hou, M.-M. Cheng, X. Hu, A. Borji, Z. Tu, and P. Torr, "Deeply supervised salient object detection with short connections," in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, IEEE, 2017, pp. 5300–5309.

[9] J. Harel, C. Koch, and P. Perona, "Graph-based visual saliency," in *Advances in neural information processing systems*, 2007, pp. 545–552.

[10] J. Kim, D. Han, Y.-W. Tai, and J. Kim, "Salient region detection via high-dimensional color transform and local spatial support," *IEEE transactions on image processing*, vol. 25, no. 1, pp. 9–23, 2016.

[11] F. Iandola, M. Moskewicz, S. Karayev, R. Girshick, T. Darrell, and K. Keutzer, "Densenet: Implementing efficient convnet descriptor pyramids," *arXiv preprint arXiv:1404.1869*, 2014.

Chapter 2. Reviews of Tracking Techniques

[12] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Comput. Surv.*, vol. 38, no. 4, Dec. 2006, ISSN: 0360-0300. DOI: 10.1145/1177352.1177355. [Online]. Available: http://doi.acm.org/10.1145/1177352.1177355.

doi:10.6342/NTU201801036

[13] K. Litomisky, "Consumer rgb-d cameras and their applications," *Rapport technique, University of California*, vol. 20, 2012.

[14] S. Dubuisson and C. Gonzales, "A survey of datasets for visual tracking," *Machine Vision and Applications*, vol. 27, pp. 23–52, 2015.

[15] L. Schwarz, A. Mkhitaryan, D. Mateus, and N. Navab, "Human skeleton tracking from depth data using geodesic distances and optical flow," vol. 30, 217??26, Mar. 2012.

[16] Q. Cai, D. Gallup, C. Zhang, and Z. Zhang, "3d deformable face tracking with a commodity depth camera," in *Computer Vision – ECCV 2010*, K. Daniilidis, P. Maragos, and N. Paragios, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 229–242, ISBN: 978-3-642-15558-1.

[17] C. Qian, X. Sun, Y. Wei, X. Tang, and J. Sun, "Realtime and robust hand tracking from depth," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2014, pp. 1106–1113. DOI: 10.1109/CVPR.2014.145.

[18] H. Nanda and K. Fujimura, "Visual tracking using depth data," in *2004 Conference on Computer Vision and Pattern Recognition Workshop*, Jun. 2004, pp. 37–37. DOI: 10.1109/CVPR.2004.202.

[19] M. Firman, "RGBD Datasets: Past, Present and Future," in *CVPR Workshop on Large Scale 3D Data: Acquisition, Modelling and Analysis*, 2016.

[20] P. K. Nathan Silberman Derek Hoiem and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *ECCV*, 2012.

[21] M. Camplani, A. Paiement, M. Mirmehdi, D. Damen, S. L. Hannuna, T. Burghardt, and L. Tao, "Multiple human tracking in RGB-D data: A survey," *CoRR*, vol. abs/1606.04450, 2016. arXiv: 1606.04450. [Online]. Available: http://arxiv.org/abs/1606.04450.

doi:10.6342/NTU201801036

[22] S. Song and J. Xiao, "Tracking revisited using rgbd camera: Unified benchmark and baselines," in *Proceedings of the 2013 IEEE International Conference on Computer Vision*, ser. ICCV '13, Washington, DC, USA: IEEE Computer Society, 2013, pp. 233–240, ISBN: 978-1-4799-2840-8. DOI: 10.1109/ICCV.2013.36. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2013.36.

[23] L. Cruz, D. Lucio, and L. Velho, "Kinect and rgbd images: Challenges and applications," in *2012 25th SIBGRAPI Conference on Graphics, Patterns and Images Tutorials*, Aug. 2012, pp. 36–49. DOI: 10.1109/SIBGRAPI-T.2012.13.

[24] O. H. Jafari, D. Mitzel, and B. Leibe, "Real-time rgb-d based people detection and tracking for mobile robots and head-worn cameras," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 5636–5643. DOI: 10.1109/ICRA.2014.6907688.

[25] M. Luber, L. Spinello, and K. O. Arras, "People tracking in rgb-d data with online boosted target models," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2011, pp. 3844–3849. DOI: 10.1109/IROS.2011.6095075.

[26] A. G. A. Perera, C. Srinivas, A. Hoogs, G. Brooksby, and W. Hu, "Multi-object tracking through simultaneous long occlusions and split-merge conditions," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 1, Jun. 2006, pp. 666–673. DOI: 10.1109/CVPR.2006.195.

[27] B. Yang and R. Nevatia, "Multi-target tracking by online learning of non-linear motion patterns and robust appearance models," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2012, pp. 1918–1925. DOI: 10.1109/CVPR.2012.6247892.

doi:10.6342/NTU201801036

[28] H. Izadinia, V. Ramakrishna, K. M. Kitani, and D. Huber, "Multi-pose multi-target tracking for activity understanding," in *2013 IEEE Workshop on Applications of Computer Vision (WACV)*, Jan. 2013, pp. 385–390. DOI: 10.1109/WACV.2013.6475044.

[29] A. A. Perera, C. Srinivas, A. Hoogs, G. Brooksby, and W. Hu, "Multi-object tracking through simultaneous long occlusions and split-merge conditions," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, IEEE, vol. 1, 2006, pp. 666–673.

[30] A. W. M. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah, "Visual tracking: An experimental survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 7, pp. 1442–1468, Jul. 2014, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2013.230.

[31] D. M. Chu and A. W. M. Smeulders, "Thirteen hard cases in visual tracking," in *2010 7th IEEE International Conference on Advanced Video and Signal Based Surveillance*, Aug. 2010, pp. 103–110. DOI: 10.1109/AVSS.2010.85.

[32] Y. Wu, J. Lim, and M. H. Yang, "Online object tracking: A benchmark," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2013, pp. 2411–2418. DOI: 10.1109/CVPR.2013.312.

[33] ——, "Object tracking benchmark," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1834–1848, Sep. 2015, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2014.2388226.

[34] M. Kristan, J. Matas, A. Leonardis, T. Vojir, R. Pflugfelder, G. Fernandez, G. Nebehay, F. Porikli, and L. Čehovin, "A novel performance evaluation methodology for single-target trackers," *IEEE Transactions on Pattern Analysis and Machine*

doi:10.6342/NTU201801036

Intelligence, vol. 38, no. 11, pp. 2137–2155, Nov. 2016, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2016.2516982.

Chapter 3. Proposed Faster-MDNet

[1] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," *CoRR*, vol. abs/1510.07945, 2015. arXiv: 1510.07945. [Online]. Available: http://arxiv.org/abs/1510.07945.

[32] Y. Wu, J. Lim, and M. H. Yang, "Online object tracking: A benchmark," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2013, pp. 2411–2418. DOI: 10.1109/CVPR.2013.312.

[35] X. Mei and H. Ling, "Robust visual tracking using l1 minimization," in *2009 IEEE 12th International Conference on Computer Vision*, Sep. 2009, pp. 1436–1443. DOI: 10.1109/ICCV.2009.5459292.

[36] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja, "Robust visual tracking via multi-task sparse learning," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2012, pp. 2042–2049. DOI: 10.1109/CVPR.2012.6247908.

[37] M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg, "Beyond correlation filters: Learning continuous convolution operators for visual tracking," *CoRR*, vol. abs/1608.03773, 2016. arXiv: 1608.03773. [Online]. Available: http://arxiv.org/abs/1608.03773.

[38] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, "Fully-convolutional siamese networks for object tracking," *CoRR*, vol. abs/1606.09549, 2016. arXiv: 1606.09549. [Online]. Available: http://arxiv.org/abs/1606.09549.

[39] Q. Gan, Q. Guo, Z. Zhang, and K. Cho, "First step toward model-free, anonymous object tracking with recurrent neural networks," *CoRR*, vol. abs/1511.06425, 2015. arXiv: 1511.06425. [Online]. Available: http://arxiv.org/abs/1511.06425.

[40] S. E. Kahou, V. Michalski, and R. Memisevic, "RATM: recurrent attentive tracking model," *CoRR*, vol. abs/1510.08660, 2015. arXiv: 1510.08660. [Online]. Available: http://arxiv.org/abs/1510.08660.

[41] J. Dequaire, D. Rao, P. Ondruska, D. Z. Wang, and I. Posner, "Deep tracking on the move: Learning to track the world from a moving vehicle using recurrent neural networks," *CoRR*, vol. abs/1609.09365, 2016. arXiv: 1609.09365. [Online]. Available: http://arxiv.org/abs/1609.09365.

[42] G. Ning, Z. Zhang, C. Huang, Z. He, X. Ren, and H. Wang, "Spatially supervised recurrent convolutional neural networks for visual object tracking," *CoRR*, vol. abs/1607.05781, 2016. arXiv: 1607.05781. [Online]. Available: http://arxiv.org/abs/1607.05781.

[43] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015. arXiv: 1506.02640. [Online]. Available: http://arxiv.org/abs/1506.02640.

[44] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," *CoRR*, vol. abs/1405.3531, 2014. arXiv: 1405.3531. [Online]. Available: http://arxiv.org/abs/1405.3531.

[45] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, IEEE, 2009, pp. 248–255.

doi:10.6342/NTU201801036

Chapter 4. Proposed RDisp: Recurrent Detection is Powerful

[1] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," *CoRR*, vol. abs/1510.07945, 2015. arXiv: 1510.07945. [Online]. Available: http://arxiv.org/abs/1510.07945.

[5] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds., Curran Associates, Inc., 2015, pp. 91–99. [Online]. Available: http://papers.nips.cc/paper/5638-faster-r-cnn-towards-real-time-object-detection-with-region-proposal-networks.pdf.

[32] Y. Wu, J. Lim, and M. H. Yang, "Online object tracking: A benchmark," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2013, pp. 2411–2418. DOI: 10.1109/CVPR.2013.312.

[33] ——, "Object tracking benchmark," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1834–1848, Sep. 2015, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2014.2388226.

[43] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015. arXiv: 1506.02640. [Online]. Available: http://arxiv.org/abs/1506.02640.

[46] H. W. Hsu and J. J. Ding, "Fastermdnet: Learning model adaptation by rnn in tracking-by-detection based visual tracking," in *2017 Asia-Pacific Signal and In-*

doi:10.6342/NTU201801036

formation Processing Association Annual Summit and Conference (APSIPA ASC), Dec. 2017, pp. 657–660. DOI: 10.1109/APSIPA.2017.8282115.

[47] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, "SSD: single shot multibox detector," in *ECCV (1)*, ser. Lecture Notes in Computer Science, vol. 9905, Springer, 2016, pp. 21–37.

[48] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," *CoRR*, vol. abs/1612.08242, 2016. arXiv: 1612.08242. [Online]. Available: http://arxiv.org/abs/1612.08242.

[49] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'15, Montreal, Canada: MIT Press, 2015, pp. 802–810. [Online]. Available: http://dl.acm.org/citation.cfm?id=2969239.2969329.

[50] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, IEEE, 2015, pp. 3156–3164.

[51] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Čehovin, T. Vojíř, G. Häger, A. Lukežič, G. Fernández, A. Gupta, A. Petrosino, A. Memarmoghadam, A. Garcia-Martin, A. Solís Montero, A. Vedaldi, A. Robinson, A. J. Ma, A. Varfolomieiev, A. Alatan, A. Erdem, B. Ghanem, B. Liu, B. Han, B. Martinez, C.-M. Chang, C. Xu, C. Sun, D. Kim, D. Chen, D. Du, D. Mishra, D.-Y. Yeung, E. Gundogdu, E. Erdem, F. Khan, F. Porikli, F. Zhao, F. Bunyak, F. Battistone, G. Zhu, G. Roffo, G. R. K. S. Subrahmanyam, G. Bastos, G. Seetharaman, H. Medeiros, H. Li, H. Qi, H. Bischof, H. Possegger, H. Lu, H. Lee, H. Nam, H. J. Chang, I. Drummond, J. Valmadre, J.-c. Jeong, J.-i. Cho, J.-Y. Lee, J. Zhu, J. Feng, J. Gao, J. Y. Choi, J. Xiao, J.-W. Kim, J. Jeong, J. F. Henriques, J. Lang, J. Choi, J. M.

doi:10.6342/NTU201801036

Martinez, J. Xing, J. Gao, K. Palaniappan, K. Lebeda, K. Gao, K. Mikolajczyk, L. Qin, L. Wang, L. Wen, L. Bertinetto, M. K. Rapuru, M. Poostchi, M. Maresca, M. Danelljan, M. Mueller, M. Zhang, M. Arens, M. Valstar, M. Tang, M. Baek, M. H. Khan, N. Wang, N. Fan, N. Al-Shakarji, O. Miksik, O. Akin, P. Moallem, P. Senna, P. H. S. Torr, P. C. Yuen, Q. Huang, R. Martin-Nieto, R. Pelapur, R. Bowden, R. Laganière, R. Stolkin, R. Walsh, S. B. Krah, S. Li, S. Zhang, S. Yao, S. Hadfield, S. Melzi, S. Lyu, S. Li, S. Becker, S. Golodetz, S. Kakanuru, S. Choi, T. Hu, T. Mauthner, T. Zhang, T. Pridmore, V. Santopietro, W. Hu, W. Li, W. Hübner, X. Lan, X. Wang, X. Li, Y. Li, Y. Demiris, Y. Wang, Y. Qi, Z. Yuan, Z. Cai, Z. Xu, Z. He, and Z. Chi, "The visual object tracking vot2016 challenge results," in *Computer Vision – ECCV 2016 Workshops*, G. Hua and H. Jégou, Eds., Cham: Springer International Publishing, 2016, pp. 777–823, ISBN: 978-3-319-48881-3.

[52] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg, "Learning spatially regularized correlation filters for visual tracking," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4310–4318.

[53] M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg, "Beyond correlation filters: Learning continuous convolution operators for visual tracking," in *European Conference on Computer Vision*, Springer, 2016, pp. 472–488.

[54] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 583–596, 2015.

doi:10.6342/NTU201801036

Chapter 5. Proposed Dolphin Detection

[4] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, Springer, 2014, pp. 740–755.

[5] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds., Curran Associates, Inc., 2015, pp. 91–99. [Online]. Available: http://papers.nips.cc/paper/5638-faster-r-cnn-towards-real-time-object-detection-with-region-proposal-networks.pdf.

[45] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, IEEE, 2009, pp. 248–255.

[48] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," *CoRR*, vol. abs/1612.08242, 2016. arXiv: 1612.08242. [Online]. Available: http://arxiv.org/abs/1612.08242.

[55] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

Chapter 6. Proposed Dolphin Identification

[6] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

[7] Z. Luo, A. Mishra, A. Achkar, J. Eichel, S. Li, and P.-M. Jodoin, "Non-local deep features for salient object detection," in *IEEE CVPR*, 2017.

doi:10.6342/NTU201801036

[8] Q. Hou, M.-M. Cheng, X. Hu, A. Borji, Z. Tu, and P. Torr, "Deeply supervised salient object detection with short connections," in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, IEEE, 2017, pp. 5300–5309.

[9] J. Harel, C. Koch, and P. Perona, "Graph-based visual saliency," in *Advances in neural information processing systems*, 2007, pp. 545–552.

[10] J. Kim, D. Han, Y.-W. Tai, and J. Kim, "Salient region detection via high-dimensional color transform and local spatial support," *IEEE transactions on image processing*, vol. 25, no. 1, pp. 9–23, 2016.

[11] F. Iandola, M. Moskewicz, S. Karayev, R. Girshick, T. Darrell, and K. Keutzer, "Densenet: Implementing efficient convnet descriptor pyramids," *arXiv preprint arXiv:1404.1869*, 2014.

[45] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, IEEE, 2009, pp. 248–255.

[56] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

Chapter 7. Conclusions and Future Works

[1] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," *CoRR*, vol. abs/1510.07945, 2015. arXiv: 1510.07945. [Online]. Available: http://arxiv.org/abs/1510.07945.

doi:10.6342/NTU201801036

[5] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds., Curran Associates, Inc., 2015, pp. 91–99. [Online]. Available: http://papers.nips.cc/paper/5638-faster-r-cnn-towards-real-time-object-detection-with-region-proposal-networks.pdf.

[7] Z. Luo, A. Mishra, A. Achkar, J. Eichel, S. Li, and P.-M. Jodoin, “Non-local deep features for salient object detection,” in *IEEE CVPR*, 2017.

[8] Q. Hou, M.-M. Cheng, X. Hu, A. Borji, Z. Tu, and P. Torr, “Deeply supervised salient object detection with short connections,” in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, IEEE, 2017, pp. 5300–5309.

[9] J. Harel, C. Koch, and P. Perona, “Graph-based visual saliency,” in *Advances in neural information processing systems*, 2007, pp. 545–552.

[10] J. Kim, D. Han, Y.-W. Tai, and J. Kim, “Salient region detection via high-dimensional color transform and local spatial support,” *IEEE transactions on image processing*, vol. 25, no. 1, pp. 9–23, 2016.

[11] F. Iandola, M. Moskewicz, S. Karayev, R. Girshick, T. Darrell, and K. Keutzer, “Densenet: Implementing efficient convnet descriptor pyramids,” *arXiv preprint arXiv:1404.1869*, 2014.

[46] H. W. Hsu and J. J. Ding, “Fastermdnet: Learning model adaptation by rnn in tracking-by-detection based visual tracking,” in *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, Dec. 2017, pp. 657–660. DOI: 10.1109/APSIPA.2017.8282115.

doi:10.6342/NTU201801036

[57] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, “Selective search for object recognition,” *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.

doi:10.6342/NTU201801036