Exploiting scene context for on-line object tracking in unconstrained environments

Salma Moujtahid

N°d'ordre NNT : 2016LYSEI110

THESE de DOCTORAT DE L'UNIVERSITE DE LYON
opérée au sein de
l'INSA LYON

Ecole Doctorale N° UMR512
INFORMATIQUES ET MATHEMATIQUES

Spécialité de doctorat : Informatique

Soutenue publiquement le 03/11/2016, par :
Salma MOUJTAHID

Exploiting scene context for on-line object tracking in unconstrained environments

---

Devant le jury composé de :

CHATEAU, Thierry	Prof.	Université Blaise Pascal	Rapporteur
BREMOND, François	DR	INRIA Sophia-Antipolis	Rapporteur
ODOBEZ, Jean-Marc	MER	IDIAP Reseash Institure	Examinateur
BENOIS-PINEAU, Jenny	Prof.	Université Bordeaux 1	Examinatrice
BASKURT, Atilla	Prof.	INSA Lyon	Directeur de thèse
DUFFNER, Stefan	MDC	INSA Lyon	Examinateur

Département FEDORA – INSA Lyon - Ecoles Doctorales – Quinquennal 2016-2020

SIGLE	ECOLE DOCTORALE	NOM ET COORDONNEES DU RESPONSABLE
CHIMIE	**CHIMIE DE LYON** **http://www.edchimie-lyon.fr** Sec : Renée EL MELHEM Bat Blaise Pascal 3e etage secretariat@edchimie-lyon.fr Insa : R. GOURDON	M. Stéphane DANIELE Institut de Recherches sur la Catalyse et l'Environnement de Lyon IRCELYON-UMR 5256 Équipe CDFA 2 avenue Albert Einstein 69626 Villeurbanne cedex directeur@edchimie-lyon.fr
E.E.A.	**ELECTRONIQUE, ELECTROTECHNIQUE, AUTOMATIQUE** **http://edeea.ec-lyon.fr** Sec : M.C. HAVGOUDOUKIAN Ecole-Doctorale.eea@ec-lyon.fr	M. Gérard SCORLETTI Ecole Centrale de Lyon 36 avenue Guy de Collongue 69134 ECULLY Tél : 04.72.18 60.97 Fax : 04 78 43 37 17 Gerard.scorletti@ec-lyon.fr
E2M2	**EVOLUTION, ECOSYSTEME, MICROBIOLOGIE, MODELISATION** **http://e2m2.universite-lyon.fr** Sec : Safia AIT CHALAL Bat Darwin - UCB Lyon 1 04.72.43.28.91 Insa : H. CHARLES Safia.ait-chalal@univ-lyon1.fr	Mme Gudrun BORNETTE CNRS UMR 5023 LEHNA Université Claude Bernard Lyon 1 Bât Forel 43 bd du 11 novembre 1918 69622 VILLEURBANNE Cédex Tél : 06.07.53.89.13 e2m2@ univ-lyon1.fr
EDISS	**INTERDISCIPLINAIRE SCIENCES-SANTE** **http://www.ediss-lyon.fr** Sec : Safia AIT CHALAL Hôpital Louis Pradel - Bron 04 72 68 49 09 Insa : M. LAGARDE Safia.ait-chalal@univ-lyon1.fr	Mme Emmanuelle CANET-SOULAS INSERM U1060, CarMeN lab, Univ. Lyon 1 Bâtiment IMBL 11 avenue Jean Capelle INSA de Lyon 696621 Villeurbanne Tél : 04.72.68.49.09 Fax :04 72 68 49 16 Emmanuelle.canet@univ-lyon1.fr
INFOMATHS	**INFORMATIQUE ET MATHEMATIQUES** **http://infomaths.univ-lyon1.fr** Sec :Renée EL MELHEM Bat Blaise Pascal 3e etage infomaths@univ-lyon1.fr	Mme Sylvie CALABRETTO LIRIS – INSA de Lyon Bat Blaise Pascal 7 avenue Jean Capelle 69622 VILLEURBANNE Cedex Tél : 04.72. 43. 80. 46 Fax 04 72 43 16 87 Sylvie.calabretto@insa-lyon.fr
Matériaux	**MATERIAUX DE LYON** **http://ed34.universite-lyon.fr** Sec : M. LABOUNE PM : 71.70 –Fax : 87.12 Bat. Saint Exupéry Ed.materiaux@insa-lyon.fr	M. Jean-Yves BUFFIERE INSA de Lyon MATEIS Bâtiment Saint Exupéry 7 avenue Jean Capelle 69621 VILLEURBANNE Cedex Tél : 04.72.43 71.70 Fax 04 72 43 85 28 Ed.materiaux@insa-lyon.fr
MEGA	**MECANIQUE, ENERGETIQUE, GENIE CIVIL, ACOUSTIQUE** **http://mega.universite-lyon.fr** Sec : M. LABOUNE PM : 71.70 –Fax : 87.12 Bat. Saint Exupéry mega@insa-lyon.fr	M. Philippe BOISSE INSA de Lyon Laboratoire LAMCOS Bâtiment Jacquard 25 bis avenue Jean Capelle 69621 VILLEURBANNE Cedex Tél : 04.72 .43.71.70 Fax : 04 72 43 72 37 Philippe.boisse@insa-lyon.fr
ScSo	**ScSo*** **http://recherche.univ-lyon2.fr/scso/** Sec : Viviane POLSINELLI Brigitte DUBOIS Insa : J.Y. TOUSSAINT viviane.polsinelli@univ-lyon2.fr	Mme Isabelle VON BUELTZINGLOEWEN Université Lyon 2 86 rue Pasteur 69365 LYON Cedex 07 Tél : 04.78.77.23.86 Fax : 04.37.28.04.48

*ScSo : Histoire, Géographie, Aménagement, Urbanisme, Archéologie, Science politique, Sociologie, Anthropologie

"*Time will always flow. Everything will pass by. That might be why youth is beautiful. It shines, blindingly bright, for just an instant. But you can never go back to it.*"

– Deok Sun - Reply 1988

Edited on November 23, 2016

Abstract

Exploiting scene context for on-line object tracking in unconstrained environments

by Salma Moujtahid

With the increasing need for automated video analysis, visual object tracking became an important task in computer vision. Object tracking is used in a wide range of applications such as surveillance, human-computer interaction, medical imaging or vehicle navigation. A tracking algorithm in unconstrained environments faces multiple challenges : potential changes in object shape and background, lighting, camera motion, and other adverse acquisition conditions.

In this setting, classic methods of background subtraction are inadequate, and more discriminative methods of object detection are needed. Moreover, in generic tracking algorithms, the nature of the object is not known a priori. Thus, off-line learned appearance models for specific types of objects such as faces, or pedestrians can not be used. Further, the recent evolution of powerful machine learning techniques enabled the development of new tracking methods that learn the object appearance in an online manner and adapt to the varying constraints in real time, leading to very robust tracking algorithms that can operate in non-stationary environments to some extent.

In this thesis, we start from the observation that different tracking algorithms have different strengths and weaknesses depending on the context. To overcome the varying challenges, we show that combining multiple modalities and tracking algorithms can considerably improve the overall tracking performance in unconstrained environments. More concretely, we first introduced a new tracker selection framework [MDB15b] using a spatial and temporal coherence criterion. In this algorithm, multiple independent trackers are combined in a parallel manner, each of them using low-level features based on different complementary visual aspects like colour, texture and shape. By recurrently selecting the most suitable tracker, the overall system can switch rapidly between different tracking algorithms with specific appearance models depending on the changes in the video.

In the second contribution [MDB15a], the scene context is introduced to the tracker selection. We designed effective visual features, extracted from the scene context to characterise the different image conditions and variations. At each point in time, a classifier is trained based on these features to predict the tracker that will perform best under the given scene conditions. We further improved this context-based framework and proposed an extended version [Mou+16], where the individual trackers are changed and the classifier training is optimised.

Finally, we started exploring one interesting perspective that is the use of a Convolutional Neural Network to automatically learn to extract these scene features directly from the input image and predict the most suitable tracker.

Résumé

Exploitation du contexte de scène pour le suivi d'objet en ligne dans des environnements non contraints

Introduction

Avec le besoin grandissant pour des modèles d'analyse automatique de vidéos, le suivi visuel d'objets est devenu une tâche primordiale dans le domaine de la vision par ordinateur. Le suivi d'objet est en effet utilisé dans de nombreux domaines tels que la vidéo surveillance, l'IHM (interaction homme machine), l'imagerie médicale ou encore la navigation de véhicule. Un algorithme de suivi dans un environnement non contraint fait face à de nombreuses difficultés: changements potentiels de la forme de l'objet, du background, de la luminosité, du mouvement de la camera, et autres conditions d'acquisition.

Dans la configuration environnement non contraint, les méthodes classiques de suivi de soustraction de fond ne sont pas adaptées, on a besoin de méthodes de détection d'objet plus discriminantes qui peuvent s'adapter aux changements dans la scène. De plus, la nature de l'objet est a priori inconnue dans les méthodes de tracking génériques. Ainsi, les modèles d'apparence d'objets, tel quel les visages ou humains, appris hors ligne ne peuvent être utilisés. L'évolution récente d'algorithmes d'apprentissage robustes a permis le développement de nouvelles méthodes de tracking qui apprennent l'apparence de l'objet de manière en ligne et s'adaptent aux variables contraintes en temps réel.

Etat de l'art du suivi d'objet

Le but de tout algorithme de suivi d'objet ou « tracker » est d'approximer le chemin d'un objet (et potentiellement sa taille) dans l'image en trouvant sa position dans chaque trame de la vidéo, pendant qu'il se déplace dans la scène. Dans le cas du suivi d'objet en ligne dans des environnements non contraints, on peut globalement grouper les trackers en trois catégories principales: *méthodes probabilistiques*, *méthodes de tracking par détection*, et les *méthodes basées corrélation*.

Les méthodes probabilistiques formulent la problématique de suivi d'objet comme une correspondance de l'objet à suivre à travers les trames, en prenant en compte les possibles perturbations et bruits dans la vidéo. C'est donc une estimation statistique de l'état futur d'un objet. La plus part des méthodes probabilistiques, tel que les filtres de Kalman [BC86; RS99; BH00] ou les filtre à particule [IB98b; Pér+02; NKMVG03; Bad+07], utilisent un type de chaine de Markov pour modéliser l'évolution temporelle ainsi que des techniques d'inférence Bayésienne.

Les méthodes de suivi par détection quant à elles utilisent un algorithme de détection d'objet afin d'obtenir les régions contenant l'objet à chaque trame, puis correspondent les positions à travers les trames. Ce sont des méthodes qui ont beaucoup de succés dans la communauté tracking grâce au développement d'algorithmes performants de détection d'objet et méthodes

d'apprentissage automatiques [DT05; LSS05; VJ01; VJ04]. Deux éléments sont primordiales dans les méthodes de suivi par détection : la représentation visuel (*i.e.* caractéristiques bas niveau, ou « features », tel que Haar [BYB11], HOG [Tan+07], LBP [GGB06], *etc.*) et la modélisation. La modélisation peut être générative, c'est-à-dire modélisation l'apparence de l'objet uniquement, ou discriminative, c'est-à-dire modélisation de l'objet et de son environnement en traitant le tracking d'objet comme un problème de classification de l'objet contre la scène. Plusieurs méthodes discriminatives ont été développées dans la littérature, utilisant différentes stratégies de classification: apprentissage basé boosting [CL05; Avi07; GB06; BYB09], apprentissage semi-supervisé [GLB08; YDM08; Tan+07], apprentissage multiple instance MIL [BYB09; BYB11], machines à vecteurs de support SVM [Avi04; HSHT11], par segmentation [NHY08; GRB13; DG13] ou encore apprentissage profond [NH15; Dan+15].

Contrairement aux méthodes de suivi par détection, les méthodes basées corrélation ne nécessitent la détection que lorsque l'objet apparait pour la première fois dans la scène. L'idée principale est de construire un modèle ou « template » de l'apparence de l'objet et rechercher à chaque nouvelle trame le patch le plus ressemblant, c'est-à-dire celui à la plus forte corrélation au template. Cette recherche peut être exécutée efficacement dans le domaine fréquentielle. Parmi ces techniques on peut citer le Template Matching simple [Bir98; BH01; SBW02] , Mean-Shift tracking [CRM00; CRM03; CM02; YDD05a], ou les filtres de corrélation [Bol+10; Hen+12; Dan+14b].

Au-delà de ces trois catégories, on peut également distinguer les méthodes dites de feature point tracking, qui formulent le suivi comme la correspondance de « features points » détectés (grâce à un détecteur de points d'intérêt) à travers les trames. On peut citer les trackers KLT [ST94; Bou01; KMM10b], ALIEN [PDB14] ou FoT [KT04; KMM10b; KMM12; VNM14].

Framework de sélection de tracker

Dans cette thèse, nous démarrons par l'observation suivant laquelle différents algorithmes de suivi ont différentes forces et faiblesses selon l'environnement et le contexte. Afin de surmonter les différentes contraintes des les environnements non contraints, nous démontrons que combiner plusieurs modalités et algorithmes peut améliorer considérablement la performance du suivi global. En effet, la combinaison de plusieurs modèles ou trackers en un seul framework de suivi d'objet est une approche commune dans la communauté du tracking. Cette combinaison peut prendre plusieurs formes selon à quel niveau elle est exécutée.

On peut alors distinguer les combinaisons *bas niveau* [Bir98; TM01; YLS04; CL05] qui opèrent au niveau du modèle, et les combinaisons *haut niveau* qui opèrent au niveau de la sortie du framework. De plus la combinaison peut prendre plusieurs formes tel que la sélection d'un tracker à partir d'un groupe de modèles [KL10; KL11; ZMS14], ou encore la fusion des sorties de plusieurs trackers [LLR06; SWC09; Li+12; KSC16; BPS14; WY14; VMN16; Zho+14a].

Les combinaisons bas-niveau mène généralement à des problèmes à cause de l'interdépendance des features dans le modèle. Nous partons de l'hypothèse que différents trackers ne performent pas correctement tout le temps ou en

même temps, ainsi la fusion haut niveau peut également mener à la divergence du suivi.

Nous proposons donc un nouveau framework de sélection de trackers basé sur un critère de cohérence spatio-temporel, nommé STC. Dans ce framework, plusieurs trackers indépendants sont combinés de manière parallèle, chacun d'entre eux utilisant des features bas niveau basées sur différents aspects visuels complémentaires tel que la couleur ou la texture. Nous appliquons le critère spatio-temporel qui permet d'éliminer, pour la trame courante, les trackers qui sont jugé comme ayant divergé. Puis se basant sur la confiance des trackers, un tracker est sélectionné, et son résultat de suivi est considéré comme la position de l'objet pour la trame courante.

En sélectionnant de manière récurrente le tracker le plus adapté à chaque trame, le système global peut switcher rapidement entre les différents trackers selon les changements dans la vidéo. De plus, le framework STC a le bénéfice d'être simple et ajoutant peu de complexité de calcul aux trackers individuels.

Nous avons choisi d'implémenter le framework STC avec trois trackers Online AdaBoost (OAB), chacun basé respectivement sur les features Haar, histogramme de gradients HOG et histogramme de couleur HOC. Dans un premier temps la performance des trackers OAB individuels a été analysée afin de comprendre leur complémentarité et les bénéfices d'une framework de combinaison de trackers. Puis dans un second temps, le framework STC a été évalué et comparé à des méthodes de combinaisons de trackers classiques.

Les résultats expérimentaux ont démontré que, même en présence de séquences vidéo difficiles, le framework proposé améliore la robustesse du suivi par rapport aux trackers OAB individuel. De plus le framework, surpasse les méthodes classiques de combinaison. Nous démontrons également qu'optimiser la stratégie d'update des trackers individuelle permet d'améliorer le suivi puisque c'est un composant principal des méthodes de suivi discriminatif.

Ainsi, combiner plusieurs trackers complémentaires et en selectionner un à chaque trame est une méthode qui permet d'améliorer la précision et la robustesse du suivi d'objet au delà de la performance des trackers individuels.

Classification du contexte de scène global pour sélection de tracker

Le développement des frameworks de combinaison de trackers est généralement dans le but de créer un algorithme de suivi capable de faire face aux différentes variations des conditions de la scène tout en maintenant un suivi robuste. Ainsi, il est logique d'inclure des informations provenant de la scène directement dans le processus de sélection ou fusion du framework afin de l'améliorer et l'adapter aux changements.

L'utilisation du contexte de la scène dans le suivi peut prendre plusieurs formes, tel que l'utilisation de supporteurs [YWH09; Gra+10; Wen+14] qui assistent le suivi en détectant des régions dans la scène ayant un mouvement similaire à l'objet. Des distracteurs, des régions d'apparence similaire à l'objet peuvent également être employés pour éviter la confusion dans le suivi [DVM11; HMT12]. D'autres méthodes exploitent le context spatio-temporel [Zha+14; Wen+14].

Dans cette seconde contribution de la thèse, nous proposons un nouveau framework de sélection de tracker (SCBT), où le contexte de scène est directement intégré dans le mécanisme de sélection de tracker. Nous avons conçu des caractéristiques visuelles, les «features de scène», extraites de l'image afin de caractériser les différentes conditions et variations de scène. Ainsi, grâce à ces features de scène, un classificateur (réseau de neurones) est entrainé dans le but de prédire à chaque instant le tracker qui performera le mieux sous les conditions de scènes données.

Les features de scènes proposées sont de simples opérations statistiques du premier et second ordre sur plusieurs domaines de l'image tel que l'intensité, la couleur ou encore les vecteurs mouvements. En plus de ces features, les confiances des trackers indépendant et l'identifiant du tracker précédemment sélectionné, sur une fenêtre temporelle, sont fournis au réseau de neurones. Le réseau produit donc un vecteur représentant la probabilité pour chaque tracker d'être le plus adéquat au données en entrée, et donc au contexte.

Le framework SCBT est constitué du réseau de neurones qui émet une première prédiction sur le tracker à utiliser pour la trame courante, qui est ensuite filtré par un modèle de Markov caché (HMM) qui permet d'éviter les changements trop fréquents et non-nécessaires entre plusieurs trackers. Enfin, un filtre de Kalman permet de lisser la trajectoire finale de l'objet.

L'évaluation de la performance du framework a procédé pour chaque composant de celui-ci. En commençant par le classificateur de scène, l'apprentissage du réseau avec plusieurs groupes de features de scène a été testé. Nous observons que l'utilisation des features de scènes seule n'est pas suffisante, mais l'intégration des confiances des trackers et du identifiant du précédant tracker permet d'avoir un meilleur apprentissage du réseau. L'utilisation d'une fenêtre temporelle de trois trames améliore également le taux d'apprentissage.

Ensuite, la performance du framework en terme de suivi a été évaluée suivant le benchmark VOT2013, montrant l'efficacité de l'utilisation du contexte de scène en surpassant les performance les trackers individuels (OAB), le framework de sélection précédemment présenté STC, et en se classant parmi les trackers de l'état de l'art dans le benchmark.

Extension du framework de sélection de tracker basé contexte de scène

Dans le framework SCBT présenté précédemment, deux composants sont essentiels pour la performance du suivi: les trackers individuels et le classificateur de scène. D'un côté, la performance des trackers individuels influence directement la performance de suivi global, puisqu'elle est limitée par ceux-ci. D'un autre côté, un apprentissage plus effectif du réseau de neurones peut également augmenter la performance globale. Ainsi, nous procédons à une analyse approfondie et évaluation de chaque composant du framework SCBT pour en présenter une version étendue, SCBT+.

Dans un premier temps, nous augmentons la base de données d'apprentissage qui permet d'améliorer l'apprentissage du classificateur de scène et d'en augmenter le taux. De plus, plusieurs stratégies d'apprentissages ont été investiguées, de la classification usuelle à la régression, pour trouver la configuration la plus adaptée à notre problématique. C'est la stratégie de «

ranking » qui a démontré les meilleurs résultats en classification et également en suivi. Cette stratégie attribue des valeurs croissantes du pire au meilleur tracker en termes de performance, c'est un croisement entre la régression et la classification binaire.

Dans un second temps plusieurs algorithmes de suivi par détection basées sur des features bas-niveau et accessibles ont été évalués sur le benchmark de tracking VOT2014. L'apprentissage du réseau a été évalué utilisant plusieurs combinaisons de trackers différents afin choisir la combinaison de trackers qui permet un meilleur apprentissage et une meilleure performance globale de suivi.

Ces expérimentations montrent que le choix des trackers peut jouer un rôle important dans l'apprentissage du réseau. Nous observons tout d'abord que le taux de classification du réseau ne reflète pas nécessairement la performance de suivi du classificateur. En effet, puisque le taux de classification du réseau prend les trames comme des exemples séparés et non en tant qu'une continuité, il est importance d'évaluer le suivi en plus de classification des différents réseaux appris. Nous observons également que lorsque la performance des trackers en termes de suivi est trop disparate, la combinaison de ceux-ci devient déséquilibrée. Dans cette configuration, les mauvaises prédictions du réseau peuvent considérablement diminuer la performance de suivi d'objet. D'un autre côté, en utilisant le même algorithme de suivi, Kernalized Correlation Filter KCF, avec plusieurs features, la performance du classificateur utilisant cette combinaison est améliorée et arrive à surpasser la performance des trackers individuels. Ainsi, les trackers OAB sont remplacés par trois trackers KCF basés sur des features RAW, HOG et LAB.

Enfin, le framework entier SCBT+ est évalué sur de multiples benchmarks de suivi, montrant une meilleur robustesse et précision dans le suivi par rapports aux précédents framework STC et SCBT et une performance d'état de l'art. Cependant, sur le benchmark le plus récent VOT2015, on peut voir que la performance du framework SCBT+ reste limitée par celles des trackers individuelles.

Apprentissage profond

Une des perspectives d'amélioration possibles des framework de selection basés context est l'utilisation de réseau de neurones convolutifs afin d'apprendre à extraire automatiquement les features de scène à partir de l'image d'entrée et prédire le tracker le plus adapté.

Ainsi, nous avons implémenté une architecture de réseaux convolutifs utilisant des images de trois trames en entrée, puis les confiances et les trackers sélectionnés précédement en tant de données d'entrée numérique. Le réseau est composé de trois couches parallèles de convolution et pooling pour chaque canal couleur HSV pour ainsi pouvoir extraire des features différentes sur chaque canal. Les expérimentations préliminaires ont montré que le réseau arrive à apprendre un peu du context de scène mais à tendance à sur-apprendre et finir par prédire le même tracker que le précédent. En effet, la problèmatique à apprendre ici qui est celle d'extraire des features associées à un tracker spécifique est une problématique très abstraite et de nature sémantique. d'autres architecture de réseau peuvent être explorées afin d'adapter le réseau à la problématique.

Conclusion

Durant cette thèse, plusieux frameworks innovants de suivi ont été proposés. Nous avons démontré dans un premier temps que la combination (set selection) de tracker est une stratégie interessante qui permet d'améliorer la performance des trackers individuels. De plus, nous avons intégré le context de scène dans le processus de selection ce a permit au framework de suivi de mieux s'adapter aux différents changements dans la scène et donc produire un suivi plus robuste. Nous avons également amélioré ce framework en optimisant l'apprentissage du classificateur de scène et en montrant l'influence du choix des trackers individuels sur la combinaison. Les différentes méthodes proposées ont été évaluées sur plusieurs benchmarks publiques, et ont démontré que l'utilisation du contexte de scène améliore la performance globale du suivi d'objet.

Nous avons commencé à explorer une perspective de recherche qui est l'utilisation d'un réseau convolutif et qui semble intéressante. D'autres perspectives seraient d'introduire des features de scène sémantiques afin de mieux caractériser la scène et l'objet ou d'utiliser un réseau pré-appris à notre tâche et l'adapter en ligne.

Remerciements

Je souhaite adresser mes sincères remerciements à toutes les personnes m'ayant aidé et appuyé tout au long de ces trois années de thèse.

En premier lieu, je remercie mes encadrants de thèse Stefan Duffner et Atilla Baskurt. Merci à Stefan de toujours être présent, de répondre à mes questions et à mes mails mêmes aux heures les plus tardives, ainsi que pour les nombreuses corrections et débogages. Merci à Atilla d'avoir été présent malgré son programme chargé, de m'avoir poussé et aidé durant toutes les épreuves. Merci à vous pour votre soutien, vos conseils, vos encouragements, et merci à vous de m'avoir donné l'opportunité de travailler avec vous.

Je tiens également à remercier l'INSA pour m'avoir accueilli durant ces 8 années, ainsi que les membres LIRIS dans notre petit couloir au bâtiment Jules Vernes, j'y garde de précieux souvenirs.

Je souhaite également remercier, en toute évidence, ma famille: Mama, Sara et Steve. Merci d'avoir été patients avec moi tout au long de ces années, de m'avoir soutenu même de loin et d'avoir pris le temps de relire mon manuscrit.

Je ne peux pas oublier Diandra, éternelle compagne de misère depuis nos années co-TP, quand on apprenait à peine à coder en C. Merci de toujours être présente pour prêter oreille à mes préoccupations, j'espère avoir été pour toi au moins un millième du soutien que tu as été pour moi. Et nous voilà bientôt toutes les deux docteurs, qui l'eut cru!

Merci à vous Aicha, Petru, Ana, Jauris, Nina, Nicolas et Prisca d'être toujours au rendez-vous pour le soutien moral et agrémenter ma vie d'un peu de folie, qu'est-ce que je ferais sans vous.

Et enfin, je dédie ce manuscrit à mon père qui m'a donné la soif d'apprendre et la curiosité scientifique. Merci pour tout.

Salma.

Contents

List of Figures

List of Tables

List of Abbreviations

AI	**A**rtificial **I**ntelligence
ALOV	**A**msterdam **L**ibrary of **O**rdinary **V**ideos
ASEF	**A**verage of **S**ynthetic **E**xact **F**ilters
CMAP	**C**onditioned **M**aximum **A** **P**osteriori
CNN	**C**onvolutional **N**eural **N**etworks
DFT	**D**iscrete **F**ourier **T**ransform
DSST	**D**iscriminative **S**cale **S**pace **T**racker
EKF	**E**xtended **K**alman **F**ilter
FFT	**F**ast **F**ourier **T**ransform
FHMM	**F**actorial **H**idden **M**arkov **M**odel
FoT	**F**lock **o**f **T**rackers
HMM	**H**idden **M**arkov **M**odel
HOG	**H**istogram **O**f **G**radients
HOC	**H**istogram **O**f **C**oulor
IFFT	**I**nverse **F**ast **F**ourier **T**ransform
JPDAF	**J**oint **P**robability **D**ata **A**ssociation
KCF	**K**ernelized **C**orrelation **F**ilter
KLT	**K**anade **L**ucas **T**omasi
LBP	**L**ocal **B**inary **P**atterns
MAP	**M**aximum **A** **P**osteriori
MCMC	**M**arkov **C**hain **M**onte **C**arlo
MHT	**M**ultiple **H**ypothesis **T**racking
MIL	**M**ultiple **I**nstance **L**earning
MOT	**M**ulti-**O**bject **T**racking
MOSSE	**M**inimum **O**utput **S**um of **S**quared **E**rror
NCC	**N**ormalised **C**ross **C**orrelation
OAB	**O**nline **A**da**B**oost
OOT	**O**nline **O**bject **T**racking
PCA	**P**rinciple **C**omponant **A**nalysis
PDF	**P**robabilistic **D**ensity **F**unction
PHD	**P**robability **H**ypothesis **D**ensity
PROST	**P**arallel **R**obust **O**nline **T**racking
ROI	**R**egion **O**f **I**nterest
SAMF	**S**cale **A**daptive with **M**ultiple **F**eatures
SDAE	**S**tacked **D**enoising **A**utoEncoder
SIFT	**S**cale **I**nvariant **F**eature **T**ransform
SIS	**S**equential **I**mportance **S**ampling
TLD	**T**racking-**L**earning-**D**etection
VOT	**V**isual **O**bject **T**racking
VTS	**V**isual **T**racker **S**ampler

Chapter 1

Introduction

1.1 Context

We, humans, use our eyes and brain to see and visually sense the world around us. We naturally inherit and improve our vision and interpretation of images and visual inputs at a very young age. From detecting objects, recognising faces to following moving objects, the human eyesight and brain processing are very powerful. Using stereoscopic vision, depth perception, shadow analysis and prior knowledge, human vision is capable of analysing and solving various types of complex vision processing problems. Yet, human vision is fallible as illusions and ambiguities can trick our minds.

With the advent of computers and numerical image acquisition systems, *Computer Vision* greatly flourished in the aim of modelling, replicating, and more importantly exceeding human vision using computer software and hardware. Computer vision is concerned with the automatic extraction, analysis and interpretation of useful information from a single image or a sequence of images to achieve automatic visual understanding.

Object tracking is one of the most important components in a wide range of applications in computer vision. Visual object tracking is defined as "the estimation of the trajectory of an object in the image plane as it moves around a scene." by Yilmaz *et al.* [YJS06]. A visual object tracker will aim to locate a target object or multiple objects throughout a video, either in the image plane (2D) or in the real world (3D).

The nature of target objects in tracking scenarios are as diverse as anything that is of interest for computer vision applications. For instance, target objects can be living beings (*e.g.* humans, animals), specific parts from human beings (*e.g.* faces, hands), rigid objects (*e.g.* cars, balls) or non-rigid objects (*e.g.* clothing, bags).

When looking at this task of visual tracking, the first question would be: how can we describe or model an object in an image? Human vision can easily segment and observe an object in its physical entity. However, for a computer, an object is represented by a cluster of pixels in an image.

In order to represent an object's appearance, a tracking algorithm needs to extract characteristic and pertinent information from these pixels. Classically, these low-level visual features capable of describing an object can be categorised into:

- *Edges*: Object boundaries usually generate strong changes in image intensities. These edges form contours and can help matching the object's shape.

- *Colour*: The object's colour distribution is a representation of its appearance that is often used to discriminate the object from the background.

- *Texture*: Texture is a measure of the intensity variation of the object's surface which quantifies its properties such as smoothness and regularity.

- *Optical Flow*: The field of displacement vectors of the object help to predict its next location.

Tracking algorithms usually use one or a combination of these features in order to learn an appearance model of the object.

The next question would be: how to represent the location or more generally the state of an object in the image? In order to visually track an object, a shape representation of the object is needed and because of the variety of objects that could be tracked, many object representations are possible and tailored to the target object's nature. Among the most common object shape representations, we can cite (*c.f.* figure 1.1):

- *Centroid*, suitable for small objects or objects occupying small pixel spaces, the object is presented by a point, usually the centroid of the object.

- *Bounding Box*, a simple rectangle centred on the object. It is a very common object representation when the object's shape or nature is not know a priori.

- *Skeletal Model*, commonly used on articulated objects (*e.g.* humans), the skeleton can be extracted by applying medial axis transform to the object silhouette.

- *Object Contour*, also called object silhouette, defines the boundary of the object through interconnected points. Contours are more suitable for tracking complex non-rigid objects.

The type of object representation is chosen according to application domain and nature of the target object. Using this representation and the corresponding image data, a tracking algorithm can construct an appearance model of the object with probabilistic models, templates, active appearance models or other appearance representations based one or more visual features.

1.2 Visual object tracking

There are numerous possible applications for visual object tracking algorithms as they are applied to a wide range of home, business, and industrial real-world scenarios.

Recently, intelligent and automated security surveillance systems have become an important field of application due to an increasing demand for such systems in public areas such as airports, underground stations and mass events. The automated detection and tracking of moving objects in surveillance video streams is of prime importance for these security systems monitoring human activities, monitoring traffic, counting pedestrians, *etc.*

FIGURE 1.1: Common object representations: (a) Centroid, (b) Bounding Box, (c) Skeleton and (d) Object Contour.

In dynamic environments, the tracking of humans and vehicles in a surveillance context is critical to fight against terrorism, crime, for public safety and for efficient management of traffic.

Human Computer interaction is also a current challenging research topic in computer vision. Detection and tracking of human body parts and activities is a key technology in understanding human behaviour and designing AI (Artificial Intelligence) systems. For instance, facial feature tracking is helpful for extracting visual cues to understand human emotions or for eye tracking to follow a person's visual attention. Hand gesture recognition can be used to control computer interfaces and the tracking and detection of cars, pedestrians and road signs, for example, is important for autonomous vehicle navigation such as cars or robots.

Each application scenario for visual object tracking differs in the nature of target objects to track, acquisition and execution conditions, or possible tracking environments. Different scientific approaches have been proposed in the literature depending on these conditions.

1.2.1 Online vs. Offline Tracking

Visual object tracking is often performed "online", *i.e.* solved as frames become available: knowing the state of the object at frame $t-1$, the tracker estimates the state of the object at frame t. Indeed, online visual tracking estimates the target state forward in time without using any observations from the future. Most previous research on tracking has been performed in the online scenario, since real-time applications require an instant response to changes in the real-world environment, such as in video surveillance [HHD00].

On the contrary, an offline tracker considers that the entire sequence of video frames is known a priori and thus can use future frames to estimate the object's current state. This is useful for video classification and indexation applications. Some applications that require high tracking accuracy employ interactive tracking systems with offline tracking (*e.g.* animation, CGI, video editing, video annotation). In an interactive system, a long video sequence can be decomposed into short ones by specifying a few key-frames [Aga+04; Sun+05]. Offline tracking can also be helpful in scenarios with significant occlusions [GZT11].

Online and offline approaches to tracking differ greatly in the learning process of the object's models and the a priori information available to it.

1.2.2 Multi-Object vs. Single Object

Multi-Object Tracking (MOT) has numerous applications in time-critical video analysis scenarios such as robot navigation and autonomous driving (*e.g.* Google self driving car [Goo]). Given an input video, the task of MOT is to locate multiple objects (generally of the same nature), maintain their identities and estimate their individual trajectories.

Recent progress on Multi-Object Tracking [LZK14] has focused on the tracking-by-detection strategy, where object detections from a detector (specialised of the nature of target objects) are linked to form trajectories of the targets. Either in an offline framework [LT+14; BC13] or online framework [XAS15; BY14], the main challenge when tracking multiple objects is the data association of previously tracked objects to noisy object detections in the current video frame.

On the other hand, in single object tracking, the tracking algorithms focus on a single target. A majority of single object trackers operate online, either adaptively building a target model of the object each frame [GGB06] or using an off-line trained object detector tailored to a specific type of object [SWC09].

1.2.3 Stationary vs. Mobile Camera

Many practical computer vision systems assume a fixed camera environment, such as surveillance applications [Col+01; TLH05; HHD00]. Tracking methods for fixed cameras commonly use background subtraction methods to detect the moving objects. By building a representation of the scene, the background model, the target object can be detected by finding the deviations from the background model in each frame.

The number of cameras may also vary as some computer vision applications need multiple stationary cameras. For example, incorporating depth information using multiple cameras improves tracking in scenarios with occlusion and lack of visibility of target objects [MD03]. Multiple cameras are also to used to increase the view area [Col+01; LRS00], as a single camera cannot cover large areas because of a finite sensor field-of-view. The main challenge when using multiple cameras is the calibration of relationship between the different camera views.

The non-stationary camera tracking scenarios are more common in state-of-the-art trackers, as it applies to a wider range of computer vision applications. These methods do not assume a fixed background, making background

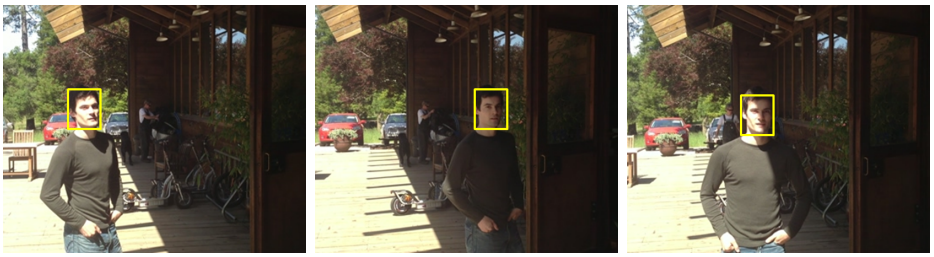


FIGURE 1.2: Scene illumination changes in "sunshade" video from the VOT2013 dataset [Kri+13].

subtraction techniques unsuitable. Discriminative models are commonly used to discriminate the object from the possibly changing background.

1.3 Objectives

In this thesis, we concentrate on solving the common problematic of *online*, *single-object* tracking with *non-stationary* cameras.

In the recent years, this visual tracking scenario became popular in literature, we can cite numerous recognised trackers like Struck [HSHT11], MIL [BYB11], DSST [Dan+14a], OAB [GGB06], KCF [Hen+15], *etc.* The increase of various tracking benchmarks [DG16] can also attest to, such as *Online Object Tracking* benchmark (OOT2013 [WLY13], also mentioned CVPR2013) or *Visual Object Tracking* benchmark (VOT2013 [Kri+13], VOT2014 [Kri+14], VOT2015 [Kri+15] and the ongoing VOT2016).

Despite extensive studies during the past several decades, tracking objects under unconstrained scenarios is still a complex and difficult task due to the many scientific challenges that we will briefly explain in the following section.

1.3.1 Challenges

The online visual tracking scenario in unconstrained environments raises many challenges and difficulties. Generally, any variation of visual cues either of the object or the scene makes the tracking task more complex. Moreover, fast changes of the object's appearance often causes the tracking models to drift because they are not able to quickly adapt to the changes.

- Scene illumination

  Changes in the scene illumination directly affect the appearance of an object as illustrated in figure 1.2. Not only changes in lighting intensity but also lighting direction disturb the object's appearance as the light casts different shadows depending on its direction. In this context, pixel intensity or colour based are not sufficient to construct an object appearance model, while texture-based features are more likely to be robust to illumination changes.

- Object shape and appearance

  Rigid objects are usually easier to track because of their consistent appearance. However in-plane and out-of-plane rotations of the object can still

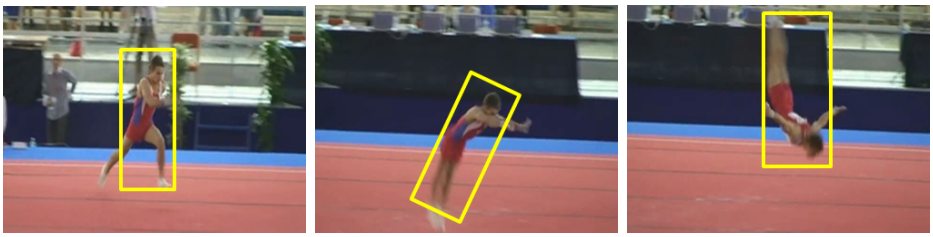


FIGURE 1.3: Object shape variations in "gymnastics1" video from the VOT2013 dataset [Kri+13].

FIGURE 1.4: Partial occlusions in "godfather" video from the VOT2015 dataset [Kri+15].

considerably change the appearance in a 2D image. On the other hand, deformable object, like humans, can greatly vary in shape and appearance depending on the object's movements (figure 1.3). Shape modelling can be difficult with such variations, however models, for example, based on colour distributions are less affected by these changes and can help to localise the object.

- Partial occlusions

  Short time total occlusions or partial occlusions occur frequently in real-world videos with a high density of moving objects (figure 1.4). They can be caused either by the object itself (hand movements in front of a face) or by neighbouring objects. It is a difficult task to handle occlusion because they corrupt the online learned object model and can cause the tracker to drift.

- Background

  Complex backgrounds, or textured backgrounds can be challenging as similar patterns or colours to the object can confuse the trackers and cause them to drift. Moreover, other objects similar to the target object can be present in the scene, for example in a basketball video figure 1.5 where all the players from the same team have the same clothing.

- Camera motion

  In real-life videos, the camera motion tends to follow the main target object. However, when the videos are taken by a small consumer camera (like a mobile phone), we can observe a lot or trembling, and jitter causing motion blur in the images or abrupt zooming as in figure 1.6. Rapid movements of the object can also have similar effect on the quality of the video.

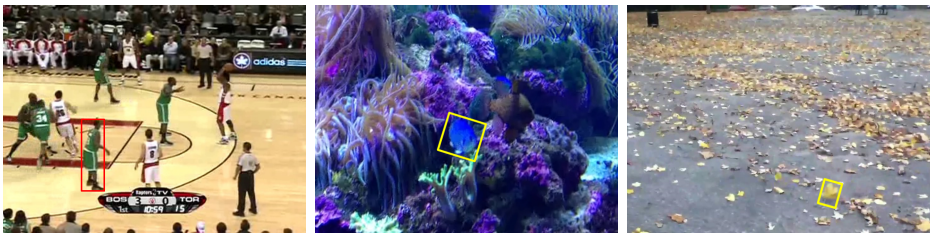


FIGURE 1.5: Complex scene backgrounds in "basketball", "fish1" and "leaves" videos from the VOT2015 dataset [Kri+15].

FIGURE 1.6: Camera motion and zoom in "woman" video from the VOT2013 dataset [Kri+13].

1.3.2 Motivation

Due the the varying conditions of online visual tracking, it is difficult to design a tracking algorithm capable of being robust to each one of these challenges.

We start from the observation that different tracking algorithms have different strengths and weaknesses depending on the environments and context. Tracking algorithms are generally specialised on different image conditions depending on the visual features used making the tracking more accurate in certain situations, but prone to drifting in other situations. Thus combining multiple modalities and tracking algorithms can considerably improve the overall tracking performance in unconstrained environments.

In this thesis, we propose several novel tracking frameworks based on this tracker combination approach, which consists of recurrently selecting one tracker from a pool of complementary trackers at a given point in time. The choice of individual trackers as well as the selection criteria is essential and critical to the overall tracking performance of the approach. We explore different types of selection criteria and especially the use of scene context.

1.4 Thesis Outline

In the following chapter, we first outline some of the most important tracking algorithms in the literature, categorising them into four main types of trackers: probabilistic, tracking-by-detection, correlation-based and feature point tracking methods.

In chapter 3, we detail the different tracker combination frameworks in the literature and then introduce a new tracker selection framework using a spatial and temporal coherence criterion.

In chapter 4, we focus on the use of scene context in the tracker selection process. We design effective visual features, extracted from the scene context and train a classifier based on these features to predict the tracker that will perform best under the given scene conditions.

In chapter 5, we extend this scene context based tracker selection framework by exploring the use of different individual trackers and training scenarios.

In chapter 6, we started exploring one possible perspective which is the use of a Convolutional Neural Network to automatically learn to extract these scene features directly from the input image and predict the most suitable tracker.

Finally chapter 7 will conclude this work with a short summary over the different contributions and some perspectives on future research directions.

Chapter 2

Visual Object Tracking, An Overview

In this section, we will present an overview of the literature in visual object tracking, concentrating on single object online tracking. The purpose of any object tracking algorithm is to approximate the path of an object (and potentially its size) in the image plane as it moves around a scene by finding its position in every frame of the video.

The tasks of detecting the object and establishing correspondence between the object instances across frames can either be performed separately, jointly or in a probabilistic manner. We can then group tracking algorithms mainly in the three following categories: *probabilistic methods*, *tracking-by-detection* algorithms and *correlation-based tracking* algorithms.

Probabilistic methods formulate tracking as the correspondence of the target object across frames and take into account the possible perturbations and noise in the video. *Tracking-by-detection* algorithms use an object detection algorithm in order to obtain possible object regions in every frame and then correspond objects across frames. On the other hand, *correlation-based tracking* algorithms require detection only when the object first appears in the scene, and they jointly estimate the object region and correspondence between frames by iteratively updating object region information obtained from previous frames.

Of course this classification of tracking methods in three groups is not exclusive. Some methods will mix different strategies like probabilistic methods using tracking-by-detection techniques or correlation based models used as a prediction function.

We will start by discussing *probabilistic methods* in section 2.1, *tracking-by-detection* methods in section 2.2, and *correlation-based tracking* in section 2.3. We will also further discuss *feature point tracking* methods in section 2.4

Moreover, we will detail later the various *tracker combination* possibilities in chapter 3, the use of context in tracking in chapter 4, and finally deep learning applied to tracking in chapter 6.

2.1 Probabilistic Methods

Object tracking can be viewed as the probabilistic estimation, or prediction, of the future state of an object. While deterministic methods use qualitative motion heuristics to constrain the correspondence of an object from a frame to the next, probabilistic methods explicitly take the object measurement

and uncertainties into account to establish correspondence. These uncertainties can be caused by video measurements containing noise or random perturbations of object motion.

Probabilistic correspondence methods use the state space approach to model the object properties. In general, state estimation methods can be used to estimate the state of any time varying system, such as tracking contours [IB98a], activity recognition [VCC03] or object identification [ZCM03].

Most probabilistic methods for tracking, as those outlined in the following, use a type of (infinite) Markov chain to model the temporal evolution and Bayesian inference techniques. In contrast to tracking-by-detection methods, they are non-deterministic, and they allow for an easy integration of different object motion and appearance models. In the following, we will briefly explain the two most common probabilistic methods used for tracking: the Kalman filter and the Particle Filter.

2.1.1 Kalman Filter

The Kalman filter [WB95] is an Optimal Recursive Data Processing Algorithm that provides efficient computational means to estimate the state of a process. The Kalman filter is used to estimate the state of a linear system where the state is assumed to follow a Gaussian distribution.

A process can be estimated by the filter by using a form of feedback control in two steps: prediction and correction. In the prediction step, the state model is used to predict the new state of variable. Then in the correction step, the current observations are used to update the object's state.

Kalman Filtering has been extensively used for tracking in early works. For instance, Broida and Chellappa [BC86] used the Kalman filter to track points in noisy images, while Rosales and Sclaroff [RS99] used an extended version of the Kalman filter to estimate a 3D object trajectory from 2D image motion. Boykov and Huttenlocher [BH00] employed the Kalman filter to track rigid objects based on an adaptive Bayesian recognition technique that incorporates dependencies between object features. In this method, a maximum a posteriori (MAP) estimate of the object parameters is found at each frame, then the selection of data points in each frame allows temporal fusion via Kalman filtering.

The Kalman filter, however, often encounters false matches and needs constraints to reject observations with large deviations. Moreover, the filter is restricted because of its linear models which can be non-linear in the Extended Kalman Filter (EKF) [May82] and because of the assumption that the state variables are normally distributed. This limitation can be overcome with particle filtering.

2.1.2 Particle Filtering

Particle filtering was first introduced in computer vision as the Condensation algorithm by Isard and Blake [IB98b]. Particle filters are sequential Monte Carlo methods based on point mass (or "particle") representations of probability densities, which can be applied to any state-space model and generalise the traditional Kalman filtering methods. In particle filtering, the object state is represented by a set of samples (particles) with weights. These weights define the importance of a sample, and the set of particles and

their associated weights can be viewed as an approximation to the probability distribution. The basis of particle filtering lies in sequentially updating a distribution using importance "sampling techniques" (see [Aru+02] for more details), the most common being the Sequential Importance Sampling (SIS).

The basic Particle Filter algorithm consists of two steps: sampling and selection. In Sequential Importance Sampling, random particles are sampled from the prior distribution, and for each particle, the corresponding importance weight is evaluated and normalised. Then in the selection step, particles with high or low importance weights are multiplied or discarded. Finally, new object position is estimated using the new particles.

Particle filtering became very popular in tracking applications for its robustness and adaptivity. Pérez *et al.* [Pér+02] introduced a probabilistic framework based on colour histograms, where the use of a particle filter allows the tracker to better handle colour clutter in the background, as well as complete occlusion of the tracked object over a few frames. Similarly, Nummario *et al.* [NKMVG03] presented an adaptive particle filter integrating colour histograms. An initialisation based on an appearance condition is also introduced to manage objects disappearing and reappearing. Later, Badrinarayanan *et al.* [Bad+07] presented a probabilistic multi-cue tracking approach constructed by employing a novel randomised template tracker and a constant colour model-based particle filter.

Particle filtering, and Kalman filtering, can also be extended to multiple object tracking by defining a joint solution of data association. Joint Probability Data Association (JPDAF) and Multiple Hypothesis tracking (MHT) are two common data association techniques (see [YJS06] for more details). In the context of multiple target tracking, Okuma *et al.* [Oku+04] combined two element: mixture particle filters to assign a mixture component to each object in the scene and track them, and AdaBoost to generate detection hypotheses of new objects entering the scene. Yang *et al.* [YDD05b] also tackled multi-object tracking with a the particle filter based on both colour and edge orientation histogram features, where the features are computed using integral images for computational efficiency allowing the use of a large number of particles.

However efficient, probabilistic methods can be cumbersome to implement because of the hyper-parameters to set and noise covariance matrices to estimate.

2.2 Tracking-by-detection

Tracking-by-detection has been a focus of recent work [GB06; Avi07; Oku+04; HSHT11] in the tracking community. This has been facilitated by the impressive advances in object detection and machine learning methods, [DT05; LSS05; VJ01; VJ04], as they inspired many tracking algorithms.

In tracking-by-detection algorithms, objects are detected in consecutive frames by an external object detection mechanism and the association of the objects is based on the previous object state which can include object position and motion. In tracking-by-detection methods, two components are of importance: visual representation and modelling.

Various visual representations of the object are used in practice, for instance: intensity [Ros+08], colour [Pér+02] , texture [Avi07], Histogram of

Gradients [Tan+07], Haar-like features [BYB11], LBP [GGB06], *etc.* As for the models used in tracking-by-detection, we regroup the different strategies into the following categories: *generative* and *discriminative* modelling.

2.2.1 Generative Models

Generative modelling-based trackers concentrate on learning the target object's appearance in order to find similar regions in each video frame. While template-based trackers tend to model only a single appearance of the object, the generative models have been proposed to model more object appearance variations. The generative models can be learned offline [BJ98], or online [Ros+08; KL10]. Black and Jepson [BJ98] used eigenvectors generated from rectangular object templates to represent the appearance. On the other hand, Ross *et al.* [Ros+08] incrementally learn a low-dimensional subspace representation, efficiently adapting online to changes in the appearance of the target with a model based on Principal Component Analysis (PCA).

However, generative trackers only model the appearance of the object, without taking into consideration the background information, which leads to model drifting and failing at tracking fast changing objects or when facing cluttered backgrounds.

2.2.2 Discriminative Models

Discriminative models alleviate the problems encountered by generative models by conjointly modelling the target object and the environment where the object moves. A discriminative model treats the object tracking as a classification problem, the aim being to distinguish the object from the background. The environment is considered as a negative class against which the tracker should discriminate, making the tracking more robust in complex scenarios. Common discriminative trackers build a binary classifier to distinguish target pixels from the background pixels, and updates the classifier with new samples coming in, as shown in figure 2.1. Methods based on the discriminative modelling approach rapidly expanded in the recent years due to their robustness [Kri+16; Sme+14] and different classification strategies were explored.

Boosting-Based Learning

Many boosting-based discriminative trackers [CL05; Avi07; GB06; BYB09] were presented in literature, concentrating on the adaptive aspect of the classifier, building the object classifier during rather than before tracking. The essential phase of adaptive discriminative trackers is the update. Positive training samples are extracted from the close neighbourhood of the current object location and the negative samples from distant surrounding of the object. These training samples are then used to update the classifier in every frame. It has been demonstrated that this updating strategy handles significant appearance changes, short-term occlusions, and cluttered background.

The Ensemble Tracker by Avidan [Avi05; Avi07] adopted the AdaBoost algorithm [FS95] where an ensemble of weak classifiers is trained online and combined into a strong classifier to distinguish between the object and the background. This method breaks the complex training phase into a set of

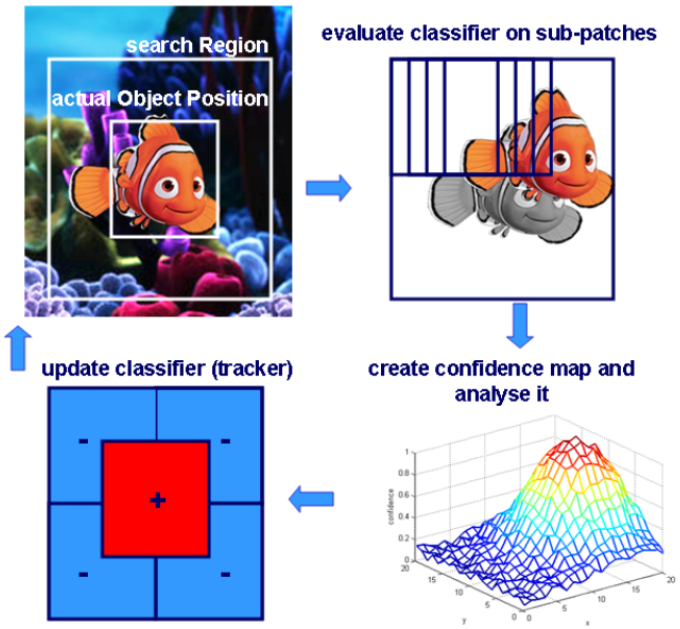


FIGURE 2.1: Principle of discriminative tracking-by-detection with a classifier. Image courtesy of [GB06].

simple and easy-to-learn tasks, *i.e.* weak classifiers, which can be computed online. As shown in figure 2.2, the ensemble of weak classifiers is trained to separate pixels that belong to the object from those that belong to the background and used to create a confidence map of the pixels in the current frame. The peak of the map, *i.e.* the new position of the object, is found using mean-shift. Finally, temporal coherence is maintained by updating the ensemble with a new weak classifier trained online on the current frame.

There has also been considerable work along these lines, using boosting in a tracking framework. Collins and Liu [CL05] proposed a method to adaptively select colour features that best discriminate the object from the current background. Lim *et al.* [Lim+04] used incremental subspace learning for tracker updating. Furthermore, Grabner and Bischof [GB06] proposed the Online AdaBoost tracker (OAB), also applying the AdaBoost classifier to an online feature selection framework. This tracking algorithm will be further detailed in section 3.3.2.

Semi-Supervised Learning

Online boosting-based methods, although discriminative, suffer from drifting because of the possible accumulated errors of the online updated weak classifiers. To address these problems, semi-supervised learning can be used to constrain the update of the tracking classifier by an auxiliary classifier, either trained in the first frame [GLB08] or by training a pair of independent classifiers [Tan+07; YDM08].

Grabner et al. [GLB08] proposed the SemiBoost tracker to explore the continuum between a fixed detector and online learning methods. The update process is formulated in a principled manner as the combined decision of

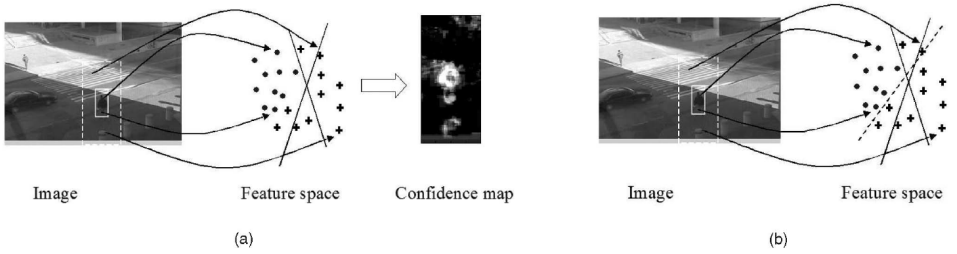


FIGURE 2.2: Ensemble Tracker [Avi05; Avi07] update (a) and test (b) phases. (a) The pixels of image at time $t-1$ are mapped to a feature space (circles for positive examples and crosses for negative examples). Pixels within the solid rectangle are assumed to belong to the object, pixels outside the solid rectangle and within the dashed rectangle are assumed to belong to the background. The examples are classified by the current ensemble of weak classifiers (denoted by the two separating hyperplanes). The ensemble output is used to produce a confidence map that is fed to the mean shift algorithm. (b) A new weak classifier is trained (the dashed line) on the pixels of the image at time t and added to the ensemble. Image courtesy of [Avi07].

a given prior and an on-line classifier in a semi-supervised learning [LGB08] fashion. Two types of data are used: labelled data (or a previously trained model) as a prior and unlabelled samples collected during tracking as seen in figure 2.3. The knowledge from labelled data is used to build a fixed prior for the on-line classifier. In order to still be adaptive during tracking, unlabelled data is used to update the classifier in a unsupervised manner. The SemiBoost tracker is able to limit the drifting problem as it retains the prior classifier during the tracking procedure while still staying adaptive to appearance changes. However, the SemiBoost algorithm discards information by leaving all extracted image unlabelled, except for the first frame, which leads to poor performance in the presence of drastic appearance changes.

Inspired by co-training, Tang *et al.* [Tan+07] proposed a semi-supervised learning algorithm for the update mechanism in the tracking framework. In a similar way to [GLB08], this method uses a small number of labelled samples for initialisation, then each new sample is treated by semi-supervised learning as unlabelled data. Using multiple online SVMs based on independent features (colour histograms and histograms of oriented gradients (HOG)), the prediction from the different features are fused. By combining the confidence maps from each classifier, a final classifier is created and the classification of new data and updating of the classifier are achieved simultaneously in a co-training framework.

Multiple Instance Learning

To overcome the drifting problematic of adaptive discriminative methods, Babenko *et al.* [BYB09; BYB11] proposed a MIL (Multiple Instance Learning, [DLLP97]) based appearance model for object tracking. The MILTrack method learns a discriminative classifier from positive and negative bags of samples. Samples are collected from the target bounding box with other neighbouring windows into the positive bag. A bag is considered as positive if it contains at least one positive instance, otherwise the bag is set to negative (*c.f.* figure 2.4).

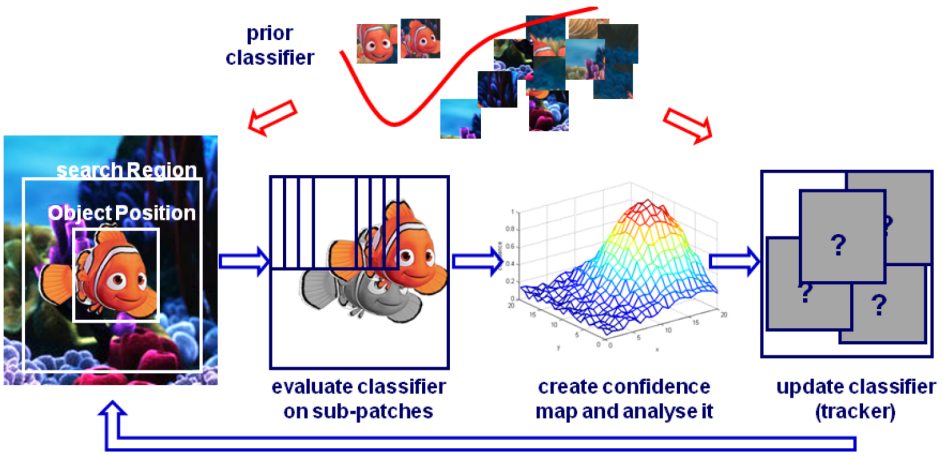


FIGURE 2.3: General framework of the SemiBoost tracker [GLB08]. Given a fixed prior and an initial position of the object in time t, the classifier is evaluated at many possible positions in a surrounding search region in frame $t+1$. The obtained confidence map is analysed in order to estimate the most probable position and finally the classifier is updated in an unsupervised manner, using randomly selected patches. Image courtesy of [GLB08].

By using these bags of samples to update the classifier, the MILTrack algorithm is able to successfully handle ambiguously labelled training examples, rather than use only one positive sample with the risk of it being suboptimal.

Support Vector Machines

Support Vector Machines (SVM) have very successful in object detection algorithms, due to their good generalisation ability, robustness to label noise, and flexibility in object representation through the use of kernels [BL08; Fel+10; Ved+09].

An early static discriminative tracker SVT (Support Vector Tracking) was proposed by Avidan [Avi04] which integrates the Support Vector Machine (SVM) classifier into an optical-flow-based tracker. The SVM classifier is trained offline to track vehicles over long video sequences, which constrains the framework to offline learning and limits the tracking to known objects.

Hare *et al.* [HSHT11] made use of SVM's flexibility and natural generalisation to structured output spaces and proposed the Struck tracker. Using the structured output SVM framework of Tsochantaridis *et al.* [Tso+05], they extended the online structured output SVM learning method [Bor+07; BUB08] and adapted it to the tracking problem. As opposed to the common adaptive tracking-by-detection methods which separate target localisation (*i.e.* sample labelling) and model update, the Struck framework integrates the two steps of labelling procedure into the learner in a common framework as we can see in figure 2.5. This separation raises some issues such as error introduction in the labelling step. In fact, when sampling patches around the object to update the classifier, the labelling of these samples usually relies on pre-defined rules of the distance of a sample from the estimated object location. Moreover, because the two objectives of predicting the label of patch and estimating the object location are not coupled during training,

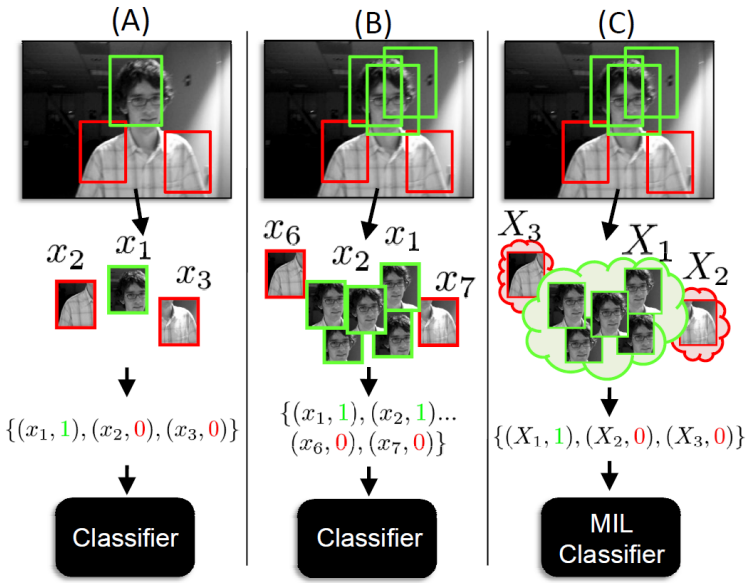


FIGURE 2.4: Comparison of different update strategies of discriminative appearance models: (A) Using a single positive image patch to update a traditional discriminative classifier. The positive image patch chosen does not capture the object perfectly. (B) Using several positive image patches to update a traditional discriminative classifier. This can confuse the classifier causing poor performance. (C) Using one positive bag consisting of several image patches to update a MIL classifier. Image courtesy of [BYB09].

the assumption that the maximum classifier confidence corresponds to the best estimate of object location may not be true. Struck integrated these two steps into one unified structured output learning framework and proved its efficiency by outperforming discriminative tracking such as Online AdaBoost [GGB06] or MILTrack [BYB09].

Segmentation

Tracking-by-detection approaches commonly use a bounding-box representation with fixed aspect ratio. Although effective, these methods can be limited by the use of bounding boxes when handling objects undergoing large deformations as it can increases the amount of noise introduced during online updating. To overcome the limitations of a rigid bounding box, some approaches integrate some form of segmentation into the tracking process.

For instance, Nejhum *et al.* [NHY08] proposed to track articulated objects with a set of independent rectangular blocks. They use a rough segmentation to find the object outline and re-arrange the blocks to maximise the overlap and similarity to the current object appearance and shape.

Godec *et al.* [GRB13] introduced segmentation in a discriminative classifier based on the generalised Hough-transform for tracking non-rigid targets. This framework aims to locate the support of the target through back-projection from a Hough Forest [Gal+11]. Using the first frame, a graph-cut algorithm is initialised to segment foreground by back-projecting the patches that voted for the object centre. The Hough Forest provides a probability map of the target and the pixel with the maximum score is selected as the centre of the target. The resulting segmentation is then used to update

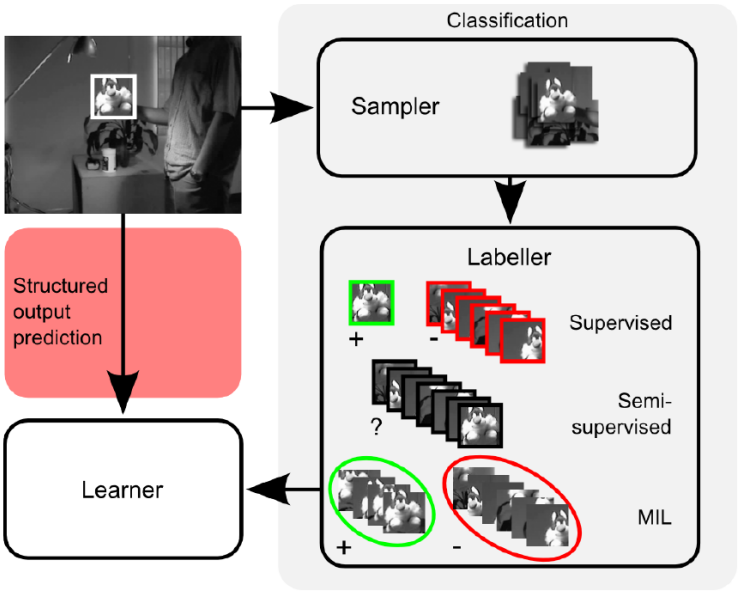


FIGURE 2.5: Different adaptive tracking-by-detection paradigms comparing Struck to supervised, semi-supervides and MIL classification techniques. Given the current estimated object location, traditional approaches (shown on the right-hand side) generate a set of samples and, depending on the type of learner, produce training labels. The Struck approach (left-hand side) avoids these steps, and operates directly on the tracking output. Image courtesy of [HSHT11].

the patches' foreground and background probabilities in the Hough forest. The segmentation result delivers a more precise description of the object than a simple bounding box representation. However, due to the graph-cut segmentation it is relatively slow.

Based on the same idea, Duffner and Garcia [DG13] extended this framework to do pixel-level Hough-based classification. A detector is implemented to make use of the generalised Hough transform with pixel-based descriptors. Then a probabilistic segmentation method based on global models for foreground and background is used to incrementally update the local pixel-based descriptors and vice versa in a co-training manner.

The local Hough-voting model and the global colour model operate both at the pixel level, allowing an efficient model representation of the object and further improving the tracking performance of [GRB13] especially on small regions.

Deep Learning

Deep learning became a recent trend in the classification community for its discriminant power and have obtained impressive results on the ImageNet dataset [KSH12]. It has also been successfully applied to visual tracking. In fact, in the recent VOT2015 [Kri+15] challenge, many deep learning-based trackers competed. Moreover, the two best performing methods in robustness and accuracy: MDNet [NH15] and DeepSRDCF [Dan+15] were both based on Convolutional Neural Networks. We will further detail the use of deep learning in the context of object tracking in chapter 6.

2.3 Correlation-Based Tracking

Correlation is an intuitive way to match an target object in the scene. The idea is to built a template of the object appearance, and search in every new frame for a the highest resembling patch, *i.e.* the patch with the highest correlation, to the template. In these methods, we typically need to evaluate a model on each candidate position in the new frame. This search procedure can be efficiently executed either using the FFT (Fast Fourier Transform) or by an iterative search. In fact, the correlation operation, *i.e.* convolution function, in the time domain is transformed into a simple multiplication in the frequency domain.

2.3.1 Simple Template Matching

Template matching is a basic approach for correlation-based tracking, it is a brute force method of searching the image for a similar region to the object template. First, the reference template is extracted form the initial frame's object bounding box. Subsequently, the method examines the Region of Interest (ROI) of each frame and samples uniformly candidate templates around the ROI. Each candidate templates is compared to the reference template and by maximising a similarity function, such as cross correlation, the candidate with the highest score, *i.e.* highest similarity, is selected as the new target location. This approach is referred to as the NCC (Normalised Cross Correlation) tracker. In terms of features, image intensity of colour are usually used to form the template. Birchfield *et al.* [Bir98] proposed image gradients as features to contour the sensitivity of image intensity features to illumination changes.

With the exhaustive template extraction and matching, this method becomes computationally expensive, especially when the template size or the search region is large. However, more efficient algorithms for template matching have been proposed [BH01; SBW02]. The NCC also acts as one of the important components in more advanced trackers such as the PROST tracker [San+10] or TLD tracker [KMM12].

2.3.2 Mean-Shift Tracking

Template-based matching methods typically lack robustness to track non-rigid objects, moreover it is difficult to integrate multiple features in the NCC framework. Instead of templates, other representations can be used for more robust tracking such as histograms or mixture models. Mean-shift tracking employs histograms in order to find the regions of a video frame that are most similar to a previously initialised model. Moreover, mean-shift-based trackers eliminate of the brute force search of the simple template matching process. In fact, a gradient ascent procedure is used to move the tracker to the location that maximises a similarity score between the model and the current image region.

Based on the mean-shift algorithm [FH75], Comaniciu *et al.* [CRM00; CRM03] designed a mean-shift tracker to perform matching with colour histograms computed on a circular region representing the object. This tracker uses the target colour distribution formed in the first frame. Then for each new frame, the colour histogram of the object is compared with candidate histograms from the ROI by maximising iteratively the similarity base on

the Bhattacharyya metric. In order to find the best target location in the new frame, mean shift is used to find the mode of a function, the one which maximises the Bhattacharyya distance.

Colour histograms are invariant to object shape changes, however to increase the robustness when facing regions with similar colour distributions, spatial information can be introduced to the mean-shift framework. Comaniciu *et al.* [CM02] extended his previous work by using a joint spatial-colour histogram. Yang *et al.* [YDD05a] also integrated spacial information into the mean-shift framework by defining a new and discriminative similarity function and reduce the computational complexity with a fast Gauss transform.

2.3.3 Correlation Filters

Correlation filter-based discriminative trackers have made significant achievements in the recent years and have received increasing interest in the tracking community [CHT15] due to their simple structure, efficiency and performance. Conventionally, in image processing, correlation filters are used as detectors of expected patterns. They are designed to produce correlation peaks for each interested predefined pattern in the scene while generating low responses to background.

In fact, correlation filters have performed effectively in the context of object detection and localisation. For instance, Kumar *et al.* applied them to iris verification [KXT03], Bolme *et al.* to human detection [Bol+09] and Savvides *et al.* to face verification [SKK02]. However, the generalisation of correlation filters to online tracking is not easy since the required training makes them inappropriate for online tracking. For instance, the Average of Synthetic Exact Filters (ASEF) filter introduced by Bolme *et al.* [Bol+09] in the context of online detection, averages all the trained exact filters to obtain a general one. The resulting filter is robust in the context of object detection, but requires a large number of samples for training, which is not suitable for the online tracking task.

To overcome this and reduce this data requirement, the Minimum Output Sum of Squared Error (MOSSE) filter was proposed by Bolme *et al.* [Bol+10], producing stable correlation filters when initialised using a single frame. Using an adaptive training scheme, MOSSE is considerably robust and performs at high frame-rate in online tracking.

The general framework of a correlation filter-based tracker is illustrated in figure 2.6. After the initialisation of the correlation filter with the object patch from the first frame, the image patch of the estimated object position from the previous frame is used as input. Various visual features can then be extracted from the input and a cosine window is usually applied for smoothing the boundary effects. Subsequently, the correlation between current input and the learned filter is computed in the frequency domain using the convolution and Fast Fourier Transform FFT (in practice, the Discrete Fourier Transform DFT is used). A spatial confidence map is obtained by Inverse FFT (IFFT), whose peak can be predicted as the new position of target. Finally, the newly estimated position is used to update the correlation filter. The training and updating scheme of the correlation is what usually differs in the different versions of the filter.

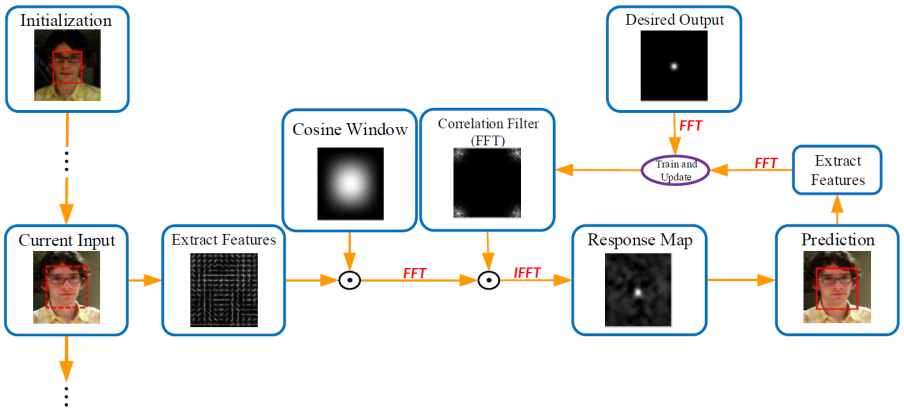


FIGURE 2.6: General framework of correlation filter-based online tracking. Image courtesy of [CHT15].

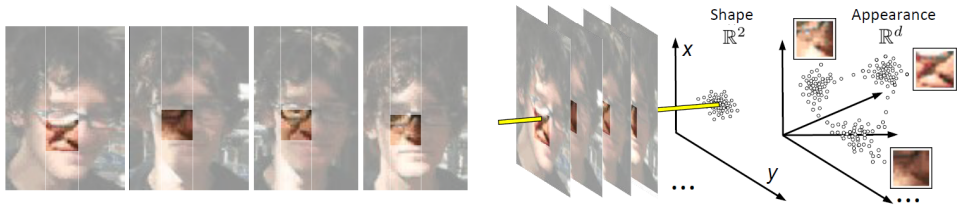
Based on the basic framework of MOSSE filter, further improvements have been made to correlation-based tracking. Henriques et al. [Hen+12] improved the MOSSE filter by introducing kernel methods, Danelljan et al. [Dan+14b] applied colour-attributes to better represent the input data. The KCF tracker will be further detailed in section 5.2.1. Scale invariance was handled in correlation-based tracking in the SAMF tracker [LZ14], DSST tracker [Dan+14a] and the improved KCF tracker [Hen+15], by proposing a multi-scale search strategy. Correlation filters-based trackers have proved great efficiency and robustness, and achieved state-of-art results in the VOT2014 challenge [Kri+14].

2.4 Feature Point tracking

In an image structure, moving objects can be represented by local feature points, and tracking can be formulated as the correspondence of detected features points across frames. Although efficient and robust algorithms exist for the detection of feature points or interest points, determining their correspondence is a complicated problem-specially in the presence of occlusions, false detections, entries and exits of objects.

2.4.1 KLT Tracker

A classical approach to tracking an object is computing its translation from one frame to the next with an optical flow method. Based on Optical Flow [LK81], Shi and Tomasi introduced the KLT (Kanade-Lucas-Tomasi) tracker [ST94] which iteratively computes the translation of the target object region its new location in the following frame. The tracker matches the candidate patches to the target bounding box in consecutive frames by computing an affine transformation of feature points in the image. The features points can be represented by edges and corners in object template. In order to deal with scale, rotation, translation and to make the template matching locally smooth, the affine transformation is computed by incremental image alignment based on spatio temporal derivatives and warping. Finally, the



(a) (b)

FIGURE 2.7: Representation of the weakly aligned multi-instance local features of the ALIEN tracker [PDB14]. (a): Four frames with highlighted appearance variations in a particular object region.(b): Region representation after weak alignment. Feature locations describing 2D shape in the xy-coordinate system of the object template are shown with their associated appearance descriptors (128D). Image courtesy of [PDB14].

new object location is determined by mapping its position in the previous frame to the current one using affine transformation.

This KLT tracker has been improved in several aspects. Bouget [Bou01] integrated a coarse-to-fine optimisation scheme on an image pyramid to deal with larger motion between two consecutive frames. A computationally efficient version of the KLT tracker was proposed by Baker and Matthews [BM04]. Kalal *et al.* [KMM10b] constructed a forward-backward KLT tracker to automatically detect tracking failures.

2.4.2 ALIEN Tracker

Pernici and Del Bimbo introduced the ALIEN tracker [PDB14] which exploits oversampling of local invariant representations to build a robust object-context discriminative classifier discriminating between the object and the context (*i.e.* background). The object is represented by multiple instances of scale invariant local features weakly aligned along the object template as presented in figure 2.7. A non parametric learning algorithm based on the transitive matching property discriminates the object from the context and prevents improper object template updating during occlusion. This local features representation of the object allows taking into account the 3D shape deviations from planarity and their interactions with shadows, occlusions and sensor quantisation for which no invariant representations can be defined.

2.4.3 FoT Tracker

The visual tracking problematic can be viewed in a different way, where the object position is estimated by combining displacement estimates from a subset of local trackers (*i.e.* feature points) that cover the object. This Flock of Trackers (FoT) concept was discussed by Kolsch and Turk [KT04] and Kalal *et al.* [KMM10b; KMM12] and showed robust short-term tracking results.

The block structure of the FoT framework is described in figure 2.8. Each local tracker is attached to a certain area specified in the object coordinate frame and generally the Lucas-Kanade Optical Flow [LK81] algorithm is used for local tracking. The FoT usually requires two components: a local

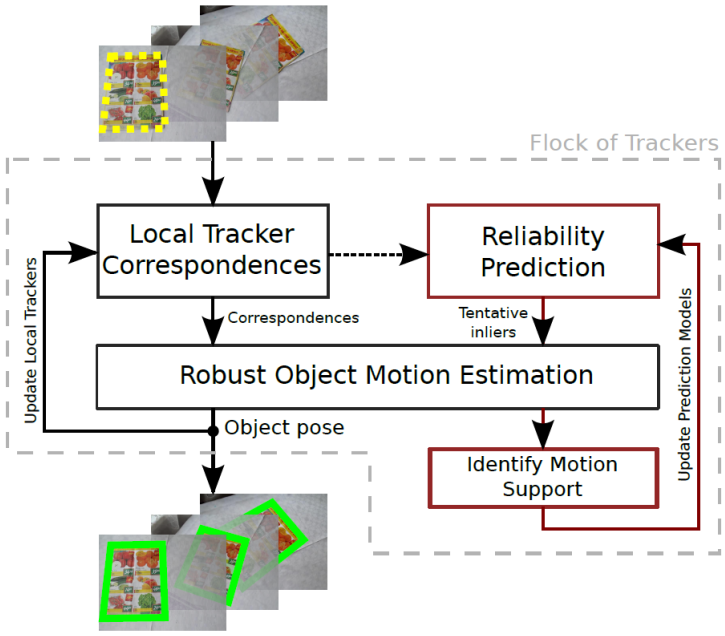


FIGURE 2.8: Block structure of the Flock of Trackers (FoT) [VNM14]. Correspondences (motion estimates) between two images, given the previous object pose and two consecutive images, are produced by local trackers. Simultaneously, reliability is estimated for each motion estimate. The object pose in the next frame is robustly estimated from a subset of most reliable motion estimates called tentative inliers. Image courtesy of [VNM14].

short-term tracker of which multiple instances are run on different areas of the object and provide image-to-image correspondences, and a (global) object motion estimation module robustly combining the local estimates.

Vojir *et al.* [VNM14] further enhanced the FoT framework with new reliability predictors for the local trackers (the Neighbourhood consistency predictor and the Markov predictor), new rules for combining the predictions and the introduction of a RANSAC-based estimator of object motion.

Feature point methods are easily adaptable to scale changes as the points are not dynamically attached to each other. However, with shape changes and object rotations, features points can be lost and new features point need to be added which can lead to drifting in tracking if the new feature points are falsely selected in the background.

2.5 Tracking Datasets

Many video dataset were collected and created throughout the years, in the aim of having a common testing or training ground for visual tracking algorithms. For instance, the Birchfield dataset [Bir], the CAVIAR dataset [Cav], The Cehovin dataset [Ceh], The FRAGtrack dataset [Fra], the MILtrack dataset [Mil], the PROST dataset [Pro], *etc.* Dubuisson and Gonzales provided an extended survey of tracking datasets in [DG16].

In this section, we will concentrate on describing the main dataset used, for either training or evaluation, during this thesis. Refer to Appendix A for snippets from these databases.

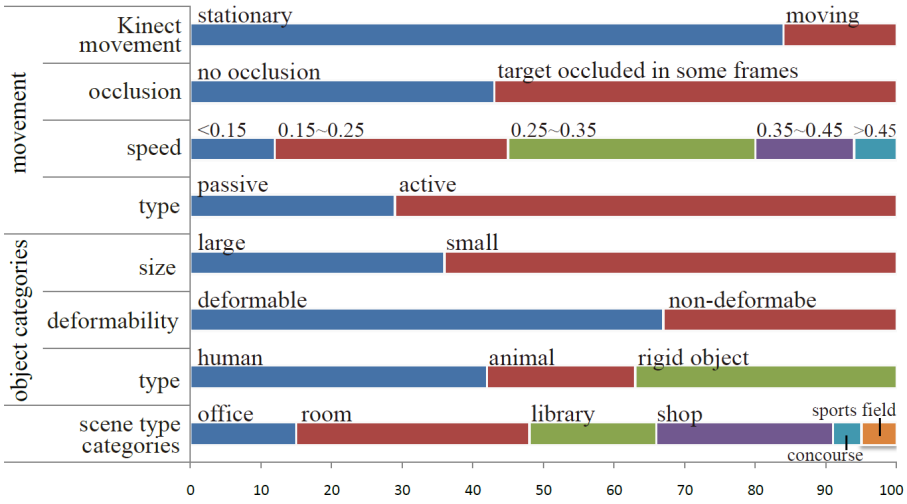


FIGURE 2.9: Statistics of the Princeton RGB-D tracking benchmark dataset. Image courtesy of [SX13].

2.5.1 ALOV++

The ALOV++ dataset [Alo], as in Amsterdam Library of Ordinary Videos dataset, was proposed by Smeulders *et al.* in his visual tracking experimental survey [Sme+14] in this aim of assessing the state-of-the-art in video object tracking, with an emphasis on the accuracy and the robustness of tracking algorithms. They composed a real-life dataset of 315 video fragments, focusing on one situation per video and covers diverse circumstances: illuminations, transparency, specularity, confusion with similar objects, clutter, occlusion, zoom, severe shape changes, different motion patterns, low contrast, *etc.*

To cover the variety of (dynamic) situations, preference was given to many assorted short videos over a few longer ones, with an average length of 9.2 seconds. The set is supplemented with ten longer videos, between one to two minutes each.

ALOV++ is also composed with videos from other tracking dataset [ESF09; CS10; KMM10a; BR11] or frequently used sequences in tracking evaluations and new videos. However, the main source of the data is real-life videos from YouTube with 64 different types of targets ranging from human face, a person, a ball, an octopus, microscopic cells, a plastic bag or a can. The collection contains also various difficult object to tracker, like a dancer, a rock singer in a concert, complete transparent glass, octopus, flock of birds, soldier in camouflage, completely occluded object and videos with extreme zooming introducing abrupt motion of targets.

The total number of frames in ALOV++ is 89364. The videos are annotated by a rectangular bounding box along the main axes of flexible size every fifth frame.

2.5.2 Princeton

Song and Xiao presented the Princeton Tracking Benchmark Dataset [SX13] to establish a unified benchmark for tracking algorithms with RGB-D data. It contains 100 RGB-D videos which includes deformable objects, various occlusion conditions, moving camera, and different scenes. The videos are

recorded with both RGB and depth data using a standard Microsoft Kinect 1.0, and each frame is manually annotated with ground truth bounding boxes.

As illustrated in figure 2.9, the dataset presents varieties in the different aspects. The target object is divided into three types: human, animal and relatively rigid object. Rigid objects, such as toys and human faces, can only translate or rotate. Animals include dogs, rabbits and turtles, whose movement usually consists of out-of-plane rotation and some deformation. The degree of freedom for human body motion is very high, which may increase the difficulty in tracking. Different scene types are present in the dataset with different levels of background clutter. The living room scene, for example, has a simple and mostly static background, while the background of a café is complex, with many people passing by. The videos also cover several aspects of occlusion, *e.g.* how long the target is occluded, whether the target moves or undergoes appearance change during occlusion, and similarity between the object occluding and target object. Moreover, the dataset covers a wide bounding box size distribution over all sequences, from long and short, to wide and narrow objects.

2.5.3 VOT2013

The Visual Object Tracking (VOT) challenge and workshop was organised in the aim of providing an evaluation platform for online tracking algorithms. The challenge started in 2013 (with VOT2013), and is held every year at the International Conference on Computer Vision (ICCV) or European Conference on Computer Vision (ECCV) workshops.

Dataset

The VOT2013 dataset is compiled of widely used sequences showing a balanced set of various objects and scenes and real-life visual phenomena. It contains a small number of sequences (16) to keep the time for performing the experiments reasonably low. All the sequences are labelled per-frame with different visual attributes to aid a less biased analysis of the tracking results: occlusion, illumination change, motion change, size change, camera motion. The total number of frames is 8394.

Benchmark

Along with the dataset, an evaluation kit is publicly available in Matlab/Octave which automatically performs the experiments on a tracker using the provided dataset. More than evaluating a tracker's performance, the benchmark allows the comparison between multiple trackers. The valuation protocol also explicitly addresses the statistical significance of the results and addresses the equivalence of trackers.

The benchmark evaluates two orthogonal performance measures: accuracy and failures. *Accuracy* measures how well the bounding box predicted by the tracker overlaps with the ground truth bounding box. The accuracy of a bounding box B_t with respect to the ground truth box B_t^{gr} is measured the same way as the F-score (*c.f.* equation 3.11):

$$accuracy_t = \frac{B_t \bigcap B_t^{gr}}{B_t \bigcup B_t^{gr}} \tag{2.1}$$

On the other hand, *Failures* is the failure rate measure, which counts the number of times (on average) the tracker drifted from the target and had to be reinitialised. It is a measure of robustness.

2.5.4 VOT2014

The VOT2014 challenge [Kri+14] follows the VOT2013 challenge and considers the same class of trackers: single-camera, single-target, online trackers, applied to short-term tracking. This benchmark provides an evaluation kit and a new dataset for automatic evaluation of tracking algorithms.

Dataset

The VOT2014 dataset was prepared using sequences form various tracking datasets such as the VOT2013 dataset, the ALOV dataset [Alo], or the Online Object Tracking Benchmark [WLY13] and additional sequences. From these, 25 sequences were manually selected such that the various visual phenomena like, occlusion or illumination changes, were still represented well within the selection.

The VOT2014 dataset is per-frame annotated with visual properties, while the objects are annotated with rotated bounding boxes to more faithfully denote the target position, which was not the case in the VOT2013 dataset.

Benchmark

The evaluation kit records the output bounding boxes from the tracker, and if it detects tracking failure, re-initialises the tracker. Then results are analysed by the VOT2014 evaluation methodology.

The new VOT2014 evaluation system incorporates direct communication with the tracker and offers faster execution of experiments and is backward compatible with VOT2013. The evaluation methodology from VOT2013 (accuracy and robustness) is extended to take into account that while the difference in accuracy of pair of trackers may be statistically significant, but negligibly small from perspective of ground truth ambiguity. A new unit for tracking new speed is introduced that is less dependant on the hardware used to perform experiments.

2.5.5 VOT2015

The latest edition of the VOT challenge, VOT2015 [Kri+15], follows the previous VOT2013 and VOT2014 challenges with the same goal of assessing tracking algorithms' performance through an evaluation system.

Dataset

The new fully-annotated VOT2015 dataset is introduced which doubles the number of sequences compared to VOT2014 to 50 sequences. The sequences are selected on based on new automatic dataset construction methodology

which focuses on the sequences that are considered difficult to track based on the visual attributes: illumination change, object size change, object motion, clutter, camera motion, blur, aspect-ratio change, object colour change, deformation, scene complexity and absolute motion.

The dataset is per-frame annotated with visual properties and the objects are annotated with rotated bounding boxes, as it was for VOT2014.

Benchmark

As in VOT2014, the two weakly correlated performance measures, accuracy and robustness, are used due to their high level of interpretability. The evaluation system from VOT2014 is extended for easier tracker integration and ranking.

2.5.6 ILSVRC15

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) is a benchmark in object category classification and detection on hundreds of object categories and millions of images. The latest edition, ILSVRC15 [Rus+15] presented a dataset with manually annotated training and validation images of 1000 different object classes.

The dataset is not directly designed for tracking but rather for image classification, single-object localisation and object detection tasks. However, providing a large set of annotated sequences and various objects. This dataset can be used to train or augment training datasets for object tracking.

Chapter 3

Tracker Selection Framework

3.1 Introduction

Visual tracking can face many challenges due to the unexpected variations in the object's shape, appearance, motion or the scene's illumination, as we discussed previously in chapter 1. In such conditions, the tracking task becomes difficult to generalise for all the possible scenarios. In order to face these challenges, one intuitive approach is to combine the strengths of multiples complementary trackers. In fact, visual tracking algorithms are generally specialised on different image conditions, *i.e.* an individual tracker will be more accurate in certain situations, but will lose the target object in other situations.

Figure 3.1 shows examples of variations in scene context that can occur during tracking. In the first video, we can see multiple changes in the background as the object moves across the scene. In this scenario, a discriminant tracker is needed to adapt to the changes of the background. In the second video, the object moves from a dark lit scene to a bright one. In the first part of the video, lighting is very poor and texture-based trackers usually work better, whereas in the second part, the texture changes due to varying pose and colour-based trackers would be more robust. Then, in the third video, the target object is subject to large deformations, a keypoint-based tracker might be more appropriate than a fixed template-based method. Moreover colour features would be more suitable compared to texture-based or motion-based features.

This motivates our choice of combining several independent trackers at a higher level, each of them using features based on different visual aspects like colour, texture and shape. Our intuition is that, in order to cover all the possible variations in scene conditions, combining multiple existing complementary trackers can improve the overall robustness of the tracking performance.

In the following, we will describe some of the most important works of tracker combinations in its various forms, in section 3.2. We will then present our Spacial and Temporal Coherence tracker selection framework (STC) in section 3.3. In this framework, by recurrently selecting the most suitable tracker, the overall system can switch rapidly between appearance models depending on the changes of the scene and the object. Each of the individual trackers alone does not perform very well on average, measured on a challenging tracking benchmark dataset. However, as we will show experimentally, our proposed combination scheme of multiple trackers using different types of features outperforms each of these individual trackers as

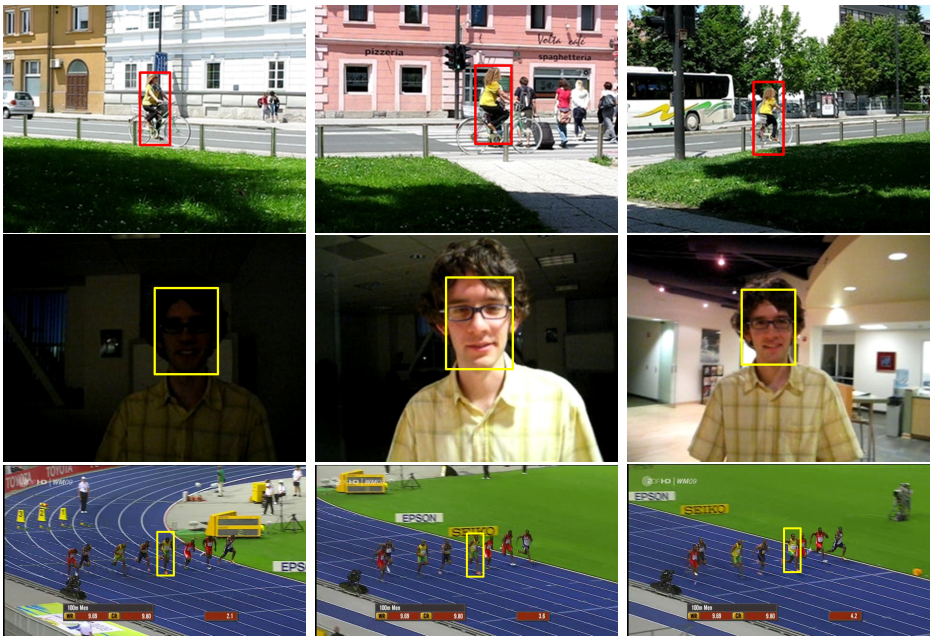


FIGURE 3.1: Illustration of different scene context variations in lighting, texture or background throughout the "bicycle" (1^{st} row), "david" (2^{nd} row) and "bolt" (3^{rd} row) videos from the VOT2013 dataset.

well as a *single* tracker that combines *all* types of features in the classical way.

3.2 State-of-the-art of Tracker Combination

Combining multiple models or trackers into one tracking framework is a common approach employed to increase the tracking robustness in unconstrained environments. Many ways of combining, fusing or selecting visual models or features for tracking exist in the literature.

We can distinguish mainly two groups of combination depending on the level at which the combination is performed: *low-level combination* of visual cues or features, which happens at model level, and *high-level combination* of models of trackers which happens at output level of the tracking framework.

Moreover, the *combination* can either be a type of *fusion* where the different modalities are fused together to produce an output, or a *selection* where one modality or tracker is selected as the best performing one and its result is considered the overall output.

3.2.1 Low-Level Combination

Low-level fusion is the combination of multiple complementary visual features in a single tracking model. It is a common practice in object tracking as the combination of visual cues will be more robust than a single cue in the context of various scene conditions. For instance, colour based features tend to be invariant to shape changes while gradient based features tend to be invariant to illumination changes. Thus, having the combination of both features would give to most robust result in a variety of different situations.

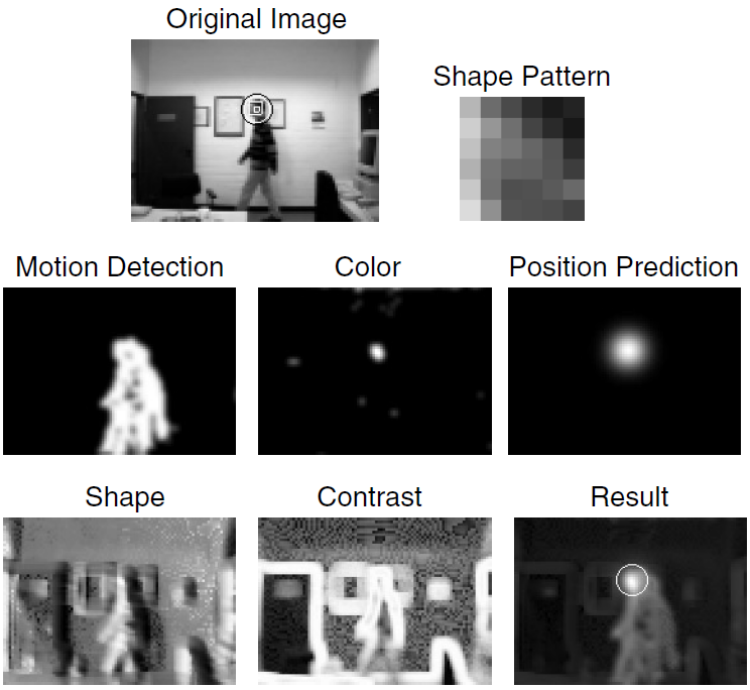


FIGURE 3.2: Overview of the different cues used by Triesch *et al.* [TM01]. (Top row) Original image with circle marking the estimated face position. The squares inside the circle define the image area the information for updating the prototypes of the shape and colour cue is extracted. The shape pattern is the current prototype used by the shape cue. (Centre and bottom rows) Cues' saliency maps and the averaged result. Image courtesy of [TM01].

An early work from Birchfield *et al.* [Bir98] presented a head tracking algorithm using the sum of colour histograms and intensity gradient likelihoods. This work integrated the two visual modules in order to match both the intensity gradients around the object boundary and the colour histogram computed within the object region.

Triesch *et al.* [TM01], on the other hand, introduced a weighting system for the features in order to fuse them in a "democratic" integration. Five different cues are used operating on the basis of saliency maps as represented in figure 3.2: motion detection, colour, position prediction, shape and contrast. The resulting weighted average of the saliency maps derived by the different cues is fed back to the cues and used for adaptation. The weights are, in fact, updated according to the agreement of the cues on the overall result. Moreover, discordant cues are suppressed and recalibrated, while cues having been consistent with the result in the recent past are given a higher weight in the future. This democratic integration framework was applied to the specific task of face tracking.

A Bayesian framework introduced by Yilmaz *et al.* [YLS04] fuses Probabilistic Density Functions (PDF) based on different modalities for object contour tracking. Using texture and colour features, the PDFs are fused in an independent opinion polling strategy, where the contribution of each feature is defined by its discriminative power. The associated energy functional combines region-based and boundary-based object segmentation approaches into one framework. This framework is a mix between the boundary-based

and region-based variational tracking methods.

Other existing low-level fusion works [PVB04; YPC08; SGG09] combine different visual modalities, like motion or shape, in order to improve the overall foreground-background discrimination.

A different approach to low-level fusion is to select different features over time. Collins *et al.* [CL05] for example proposed a tracking framework for evaluating multiple feature spaces and adjusting the set of features used to improve tracking performance. This framework was based on the hypothesis that the features that best discriminate between object and background are also best for tracking the object. The authors proposed to built an online feature ranking mechanism based on the two-class variance ratio measure, applied to log likelihood values computed from empirical distributions of object and background pixels with respect to a given feature. This feature ranking mechanism is embedded in a tracking system that adaptively selects the top-ranked discriminative features for tracking. Although only using RGB colour histograms, this work can extend to other feature spaces.

The low-level fusion of features might lead to problems because of the interdependence of the features in the tracking model. In fact, if some of the visual cues are altered or occluded, for example because of changing scene conditions, the whole model is prone to drift.

3.2.2 High-Level Combination

High-level fusion considers the combination of the outputs of multiple trackers or models. The trackers are usually combined in a parallel way, with each tracker specialising on a different conditions related to the environment, the type of object, appearance, motion, *etc.* In such frameworks, the trackers are treated as "black boxes", that is only the output of the trackers are used. Usually, the tracking outputs used are the estimated object position and/or some sort of confidence measure.

Fusion

A probabilistic combination was proposed by Leichter *et al.* [LLR06] with multiple synchronous trackers operating in different state-spaces. In this framework, each separate tracker outputs sequentially a probability density function of the tracked state at each frame. The trackers may compute either an explicit probability density function, or a sample-set of it via CONDENSATION. This need of a PDF output from the tracker limits the generalisation of the framework to other common trackers that do not provide it.

Stenger *et al.* [SWC09] designed a tracking framework with multiple observation models and addressed the problem of learning which models are the most suitable for tracking at a given time, and they applied this method for the particular tasks of hand and face tracking. Using an offline training set, various off-the-shelf trackers are evaluated as their error distributions are learned and then used to evaluate different combinations of observers, including parallel and cascaded combinations. The framework therefore construct a tracker combination adapted to a specific tracking task in order to ensure its performance. An offline trained detection component is also included in order to initialise the tracker and prevent drifts.

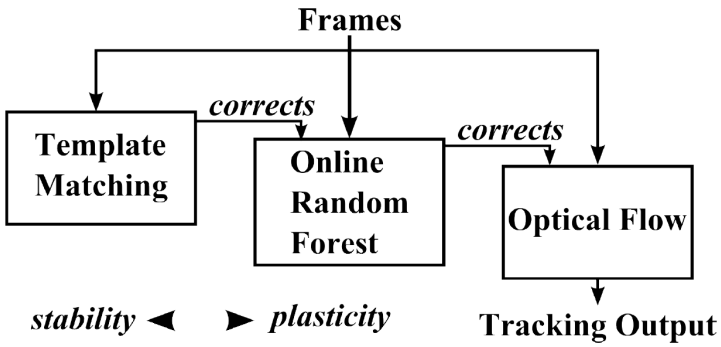


FIGURE 3.3: Overview of the fall-back cascade of the PROST tracker [San+10]. Highly-flexible parts of the system take care of tracking, while the conservative parts correct the flexible ones when they have drifted away. Image courtesy of [San+10].

Instead of the common parallel fusion setting, Santner *et al.* proposed the PROST (Parallel Robust Online Simple Tracking) algorithm [San+10] fusing three trackers in a cascade: a template model is used as a non-adaptive method for stability, an optical-flow based mean-shift tracker as a highly adaptive element, and an online random forest tracker as a moderately adaptive appearance-based learner. These trackers are combined in a simple fall-back cascade where each tracker corrects the previous one as illustrated in figure 3.3. They cover the adaptivity spectrum in the aim of alleviating the drifting problem of appearance based trackers while still being highly adaptive. The optical-flow-based tracker is set as the main tracker, whose output is verified and can be overruled by the random forest tracker while the template model is deployed as a superior safe-guard to monitor and to prevent the random forest tracker from making incorrect updates.

In a different direction, tracker performance within a parallel framework can be measured as the *disagreement* of a tracker with respect to the other trackers. Li *et al.* [Li+12] exploited this idea where instead of assuming a multi-view based framework, they employed existing trackers and fused them in a way that seeks a balance between the coherence of the current tracker and the degree of agreements among other trackers. In such a way, it enables the interaction between trackers and keeps the appealing characteristics of the trackers at the same time.

Inspired by the *test and select* framework [Sha+00] for ensemble combination, the recent work by Khalid *et al.* [KSC16] presented a disagreement-based multi-tracker fusion framework. In this framework, the trackers are grouped hierarchically based on their agreement in estimating the target state in terms of spatial location and direction of movement. Based on a reverse-time analysis performance evaluation, a cluster of trackers is defined, with trackers that appear to be successfully following the target as the on-target cluster. Then, the state estimations produced by trackers in the on-target cluster are fused to obtain the target state.

Using the object trajectory as a fusion criteria is another possibility. For example, Bailer *et al.* [BPS14] used trajectory optimisation to fuse the tracking results (bounding boxes) from different tracking algorithms. This method is based on the notion of *attraction*, and the result that maximises the attraction of all the trackers is chosen as global tracking result. In fact,

instead of using a voting system which requires a threshold in the case of two result boxes voting for the same position, an attraction measure is defined. The closer a fusion candidate is to a tracking result box the stronger it is attracted by it. The sum of attractions for all tracking results is integrated in an energy function to be maximised in order to find the fusion result. This optimisation results in a continuous and smooth overall trajectory.

Wang and Yeung [WY14] also modelled the object trajectory and the reliability of each of five independent trackers combining them with a Factorial Hidden Markov Model (FHMM). Inspired by machine learning methods developed for crowd-sourcing ([Bac+12]), they jointly learn the unknown trajectory of the target and the reliability of each tracker in the ensemble. For efficient online inference of the FHMM, a conditional particle filter algorithm is devised by exploiting the structure of the joint posterior distribution of the hidden variables.

Inspired by Boosting, Penne *et al.* [Pen+13] proposed a machine learning based strategy for ensemble tracking. They built a global observation model of tracking systems as a linear combination of several simplest observation functions so-called modules for each visual cue (RGB values and HOG). Each module is built using a Adaboost-like algorithm. Then, the importance of each module (*i.e.* the weight in the linear combination) is estimated using an probabilistic sequential filtering framework with a joint state model composed by both the spatial object parameters and the importance parameters of the observation modules. Thus, using the several homogeneous feature spaces, the object is tracked using the Adaboost-like algorithm on each of these spaces and results from each space are combined into a unique one, managing their complementarity, reliability and their redundancy.

More recently, Vojir *et al.* [VMN16] proposed an active fusion framework HMMTxD, where an HMM is utilised to fuse observations from complementary trackers and a detector. The HMM's latent states correspond to a binary vector expressing the failure of the individual trackers. The Markov model is trained in an unsupervised way, relying on an online learned detector to provide a source of tracker-independent information as presented in figure 3.4. The HMM estimates the most probable state of the trackers (correct or incorrect operation) and outputs an average bounding box of the correct trackers.

In the aim of addressing the problem of long-term persistent tracking in changing environments, Zhong *et al.* [Zho+14a] presented a passive tracker fusion framework. They consider visual tracking as a weakly supervised learning scenario where (possibly noisy) labels, but no ground truth, are provided by multiple imperfect oracles (*i.e.* trackers). The most likely object position is inferred by considering the outputs of all trackers, and estimating the accuracy of each tracker in a probabilistic manner. Thus, an online evaluation strategy of trackers and a heuristic training data selection scheme are adopted to make the inference more effective and efficient. The general framework is described in figure 3.5.

Selection

In tracker *selection* algorithms, as opposed to tracker *fusion*, only one result is selected and chosen as the most confident from the ensemble of models rather than fusing the information provided by all the trackers. The main

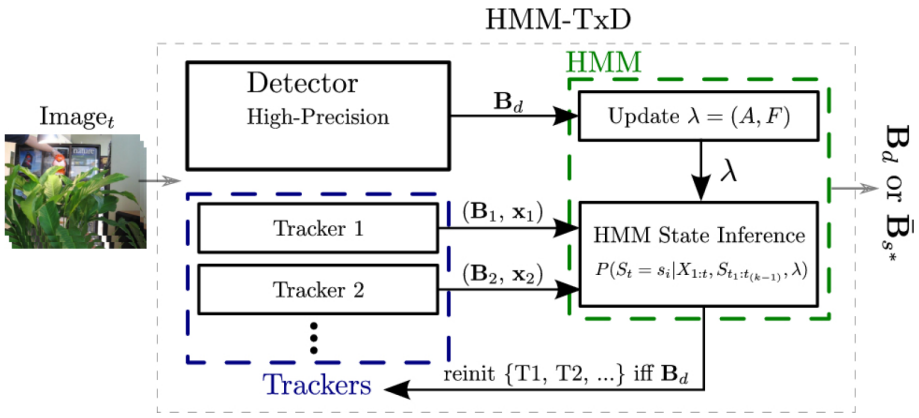


FIGURE 3.4: Overview of the HMMTxD [VMN16] structure. For each frame, the detector and trackers are run. Each tracker outputs a new object pose and observables $(\mathbf{B}_i, x_i)$ and the detector outputs either the verified object pose $\mathbf{B}_d$ or nothing. If the detector fires, the HMM is updated and trackers are reinitialised, and the final output is $\mathbf{B}_d$, otherwise, the HMM estimate the most probable state s^* and outputs an average bounding box $\bar{\mathbf{B}}_{s^*}$ of trackers that are correct in the estimated state s^*. Image courtesy of [VMN16].

advantage of selection algorithms is that the resulting tracking output is not altered by one or more trackers that may have drifted as long as they are not selected.

A probabilistic sampling framework, named Visual Tracker Sampler (VTS), was introduced by Kwon *et al.* [KL10; KL11] integrating estimates from basic complementary trackers using different observation models and motion models. Several samples are gathered from the different states of the target and the trackers during the sampling process. The trackers are sampled using the Markov Chain Monte Carlo (MCMC) method from the predefined four-dimensional tracker space in which the axes are the appearance model, motion model, state representation type, and observation type. Then, the sampled trackers run in parallel and interact with each other as illustrated in figure 3.6. The accepted sample giving the highest value on the Conditioned Maximum A Posteriori (CMAP) estimate is chosen, thus giving a highly possible tracker (the best tracker that tracks the target robustly with high probability) and a highly possible state (the best state where the target might be).

The problem with approaches that use confidence maps and also methods based on tracker sampling such as VTS is that the likelihood needs to be computed on many image positions (or at least a search window) and possibly at different scales, which is computationally expensive.

In a different manner, rather than using multiple trackers, Zhang *et al.* [ZMS14] designed a restoration scheme where snapshots of a base tracker are retained in time, constituting an ensemble of its past models. In addressing the common model drift problem in visual tracking, the proposed formulation aims to correct the effects of bad updates of the tracking model after they happen, instead of trying to prevent the bad updates from happening. Using a discriminant online SVM base tracker, a set of the base tracker snapshots is maintained throughout the tracking process and constitute an expert ensemble. The best expert is then selected based on a minimum

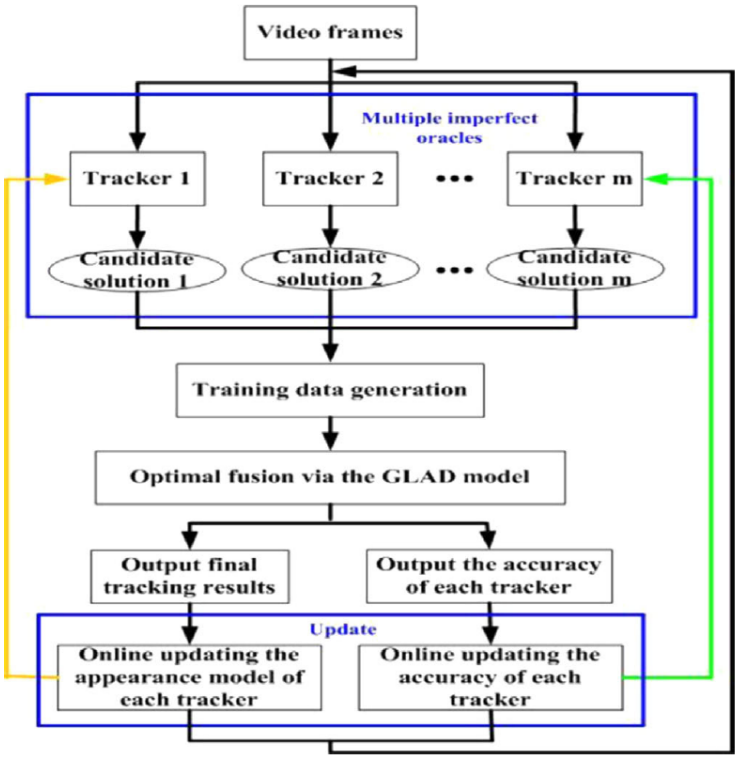


FIGURE 3.5: Overview of the fusion method proposed by Zhong *et al.* [Zho+14a]. For each video frame, a set of candidate solutions are first estimated by the set of tracking methods. Then, the training data is adaptively selected according to a heuristic strategy and used by the GLAD model [Whi+09] to simultaneously infer the most likely object position and the accuracy of each tracker. A testing sample with the maximum probability of belonging to the positive sample set is chosen to be the new object position, and is also retained as a positive training sample for the tracker update in the following step. Finally, both the accuracy of each tracker and the their appearance models are updates. Image courtesy of [Zho+14a].

loss criterion to restore the current tracker when a disagreement among the experts occurs.

3.3 Spatial and Temporal Coherence based Tracker Selection

Most of the published tracker combination approaches rely on high level fusion of state-of-the-art trackers. However, we start from the assumption that different trackers do not perform well all the time, nor at the same time. Fusing all individual tracking results is therefore not optimal, we rather chose to *select* the most suitable tracker from a pool of complementary methods all running in parallel, and retain only the result of this selected tracker for each video frame.

We propose the Spatial and Temporal Coherence-based tracker selection framework (STC) using multiple trackers based on complementary features. The selection is performed iteratively every video frame by first applying a spatial and temporal coherence criteria to the trackers' results, and then by

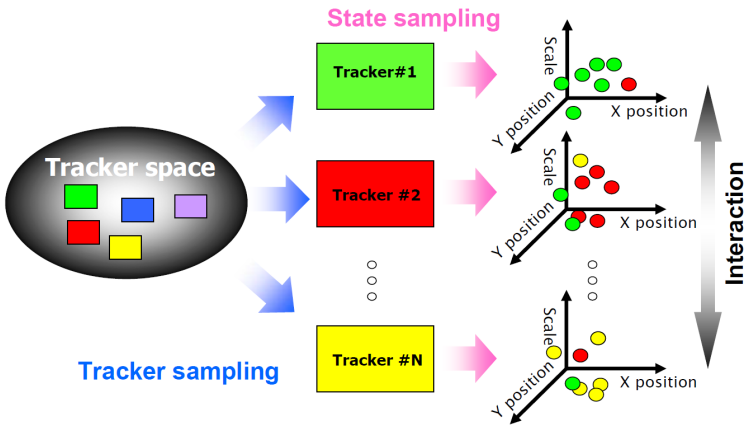


FIGURE 3.6: Overview of the VTS method by Kwon *et al.* [KL11]. The "sampler" constructs the trackers by sampling them from the tracker space, runs the sampled trackers in parallel and interactively, and obtains samples of the target state utilising the trackers. Image courtesy of [KL11].

observing the confidence of each tracker that is normalised in a principled way using the parameters learnt offline from a separate training set.

The proposed framework is general and, in principle, any kind of tracker can be used as long as it provides a confidence measure, a score, or a probability along with the estimated object position. Thus, we do not make use of likelihood or confidence maps as in previously proposed combination frameworks because they are quite specific to only a few existing methods. Here, we choose trackers relying on the Online AdaBoost (OAB) algorithm [GB06] enhanced with different types of features.

3.3.1 Principle Approach

The general framework of the proposed tracker selection algorithm is demonstrated in figure 3.7. We employ multiple trackers $\{T_1, .., T_L\}$ in a parallel setting, each of these trackers $T_k, k \in (1..L)$ provides an estimation of the object's position, *i.e.* a bounding box B_t^k, and a confidence measure of this result c_t^k, for every video frame t.

The first step is the normalisation of confidence values $\{c_t^1, .., c_t^L\}$. They are normalised based on parameters that have been trained before on a separate training data set. This normalisation is an important step as the different trackers can be heterogeneous and their confidence values may have different dynamics. During tracking, for the current frame, a process of tracker elimination is applied based on the overall temporal and spatial coherence of the trackers' results. This process filters out the trackers outside a given limit, *i.e.* the out-liners, resulting in a smoother and spatially more coherent tracking result. In the final step, the best tracker T_t^s for the current frame is selected from the remaining individual trackers based on their normalised confidence values and its estimated object position B_t^s represents the final tracking output for the current frame. All the trackers keep their individual states (*i.e.* bounding boxes) and are re-considered for selection at each point in time.

The advantage of this tracker selection approach is that it avoids unnatural jumps of the resulting bounding boxes while still being able to quickly

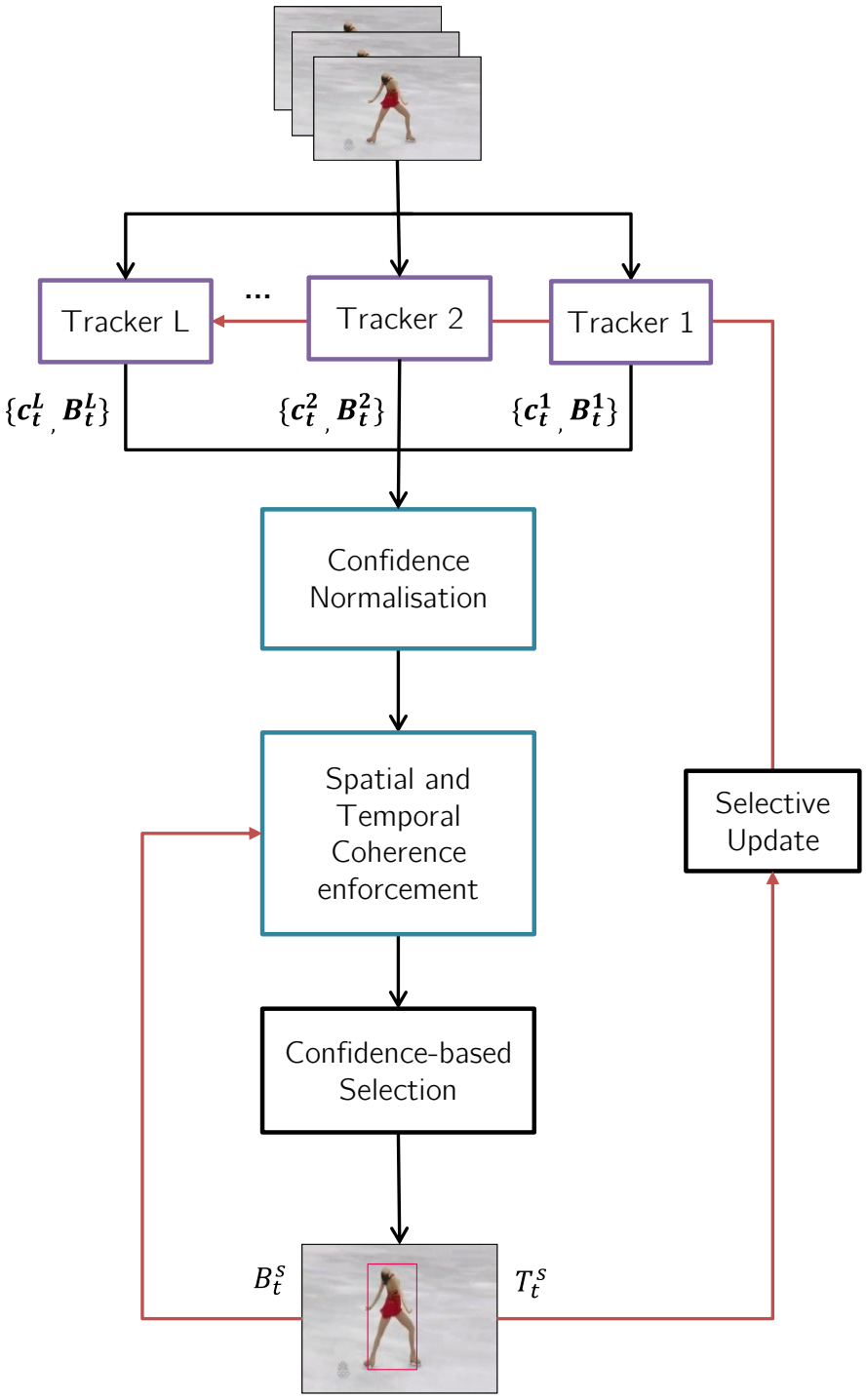


FIGURE 3.7: Overall procedure of the proposed Spatial and Temporal Coherence (STC) tracker selection framework.

change from a less confident tracker to more confident one, for example when the object's appearance rapidly changes.

An additional selective update is also performed, where not all the trackers are updated with their results $B_t^k, k \in (1..L)$, but only the *selected* tracker for the current frame is updated with its resulting bounding box. This helps to avoid the eventual drift of the trackers.

Also there is no direct interaction between the different trackers. Each tracker remains independent, and only one tracker is selected based on the individual confidence values and the global spatio-temporal coherence. This has the advantage that a single tracker that has drifted or even lost the object does not impair the trackers that perform better if it gets selected mistakenly.

3.3.2 Individual Trackers: Online AdaBoost

The concept for this framework is to use relatively simple, fast and discriminative trackers, that are based on low-level independent and complementary features. Here, we propose to use different *Online AdaBoost* trackers, each of them using a different type of visual feature which will be described. The choice of using AdaBoost-based trackers is motivated by their computational efficiency and the simplicity of using different independent features with the same architecture. Also, it easily allows a baseline comparison with a single AdaBoost tracking performing a lower-level fusion of the same visual features. Of course, a different tracking method (*e.g.* structured output SVMs) can be used as well as the proposed tracker selection framework is general and only requires trackers with bounding box outputs and corresponding confidences measures.

Online AdaBoost (OAB) [GB06] is an extension of the well-known AdaBoost (Adaptive Boosting) classification technique proposed by Freund and Schapire [FS97] for on-line learning. It has been effectively applied to the on-line tracking problem by Grabner *et al.* [GGB06] by training a binary classifier with foreground and background image patches.

The basic idea of AdaBoost classification is to combine several "weak" classifiers into a single "strong" classifier, where the weak classifiers perform only slightly better than just random guessing. Given an example image patch $\mathbf{x}$, we define the following:

Weak classifier: A weak classifier $h^{weak}(\mathbf{x}) \in \{-1, +1\}$ performs a simple binary classification using a single low-level feature extracted from the candidate image region $\mathbf{x}$.

Selector: Given a global pool of M weak classifiers $\{h_1^{weak}, .., h_M^{weak}\}$, a selector h^{sel} iteratively selects one of them such as the weak classifier error e_m is minimised:

$$\begin{aligned} h^{sel}(\mathbf{x}) &= h_m^{weak}(\mathbf{x}), \\ m &= \underset{m}{\operatorname{argmin}}\, e_m. \end{aligned} \tag{3.1}$$

Strong classifier: Given the set of N selectors $\{h_1^{sel}, .., h_N^{sel}\}$, the strong

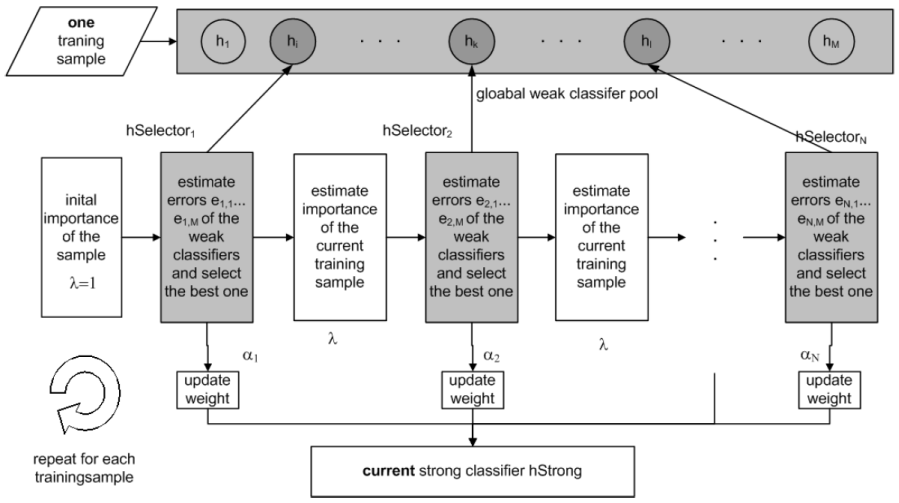


FIGURE 3.8: Principle of Online AdaBoost for feature selection. Image courtesy of [GGB06].

classifier H^{strong} is computed by a weighted linear combination of the selectors:

$$H^{strong}(\mathbf{x}) = sign\left(\sum_{n=1}^{N} \alpha_n \, h_n^{sel}(\mathbf{x})\right), \tag{3.2}$$

where the $\alpha_n \in (0,1)$ are the weak classifier weights:

$$\alpha_n = \begin{cases} 0 & \text{if } e_n < 0.5 \\ \ln\left(\dfrac{1-e_n}{e_n}\right) & \text{otherwise.} \end{cases} \tag{3.3}$$

The confidence of the strong classifier in its result, using the input image patch $\mathbf{x}$, is thus defined by the value:

$$c(\mathbf{x}) = \sum_{n=1}^{N} \alpha_n \, h_n^{sel}(\mathbf{x}). \tag{3.4}$$

In this framework, the weak classifiers correspond to relatively simple visual features as those explained in the following section, *i.e.* the hypotheses generated by a weak classifier is based on a discriminative low-level characteristic computed on the input image patch x. Thus, the selector can be interpreted as a classifier, switching between the weak classifiers. Training a selector means that each weak classifier is trained (updated) and the best one, *i.e.* with the lowest estimated error, is selected.

The overall principle of Online AdaBoost algorithm for feature selection is presented in figure 3.8, and detailed in algorithm 1. First, a fixed set of N selectors $\{h_1^{sel}, .., h_N^{sel}\}$ is initialised randomly, each with its own feature pool $\{h_{n,1}^{weak}, .., h_{m,N}^{weak}\}$, from the global feature pool. Given a new training sample $\langle \mathbf{x}, y \rangle$, $\mathbf{x}$ being the image patch and y the corresponding label (*i.e.* object or background), the selectors are updated with respect to the importance weight of the current sample λ. The weak classifiers are updated with a standard EM technique to estimate the probability distributions of positive and negative samples and generate a hypothesis. Then, the weak classifier with the smallest error is selected by the selector, where $e_{n,m}$ is the error

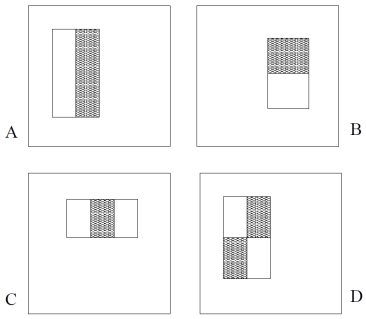


FIGURE 3.9: Examples of used Haar-like features, relative to the enclosing detection window. Two-rectangle features are shown in (A) and (B). Figure (C) shows a three-rectangle feature, and (D) a four-rectangle feature. Image courtesy of [VJ01].

of the m^{th} weak classifier $h^{weak}_{n,m}$ in the in the n^{th} selector, estimated from the weights of correctly $\lambda^{corr}_{n,m}$ and wrongly $\lambda^{wrong}_{n,m}$ classified examples seen so far. Finally, the corresponding voting weight α_n and the importance weight λ of the sample are updated and passed to the next selector h^{sel}_{n+1}. Moreover, to increase the diversity of the classifier pool and allow changes in the environment, the worst weak classifier is replaced by one randomly chosen from the feature pool. The online update is repeated for all selectors, and finally the updated strong classifier computed by the linear combination of selectors provides its result for the input image patch.

To perform visual object tracking, this procedure is applied at each frame by applying the classifier several times on image patches in a region around the last tracking position (ROI) and choosing the patch $\mathbf{x}^*$ with the highest confidence.

3.3.3 Low-Level Features

Based on the same OAB tracking architecture, we propose to use three types of complementary features to implement three trackers in the proposed tracker selection framework. Note that the original OAB method proposed by Grabner *et al.* [GGB06] only used Haar-like features. We extended this in order to perform a systematic analysis on different fusion strategies using the same individual base tracking algorithm. As mentioned before, each individual tracker relies on different types of features. Clearly, the proposed method is not limited to these three, and more trackers and/or other feature types can be added. For all the features types, a feature is computed on a sub-region of the image patch that corresponds to the object's bounding box.

We thus implemented three OAB trackers based on the following features, using 200 weak classifiers and 50 selectors for each tracker.

Haar-like Features

The Haar-like features used by Grabner *et al.* in the OAB tracker [GGB06] were originally proposed by Papageorgiou *et al.* [POP98] and efficiently implemented by Viola and Jones [VJ01] in their face detection algorithm. These features correspond to differences of the mean pixel intensities of adjacent rectangular regions. Examples of the Haar-like features used are presented in figure 3.9. The value of a two-rectangle feature is the difference between the

Algorithm 1 Online AdaBoost for feature selection

Require: training sample $\langle \mathbf{x}, y \rangle, y \in \{-1, +1\}$
Require: strong classifier H initialised randomly
Require: weights $\lambda^{corr}_{n,m}$,$\lambda^{wrong}_{n,m} = 1$

initialise importance weight $\lambda = 1$
//for all the selectors
for $n = 1, 2, .., N$ **do**

//update selector h^{sel}_n
for $m = 1, 2, .., M$ **do**

//update each weak classifier
$h^{weak}_{n,m} = update(h^{weak}_{n,m}, \langle \mathbf{x}, y \rangle, \lambda)$

//estimate errors
if $h^{weak}_{n,m}(\mathbf{x}) = y$ **then**
$\lambda^{corr}_{n,m} = \lambda^{corr}_{n,m} + \lambda$
else
$\lambda^{wrong}_{n,m} = \lambda^{wrong}_{n,m} + \lambda$
end if
$e_{n,m} = \frac{\lambda^{wrong}_m}{\lambda^{corr}_m + \lambda^{wrong}_m}$

end for

// choose weak classifier with the lowest error
$m^+ = \text{argmin}_m(e_{n,m})$
$e_n = e_{n,m^+}$; $h^{sel}_n = h^{weak}_{n,m^+}$
if $e_n = 0$ or $e_n > 0.5$ **then**
exit
end if

// calculate voting weight
$\alpha_n = \frac{1}{2} \cdot \ln(\frac{1-e_n}{e_n})$

// update importance weight
if $h^{sel}_n(\mathbf{x}) = y$ **then**
$\lambda = \lambda \cdot \frac{1}{2 \cdot (1-e_n)}$
else
$\lambda = \lambda \cdot \frac{1}{2 \cdot e_n}$
end if

// replace worst weak classifier with a new one
$m^- = \text{argmax}(e_{n,m})$
$\lambda^{wrong}_{n,m^-} = 1$; $\lambda^{corr}_{n,m^-} = 1$;
get new h^{weak}_{n,m^-}

end for

sum of the pixels within two rectangular regions. The regions have the same size and shape and are horizontally or vertically adjacent. A three-rectangle feature computes the sum within two outside rectangles subtracted from the sum in a centre rectangle. Finally a four-rectangle feature computes the difference between diagonal pairs of rectangles.

In order to compute the sums of pixels over the different patches efficiently, the image integral is employed.

HOG Features

We implemented a simplified version of the Histogram of Oriented Gradient (HOG) features [DT05]. The gradients are computed on a region and then quantised into a set of 8 orientations and 3 magnitudes, The value of a single bin, *i.e.* combination (orientation, magnitude), is used as a feature. The resulting feature is the number of times a random combination of bin and magnitude appears in the cell. To compute the features over the ROI efficiently, the image integral of the HOG features, as well as the HOC features is employed.

HOC Features

The Histogram Of Colour (HOC) features are computed the same way as the HOG features. First the RGB input image is converted into HSV channels, then the Hue and Saturation values over the cell are quantised into 3 saturations and 8 hues. Then a histogram is computed for every possible combinations, and the value of a feature correspond to the occurrence of the specific (hue, saturation) combination over the cell.

The feature selection (sub-regions, bins) is performed by the standard AdaBoost algorithm on a subset of random samples.

3.3.4 Confidence Measure and Normalisation

A variety of confidence measures exist depending on the tracking or classification algorithm. Here, we use the one originally proposed for OAB. Given a set of trackers $T_k, (k \in 1..L)$, each tracker T_k produces an object position B_t^k and a confidence value c_t^k, at a frame t. The OAB trackers' confidence is obtained by weighting the results of each one of its selectors $\{h_1^{sel}, .., h_N^{sel}\}$ on the chosen patch $\mathbf{x}^*$.
Thus, for each tracker and at each frame, the confidence is defined as:

$$c(\mathbf{x}^*) = \sum_{n=1}^{N} \alpha_n \, h_n^{sel}(\mathbf{x}^*) \, . \tag{3.5}$$

This confidence measure expresses, in a way, the proportion of selectors h_n^{sel} having correctly classified example $\mathbf{x}^*$. Note that it is directly related to the output of the strong classifier (*c.f.* Equation 3.2).

Once we have the set of confidences $\{c_t^1, .., c_t^L\}$, we need to be able to compare them. The confidence values are important in the selection procedure in the framework, so a balance comparison is essential. And having different trackers based on features that perform differently, the confidence values for each one will have different dynamics. In order to be able to compare the confidence values, we normalise them for each tracker T_k.

Similarly to [SWC09], we compute the mean μ_k and standard deviation σ_k of the confidence values of each tracker c_k over a separate training data set. We chose the ALOV++ dataset proposed by Smeulders *et al.* [Sme+14]. For this purpose, ALOV++ videos cover various changes in illuminations, transparency, clutter, occlusion, zoom, appearance, motion patterns and contrast. Moreover, it is a large dataset with 315 videos, which makes it suitable for capturing the dynamics of confidence values of the individual trackers and thus estimating μ_k and σ_k. See section 2.5.1 for more details on the dataset.

After computing μ_k and σ_k in an offline step, we can normalise our confidence values, for each frame t and each tracker T_k:

$$\bar{c}_t^k = \frac{c_t^k - \mu_k}{\sigma_k}, \tag{3.6}$$

and we use these normalised values for the following processing steps.

3.3.5 Spatial and Temporal Coherence

Using the normalised confidence to choose the best tracker at a given frame t is not sufficient, as we will see in the experiments section. When the bounding boxes of the individual trackers are further apart (because one or several of them have drifted away), jumps can occur in the overall tracking process since no continuity is present in the choice of the best confidence.

In order to avoid these jumps and to make the output smoother, a spatial and temporal coherence criterion is introduced before the selection step, which ignores the result of those trackers that are too far away from the previous object's position. At each frame t, we have a set of L trackers $\{T_1, .., T_L\}$ that return normalised confidence values $\{\bar{c}_t^1, .., \bar{c}_t^L\}$ and bounding boxes $\{B_t^1, .., B_t^L\}$ surrounding the object in the image. For each tracker $T_k, k \in (1..L)$, we define the centre position of the bounding box B_t^k as (x_t^k, y_t^k) and its dimensions as (w_t^k, h_t^k). The previous position of the target object, *i.e.* the bounding box B_{t-1}^s, its centre (x_{t-1}^s, y_{t-1}^s) and dimensions (w_{t-1}^s, h_{t-1}^s) from the *selected* tracker in the previous frame, is used to compute its distance to each of the current tracker's bounding boxes B_t^k. Then, at frame t, tracker T_k is (temporarily) eliminated if:

$$max\left(\left|x_t^k - x_{t-1}^s\right| - \Theta_x, \left|y_t^k - y_{t-1}^s\right| - \Theta_y\right) > 0\,. \tag{3.7}$$

The threshold $\Theta = (\Theta_x, \Theta_y)$ is a two-dimensional distance threshold proportional to the size of B_{t-1}^s:

$$\begin{aligned} \Theta_x &= \beta\,\frac{w_{t-1}^s}{2} \\ \Theta_y &= \beta\,\frac{h_{t-1}^s}{2}. \end{aligned} \tag{3.8}$$

The optimal coefficient β is computed empirically, as we will see in the experiments section, by evaluating the algorithm on a separate dataset, and is left constant for all of our experiments. The main goal of this step is to eliminate, for the current frame, the trackers considered drifting.

After this step, only a subset of trackers $\{T_i\}, (i \in 1..L')$, with $L' \leq L$ is left. The final decision on the best tracker T_t^s with bounding box B_t^s is

given by selecting the tracker with maximal normalised confidence $\bar{c}_t^s$:

$$s = \underset{i \in 1..L'}{\operatorname{argmax}} \left(\bar{c}_t^i \right). \tag{3.9}$$

In the case where $\{T_i\}, (i \in 1..L')$ is empty, then only the maximal confidence criterion is applied.

3.3.6 Selective Update

In the original OAB architecture, a tracker T_k updates its discriminative model using the resulting bounding boxes B_t^k at a frame t. One update scheme in the selection framework would be to keep the trackers completely independent in the sense that each tracker $T_k, (k \in 1..L)$ is update with its result separately. Another update scheme would be to use the *selected* tracker's bounding box to update all the trackers. This idea proved to be ineffective due to the rapid drift of the now co-dependent trackers. In fact, the trackers start drifting when the background information is falsely included into the foreground model.

Instead, we introduce a selective update strategy where only update the *selected* tracker T_t^s with its resulting bounding box B_t^s at each frame t. This update strategy has the advantage of keeping the trackers independent. Since we consider that the non-selected trackers may have *eventually* drifted, not updating these trackers lowers the risks of drifts and the framework gains robustness. Moreover, this strategy lowers the update rate of the framework, giving us a slower learning rate of the models and a reduced computational time, which also will help prevent this drifting phenomenon. Thus we do $L - 1$ less updates than in the original learning architecture.

3.4 Performance Evaluation

The experiments for the proposed approach, as well as for the individual trackers and baseline combination methods, were conducted on the VOT2013 dataset (detailed in section 2.5.3). We did not use the provided VOT2013 evaluation framework because the capability of our proposed fusion method to return to the object after losing it is not taken into account in the VOT2013 framework but, on the contrary, is penalised because trackers are stopped and reinitialised (with the ground truth) from the point of loss. Further, the aim here is not to show that the proposed method outperforms existing state-of-the-art tracking algorithms. We rather want to evaluate and show the benefit of our tracker selection approach compared to classical tracker fusion methods.

3.4.1 Evaluation Measures

For the different experiments presented in this chapter, we used multiple evaluation measures that we present in the following.

F-score

The F-score, also called f-measure, is a common tracking evaluation measure which combines the measures of precision and recall and estimates the overlap between a resulting bounding box B_t^k and the ground truth bounding box B_t^{gr}. The $Fscore_t^k$ at the frame t for a tracker T_k is defined as:

$$Fscore_t^k = 2 \times \frac{precision_t^k \times recall_t^k}{precision_t^k + recall_t^k} \tag{3.10}$$

$$= 2 \times \frac{B_t^k \bigcap B_t^{gr}}{B_t^k \bigcup B_t^{gr}}, \tag{3.11}$$

with the precision and recall:

$$precision_t^k = \frac{B_t^k \bigcap B_t^{gr}}{B_t^k} \tag{3.12}$$

$$recall_t^k = \frac{B_t^k \bigcap B_t^{gr}}{B_t^{gr}}. \tag{3.13}$$

For each video, the trackers are initialised with the ground truth in the first frame and run until the end of the video. At each frame, we compute the F-score of the resulting bounding box of a given tracking method. The F-score value is in the interval [0..1], and the method is considered lost the corresponding F-score $Fscore_t^k = 0$.

Success and Prediction Rates

Using the F-score, two performance measures are introduced: *success rate* and *prediction rate*. The *success rate* represents the number of frames having $Fscore_t^k > 0.1$ for a tracker T_k over the evaluation dataset. The relatively low threshold captures the overall tracking robustness of the method, not so much its bounding box accuracy.

The *prediction rate* on the other hand represents the proportion of frames where the tracker combination method correctly predicted the best tracker. In fact, knowing the ground truth position of the object, the F-score of each individual tracker is computed, then the best theoretical tracker (and F-score) at each frame in known. Based on this, we can compute the number of frames where a selection method has correctly predicted the best individual tracker. We allow for a 20% margin on the best F-score, *i.e.* the two best trackers are considered equivalent if the difference of their F-score is lower than 20%.

3.4.2 Individual Trackers Performance

In order to assess the performance of our $L = 3$ individual trackers {OAB Haar, OAB HOG, OAB HOC}, we measured the F-score of each tracker on the VOT2013 dataset at each frame.

We start by observing the evolution of F-score values for each tracker throughout different videos to better understand the behaviour of the different trackers. On the "cup" video, figure 3.10, we can see that the HOG-based tracker performs best in terms of robustness and accuracy, while the

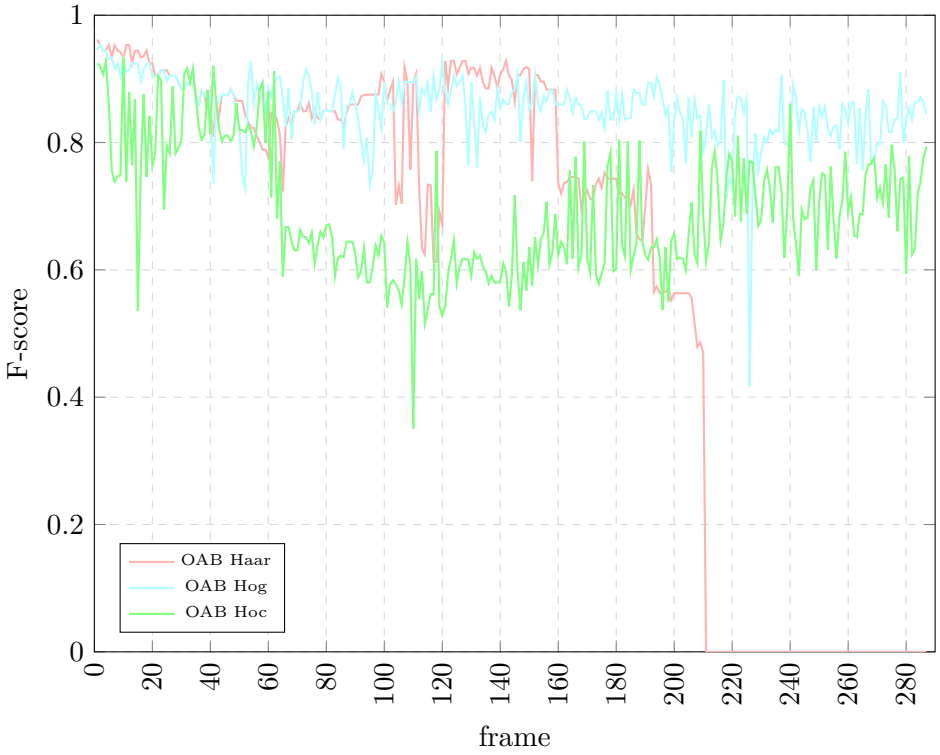


FIGURE 3.10: Evolution of F-score values for the OAB Haar, OAB HOG and OAB HOC trackers over the "cup" video from the VOT2013 dataset.

Haar-based one loses track around frame 210, corresponding to a change of background in the video. On the other hand, on the "face" video, figure 3.11, the OAB Haar tracker is the best performing one, whereas the OAB HOG and HOC trackers drift from the object multiple times. In fact, Haar-like features are more discriminant with faces. As can be observed, the performance of each tracker depends on the environment and nature of the object and selecting the right tracker at each point in time can help the overall tracking performance.

To further understand the behaviour of the trackers, we ran the three trackers {OAB Haar, OAB HOG, OAB HOC} over the entire VOT2013 dataset, collected the F-scores at each frame, and computed a histogram of the F-score values. Since the values are comprised in the interval $[0, 1]$, we quantise them over the bins $[0]$, $]0; 0.1]$, $]0.1; 0.2]$, ..., $]0.9; 1]$. When $Fscore = 0$, *i.e.* the tracker is lost, we count these values separately. The results for the trackers are presented in figure 3.12. Because of the non-deterministic aspect of the OAB, the experiment was repeated 3 times and averaged.

We can first observe that the tracker OAB HOC is the most robust, on average over the dataset, among our pool of trackers, having the least number of failures. Colour proves to be a very discriminative cue throughout the dataset. However, the OAB HOC tracker is not sufficient on its own. The three trackers have a high number of failures, considering the total number of frames.

Considering our goal of combining the trackers, a more important question would be to know if the cases where the trackers fail actually overlap,

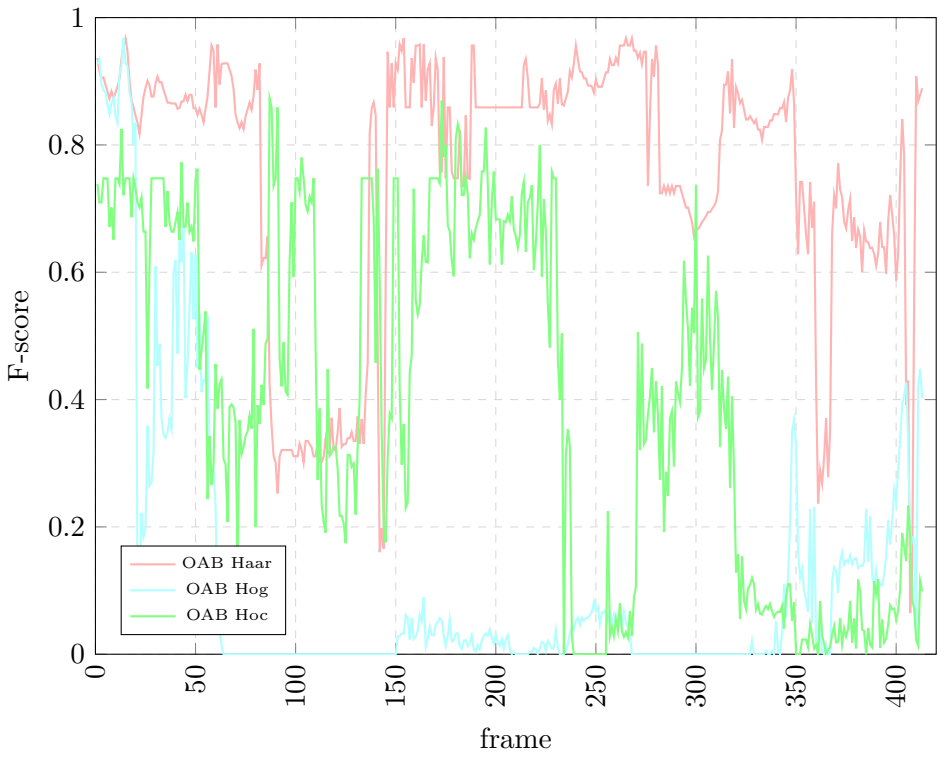


FIGURE 3.11: Evolution of F-score values for the OAB Haar, OAB HOG and OAB HOC trackers over the "face" video from the VOT2013 dataset.

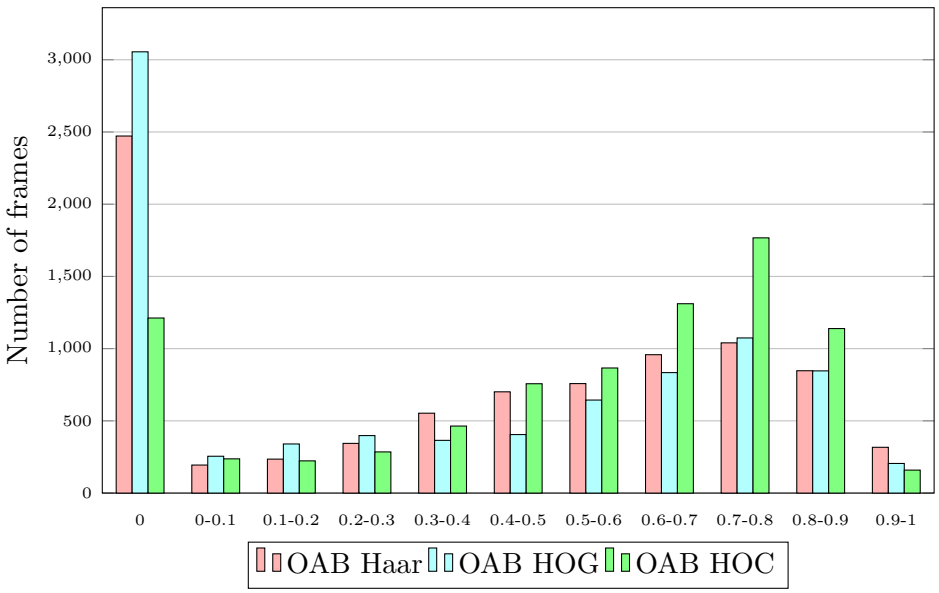


FIGURE 3.12: Performance of OAB Haar, HOG and HOC trackers. Histogram of quantified F-scores over the VOT2013 dataset.

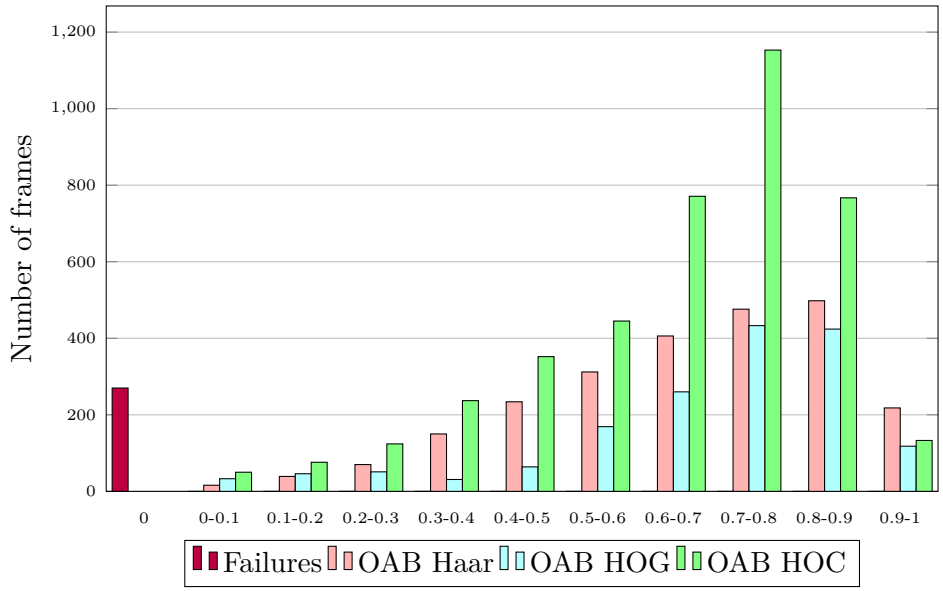


FIGURE 3.13: Best theoretical selection for the OAB Haar, HOG and HOC trackers. Histogram of quantified f-scores over the VOT2013 dataset.

i.e. if the different trackers tend to fail at the same time or not. As we combine the trackers we want the failures of the trackers to be compensated by at least one other tracker. To this end, we compute the tracking results of the "*best theoretical combination*". This hypothetical tracking combination assumes that the ground truth and F-scores of the trackers are known to find the "best theoretical" tracker. That is, at each frame, the tracker with the highest F-score: $Fscore_t^s$, where $s = \text{argmax}_{k \in 1..L}(Fscore_t^k)$, is considered the selected one for the frame.

In the same way as before, we build a histogram of quantised F-scores of the *best theoretical combination* counting the number of frames each tracker was selected. The experiment is also repeated 3 times and averaged. First, we can see in figure 3.13 that the total number of failures (*i.e.* when all the trackers are lost at the same time) is greatly reduced. This leads to a more robust tracking throughout the videos, in comparison each single tracker performance. Moreover, the F-score values are globally increased. The proportion of F-scores with high values $[0.5; 1]$ is more important in comparison to the individual F-scores of each tracker, leading to a more accurate tracking performance. This *best theoretical combination* represents the maximal performance a tracker combination can achieve using our trackers: OAB Haar, OAB HOG and OAB HOC.

3.4.3 Compared Methods

To be able to correctly evaluate the performance of the proposed approach and to comprehend the impact of the different components (*i.e.* the confidence, the spatial and temporal coherence and selective update), we compared it to a certain number of baseline methods. These methods are different combination or selection algorithms using the same individual trackers: OAB Haar, OAB HOG and OAB HOC. In the following, we describe the different methods that have been tested.

- **Proposed method (PM):** As described above, and without selective update, the proposed approach uses $L = 3$ trackers denoted HAAR, HOG and HOC with 50 selectors each. The spatio-temporal coherence and normalised confidence is used for tracker selection. All the trackers are updated independently every frame.

- **Proposed method with selective update (PM+):** This method is the same as PM, but with the selective update strategy described in Section 3.3.6. This method corresponds the selection framework STC.

- **Best confidence (BC):** The selection of the best tracker is only based on the normalised confidence values, that is the tracker T_t^s with the best normalised confidence $s = \text{argmax}_{k \in 1..L}(\bar{c}_t^k)$ is selected for the current frame.

- **Fusion of features (FoF):** The OAB method allows different sets of low-level features to be used together in the global pool of weak classifiers. In the FoF baseline, one strong classifier is constructed based on all the different feature types. This type of feature fusion is a classical way of combining different types of low-level observations. In order to provide for a fair comparison, we used the same number of features as in the proposed approach, *i.e.* 3×50 selectors.

- **Fusion of features with minimal update (FoF+):** To be able to correctly compare to the proposed selection framework with selective update, we decreased the number of updates of FoF by $L - 1$. To this end, the tracker is only updated one frame out of L.

- **Centroid of Trackers (CoT):** This is a high-level fusion approach, where the resulting bounding box is computed from the mean of the bounding boxes of the individual trackers weighted by their respective confidence values.

3.4.4 Parameter Optimisation

In our proposed tracker selection framework, the parameter β (*c.f.* equation 3.8) in the spatial and temporal coherence step needs to be optimised, as discussed in section 3.3.5. This parameter is crucial as it defines the spatial limit around the current position beyond which a tracker can be eliminated from the selection process. In figure 3.14, the success rates of the proposed selection method (PM) are presented for various values of β. Small values of β show limited success rates as they do not allow enough object movement. In that case, the distance threshold of the object's displacement from one frame to the next is too narrow. On the other hand, too large values of β ane not effective enough in eliminating lost or drifting trackers. The optimal value of $\beta = 0.8$ is chosen and left constant for the remaining experiments.

3.4.5 Experimental Results

The proposed selection framework, and the different presented baselines, are compared based on their success and prediction rates over the VOT2013 dataset. Moreover, to provide an upper bound, we also computed the success rate of a best theoretical selection, *i.e.* the success rate of a method

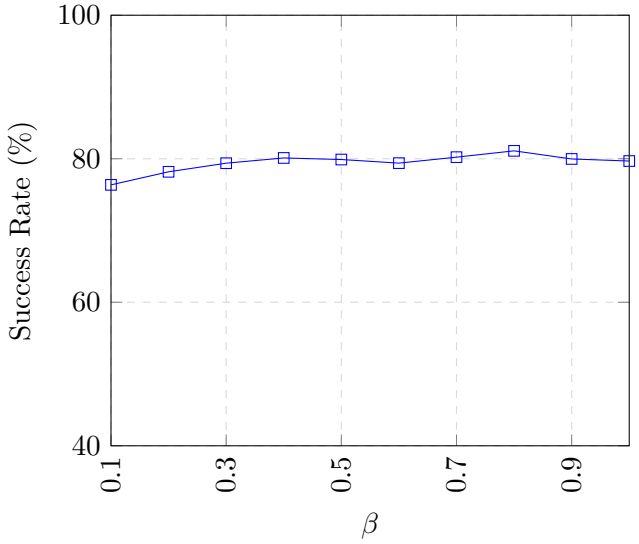


FIGURE 3.14: Optimisation of β parameter. Success rate measures of the proposed method (PM) over the VOT2013 dataset.

Method	Success Rate
Best theoretical selection	*(93.43%)*
Best Confidence (BC)	78.32%
Fusion of Features (FoF)	80.13%
Centroid of Trackers (CoT)	61.21%
Proposed Method (PM)	**81.10%**
Best theor. sel. + selective update	*(93.86%)*
Fusion of Features + min. update (FoF+)	77.45%
Proposed Method + selec. update (PM+)	**83.93%**

TABLE 3.1: Success rates for the different methods evaluated on the VOT2013 dataset.

that always predicts the tracker with the highest F-score. This measure is important as it expresses the general performance of the individual trackers. All presented numbers are averaged measures over 10 tracker runs over the evaluation dataset.

As shown in table 3.1 and table 3.2, the success and prediction rate of our proposed method (PM) outperforms Best Confidence (BC). In figures 3.15, 3.16, 3.17, 3.18 and figure 3.19, we also introduce some results from the video 'bolt'. It is a particularly challenging sequence due to the number of similar objects and colours, complex background, and change in appearance. We can see in figure 3.19 (1^{st} row) that the use of our spatio-temporal coherence criterion eliminates the jumps that are introduced when only using the normalised confidence BC. Although surpassing BC, PM cannot avoid all the jumps. However, thanks to the simultaneous use of the spatio-temporal coherence and the confidence, the proposed method has the ability to "re-lock" on the object without any re-initialisation of the trackers. This is illustrated in figure 3.15, where we can observe that the OAB HOC tracker (green) is the only one able to correctly track the object. BC (figure 3.16)

Method	Prediction Rate
Best Confidence (BC)	76.22%
Proposed Method (PM)	**77.99%**

TABLE 3.2: Prediction rates for Best Confidence and the proposed method evaluated on the VOT2013 dataset.

FIGURE 3.15: Comparison of tracking results on the "bolt" video for (a) the individual trackers (*white*: ground-truth, *pink*: Haar tracker, *blue*: HOG tracker, *green*: HOC Tracker).

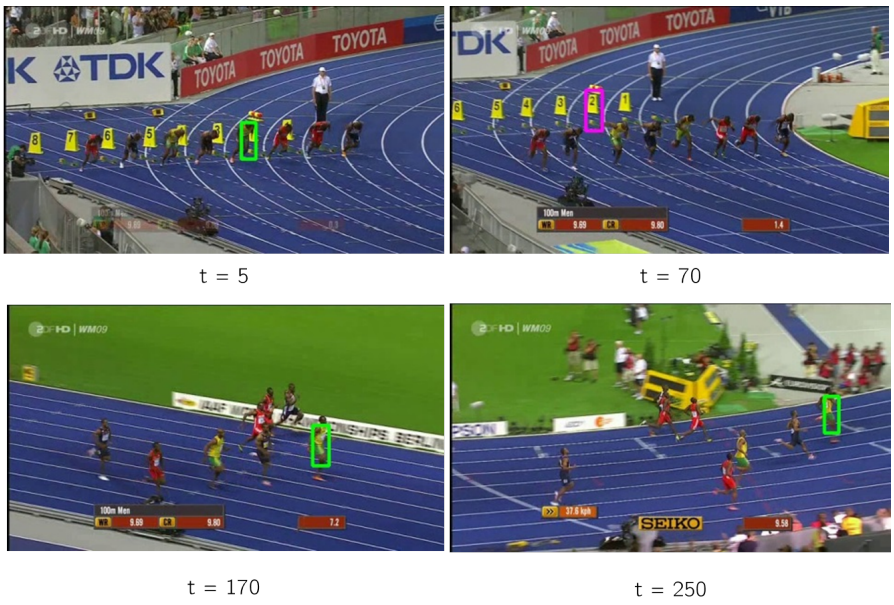


FIGURE 3.16: Comparison of tracking results on the "bolt" video for (b) Best Confidence (BC) (*white*: ground-truth, *pink*: Haar tracker, *blue*: HOG tracker, *green*: HOC Tracker).

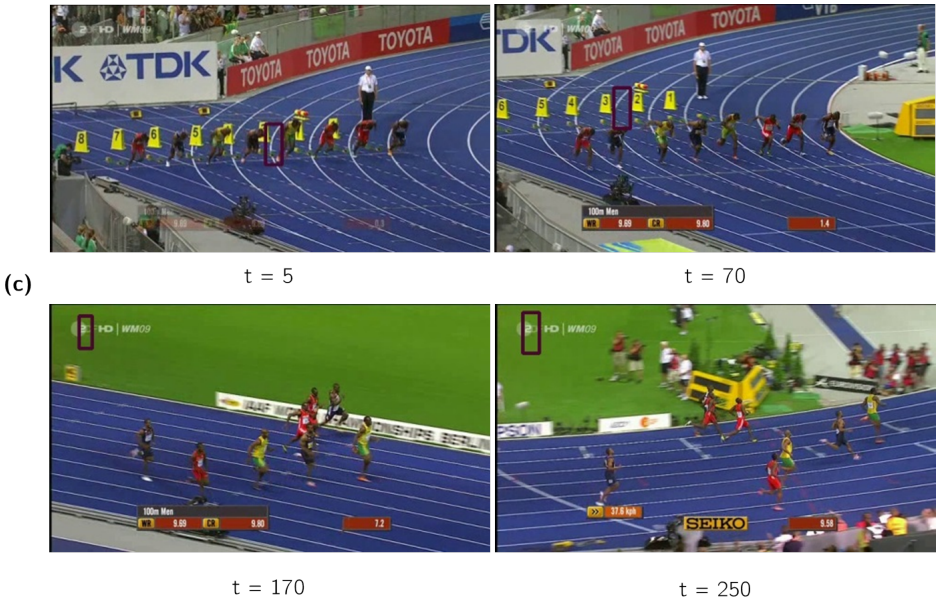


FIGURE 3.17: Comparison of tracking results on the "bolt" video for (c) Fusion of Features (FoF).

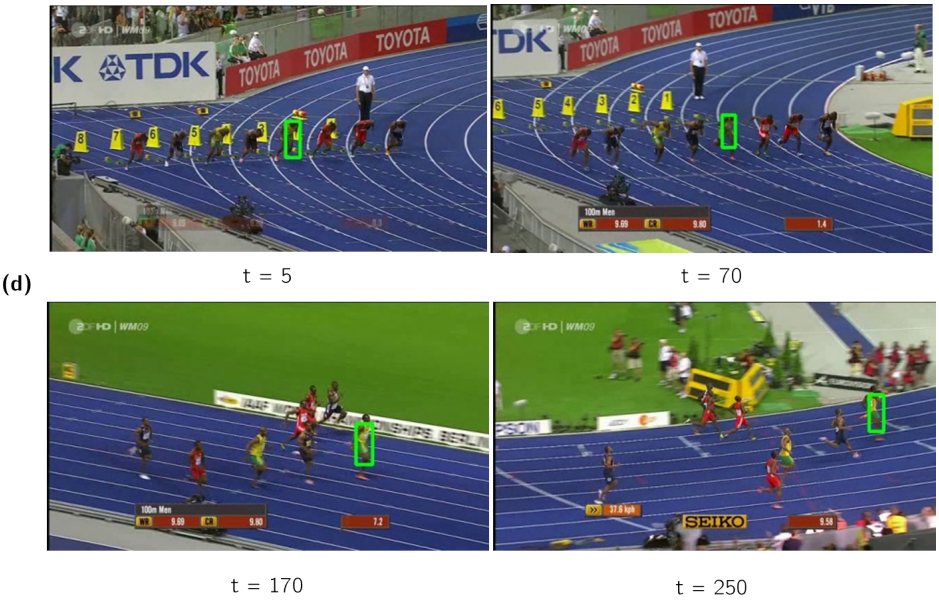


FIGURE 3.18: Comparison of tracking results on the "bolt" video for (d) the proposed method (PM)(*white*: ground-truth, *pink*: OAB Haar tracker, *blue*: OAB HOG tracker, *green*: OAB HOC Tracker).

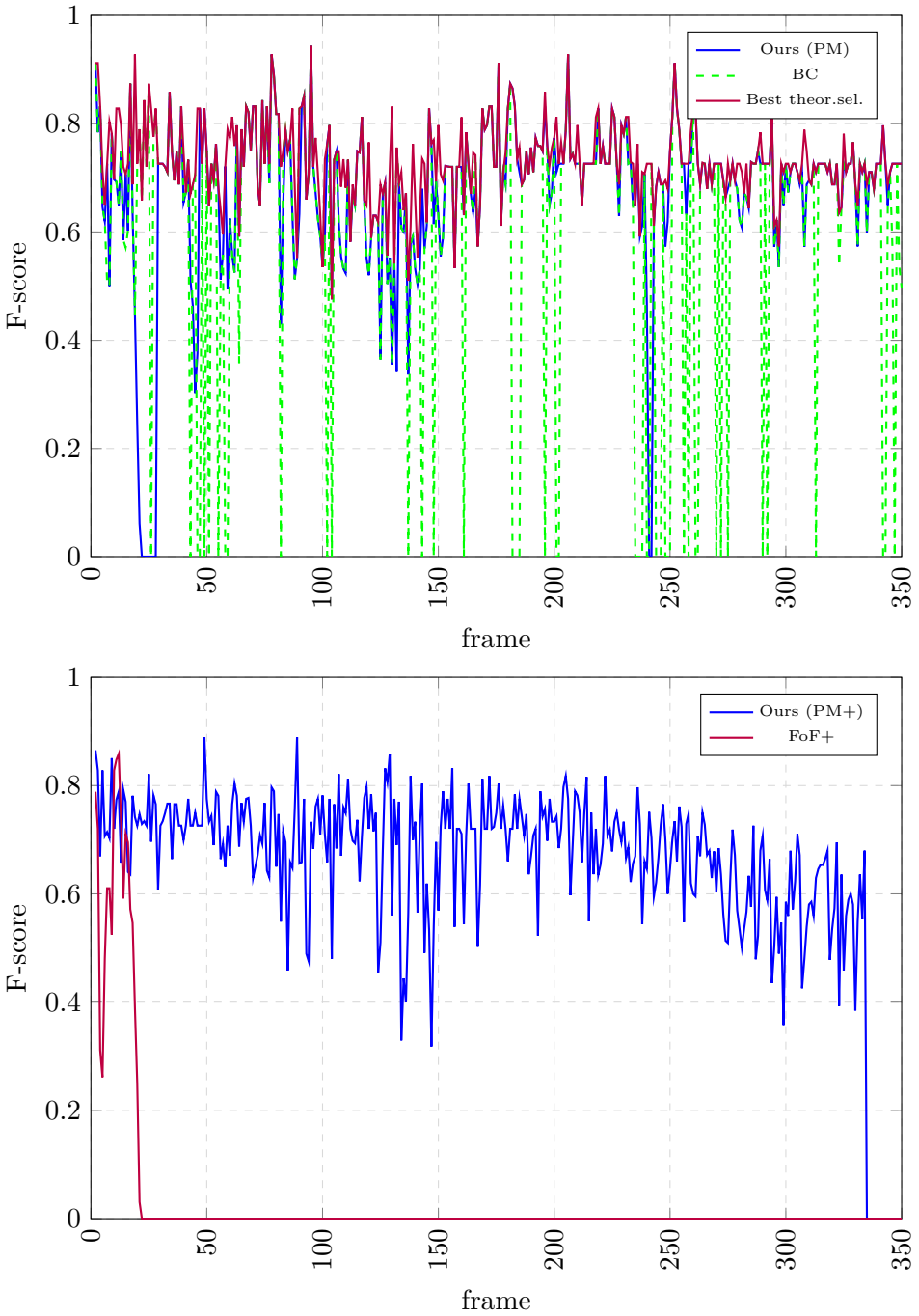


FIGURE 3.19: F-score measures of the "bolt" video, (1^{st} row) proposed method (PM), Best confidence (BC), best theoretical results and (2^{nd} row) proposed method with selective update (PM+), Fusion of Features with minimal update (FoF+).

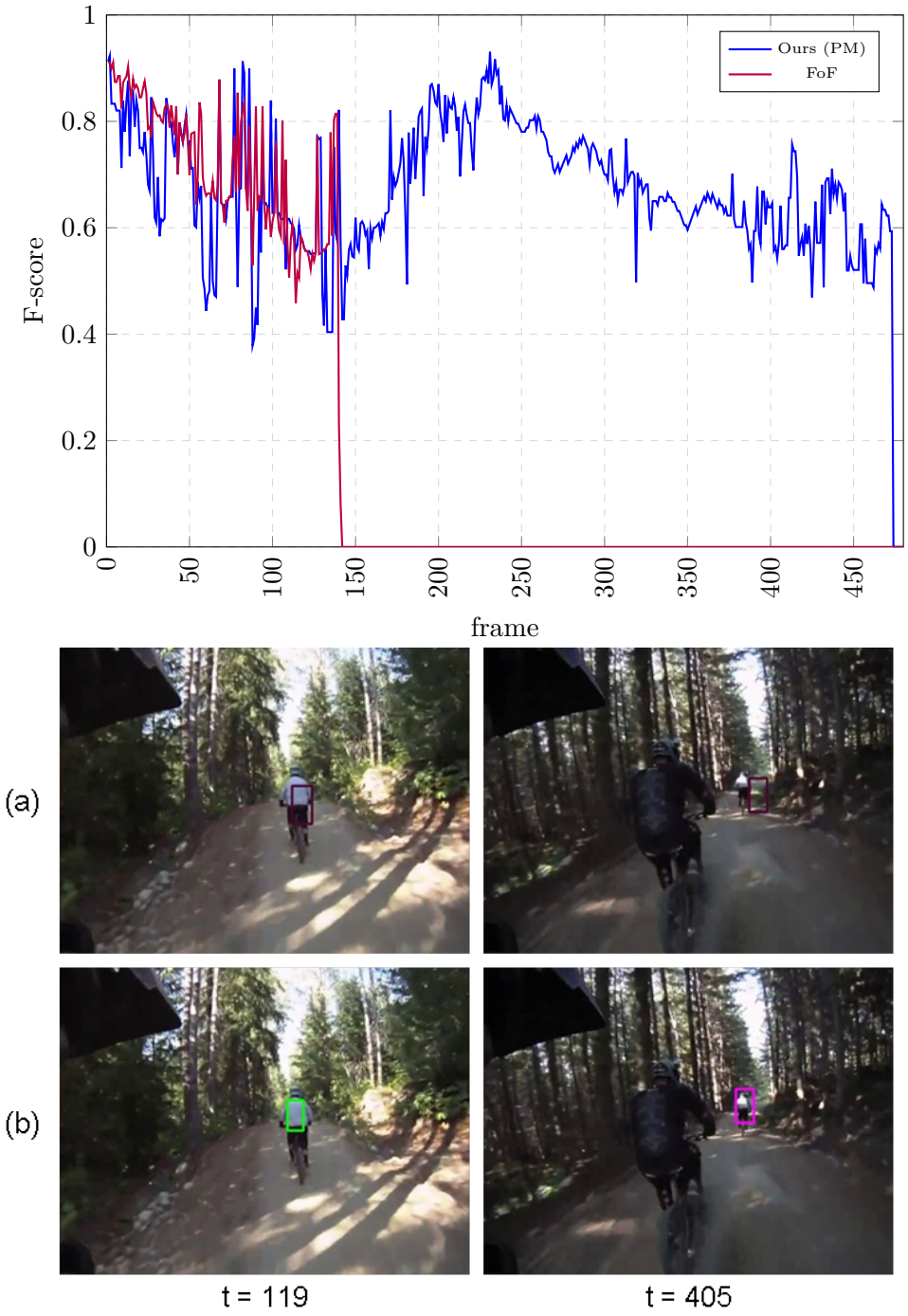


FIGURE 3.20: Comparison of results on the "dh" video for (a) Fusion of Features (FoF), (b) the proposed method (PM) where (*pink*: Haar, *blue*: HOG, *green*: HOC Tracker).

has a very discontinuous tracking while PM (figure 3.18) succeeds to "lock" on the OAB HOC tracker.

The proposed method also outperforms the Fusion of Features (FoF) approach. In this approach, the features are not independent, since they are all updated with the same data. As a consequence, as soon as a tracker begins to drift, wrong data or noise is introduced, and it is impossible for the trackers to recover as illustrated in figure 3.20 (a). In figure 3.17, the FoF method started drifting in frame 5, and completely loses the object afterwards. Unlike this approach, PM (d) uses independent trackers. Thus, the selection scheme makes it possible to switch from one tracker to another if it is lost. PM succeeds in tracking the object at almost the performance of the best theoretical results (figure 3.19, 1^{st}row).

As for the Centroid of Trackers, the low success rate (Table 3.1) shows that fusion approach is not effective as the individual trackers do not always perform well simultaneously, and a lost tracker may considerably disturb the overall result.

Finally, the bottom of Table 3.1 shows the success rates for the proposed method with selective update (PM+) and the corresponding baseline FoF+. The selective update slightly increases the performance and produces the best results among the compared methods. In figure 3.19 (2^{nd}row), we can see that PM+ successfully tracks the object whereas the FoF+ only tracks correctly for a certain time and then starts drifting and loses the object. The selective update also improves the speed by around 50%. The proposed method runs at around 6 fps (with non-optimised and single-core C++ code), where the majority of computation time is spent on feature computation and update of the AdaBoost trackers. The combination method using temporal and spatial coherence with confidence (PM) adds very little computational overhead.

3.5 Conclusion

In this chapter, we presented a simple tracker selection framework, namely Spatial and Temporal Coherence (STC), in which given a pool of trackers, the most suitable one is selected at each video frame. This framework integrates a spatial and temporal coherence criterion, a consistent confidence evaluation for tracker selection and a selective update strategy. We employed, as individual trackers, OAB-based trackers with simple low-level features Haar, HOG and HOC.

We first evaluated the performance of each individual tracker and examined the best theoretical selection possible with these trackers to understand the benefits of the combination framework. We then proceeded in evaluation the proposed selection framework, with and without selective update, and compared them with baseline combination methods. Experimental results demonstrate that, even in very challenging sequences, the proposed method improves the overall robustness respect to the individual trackers. Moreover, the proposed framework outperforms classical tracker combination strategies. We also showed that optimising the update strategy can further improve the tracking result as it is a central component of discriminative online tracking.

This selection framework has the benefit of being simple and adding very little computational complexity over the individual trackers. Using only spatial and temporal information and the confidence value, the framework succeeds in improving the overall tracking performance. However efficient, the framework doesn't take into account information from the scene context, which would help improve the selection and prediction rate and further increase the tracking performance.

Chapter 4

Classifying Global Scene Context For Tracker Selection

4.1 Introduction

The development of tracker combination frameworks is usually in the light of creating a tracking algorithm capable of coping with the different variations in scene conditions while maintaining a robust tracking performance. Thus, it makes sense to include the scene context directly in the combination, selection or fusion mechanism or decision making.

"Context" can represent any knowledge on the object or its surroundings: from the nature of the object to the lighting conditions in the scene, or the location or background in which the tracking is taking place, *etc.* All this information is of great importance, and being able to extract or extrapolate this prior knowledge can immensely improve the overall tracking.

In a tracker combination framework, since the performance of each tracker depends on the scene context (as previously discussed in chapter 3), the most intuitive approach is to directly use the scene context and recurrently select one tracker from a pool of complementary trackers depending on the scene context.

Along these lines, we present a novel framework for combining several independent online trackers in a principled way, based on the visual scene context. The aim of our method is to decide, automatically and iteratively at each point in time, which specific tracking algorithm works best under the given scene or acquisition conditions. To this end, we propose a set of generic global context features tailored for this task, and a classifier trained to predict for a video frame the tracker that performs best, given the global scene context.

In this chapter, we will first briefly investigate the use of "context" in visual tracking literature. We will propose a Scene Context-Based Tracker selection framework (SCBT), in which context information is extracted from the scene and used to select the most suitable tracker at each video frame. Finally, we present experimental results showing the effectiveness of this approach in predicting the best tracker and in increasing the overall robustness and accuracy with respect to the individual trackers.

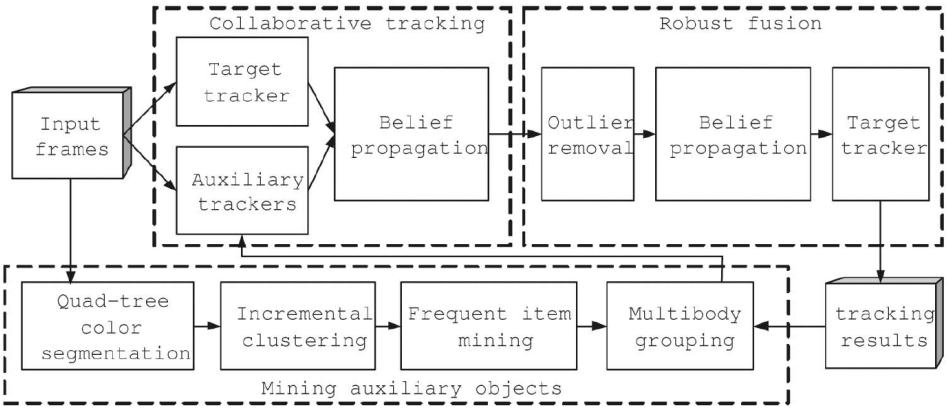


FIGURE 4.1: Block diagram of the CAT algorithm [YWH09] presenting the three sub-modules: auxiliary object mining, collaborative tracking, and robust fusion, enclosed in dash rectangles. Image courtesy of [YWH09].

4.2 State-of-the-art of Context in Tracking

Scene context has been previously included in visual tracking and detection in various forms. Especially in online detection or tracking scenarios, where information for model construction is very limited to the number of frames, context can provide important information and greatly improve the performance. Context can be used in many ways.

Some works [YWH09; Gra+10; Wen+14] use so-called "supporters" or "contributors" to help assist the tracking by detecting image regions or interest points that move similarly to the tracked object.

In the aim of considering the context of the tracking scene in the actual tracking, Yang *et al.* [YWH09] proposed the Context-Aware Visual Tracking (CAT) algorithm. The context takes form of a set of auxiliary objects (*i.e.* supporters) that are automatically discovered in the video on the fly by data mining and integrated into the tracking process. These auxiliary objects are defined as easy to track objects with persistent co-occurrence and consistent motion with the target object. The aim of the auxiliary objects is to verify the target object tracking results as they are tracked along side the object in a collaborative way. The CAT framework, illustrated in figure 4.1, is mainly composed of three important modules. 1) Mining auxiliary objects, where auxiliary objects are automatically chosen using data mining techniques and multi-body grouping to discover the potential multi-body structure from motion and to estimate the affine motion models through subspace analysis; 2) Collaborative tracking, where the target object and the set of auxiliary objects are collaboratively tracked by formulating the problem as a random field model; 3) Robust fusion, applied to handle inconsistency among the target object and the auxiliary object trackers.

In a similar way, Grabner *et al.* [Gra+10] proposed to learn supporters, which are useful features helping to determine the position of the object of interest. Exploiting the General Hough Transform strategy, this approach couples the supporters with the target and naturally distinguishes between strongly and weakly coupled motions. The supporters vote to determine the target object location, and the position of an object can be estimated

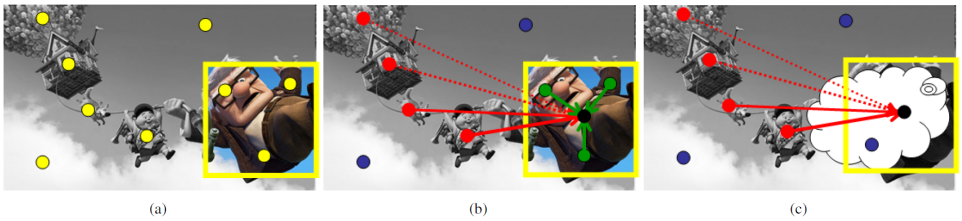


FIGURE 4.2: Principle idea of tracking with supporters [Gra+10]. (a) A frame with the target object marked, and local image features (yellow). (b) Supporters are features that vote for the position of the object, since their motion appears correlated. They can belong to the object itself (green) or not (red). Uncorrelated features (blue) are discarded. (c) Even if the object cannot be tracked based on its appearance (*e.g.* when occluded or its appearance changed), the supporters can help to infer its position. Image courtesy of [Gra+10].

even when it is fully occluded or when it changes its appearance quickly and significantly (*c.f.* figure 4.2).

Other tracking methods have exploited spatio-temporal context in order to improve the performance. For instance, Zhang *et al.* [Zha+14] exploited the spatio-temporal relationships between the object of interest and its locally dense contexts in a Bayesian framework to model the correlation between low-level features from the object and its surrounding regions. The tracking problem is then posed by computing a confidence map which takes into account the prior information on the target location and thereby is able to alleviate position ambiguities effectively. Wen *et al.* [Wen+14] employed a spatio-temporal context appearance model-based tracker, which also integrates contributors around the target object. The temporal appearance context model captures the historical appearance of the target to prevent the tracker from drifting to the background in a long-term tracking. Whereas the spatial appearance context model integrates contributors (*i.e.* patches with the same size of the target at the key-points) to build a supporting field.

Instead of supporters, other methods [DVM11; HMT12] employ "distractors", *i.e.* image regions with similar appearance to the object, in order to avoid confusion in tracking. Dinh *et al.* [DVM11] proposed a framework using both distractors and supporters which are automatically explored using a sequential randomised forest, an online template-based appearance model, and local features. Distractors are regions that have similar appearance as the target object and consistently co-occur. These regions are helpful in avoiding the tracker from drifting. On the other hand, supporters are local key-points around the target object with consistent co-occurrence and similar motion in a short time span. They help to verify the genuine target.

Duffner and Garcia [DG14] also integrated contextual motion in their tracking algorithm with the aim of extracting more efficient negative examples to update the online tracking model. In this framework, an adaptive particle filter using different visual features with motion prediction models is employed, and a discriminative online learning classifier is integrated into the framework using stochastic sampling of negative examples from contextual motion cues in videos. In fact, instead of taking negative examples only

from the surroundings of the object region, or from specific distracting objects, the algorithm samples the negatives from a contextual motion density function.

In general, these approaches are computationally expensive, due to the more complex data association and modelling of spatial and temporal relationships between the different tracked objects or interest points.

In a different manner, Maggio *et al.* [MC09] used contextual event cues such as target births (objects entering the scene) and clutters (spatial clutter of objects) for multi-object tracking. The spatial distribution of these events is incrementally learned using tracker feedback based on mixtures of Gaussians. The corresponding models are used by a Probability Hypothesis Density (PHD) filter that spatially modulates its strength based on the learned contextual information.

In fact, multi-object tracking can be effective in tracking a single object in a scene with multiple objects of the same nature, using the the other objects as distractors.

In our approach, supporters, distractors or contextual events are not used. They make inference less complex and error-prone due to detection or tracking errors. We rather classify the general scene context and conditions in order to select the most appropriate visual cue or tracker for a given situation. To this end, we compute global image descriptors based on colour, intensity and motion at each video frame.

In the past, other global image descriptors (sometimes called gist features) have been proposed (*e.g.* [OT01; Tor+03; SI07]) mostly for fixed images to classify scenes into different semantic categories, such as open, closed environments, indoor, outdoor *etc.* Garcia Cifuentes *et al.* [Cif+12] proposed to classifying videos into categories of camera motion in order to improve the tracking of interest points. Given a set of designed and specialized motion models, the method automatically determines which model to apply when tracking interest points in a given sequence.The concept of this framework is somewhat similar to ours, as we also aim to classifying the scene in order to improve the tracking performance. However, we want to classify the scene context in all its aspects, not only characterising the motion, and propose a general framework where any off-the-shelf trackers can be used.

Another example was the recent work by Nguyen *et al.* [NBT16], where a long term tracking framework is boosted by context around each tracklet and applied to multi-object tracking. A database of optimal tracker parameters is learned offline for various context, then during tracking, the context surrounding each tracklet is extracted and matched against database to select best tracker parameters. In this framework, the context is represented by a pool of features surrounding the tracklet. Three types of features adapted to the specific task of multi-object tracking are used: occlusion, mobile object density and contrast. Rather that classifier local features to adapt the parameters of a tracker, we aim, in our framework, to learn the global scene context in order to make a prediction of the most suitable tracker. To our knowledge, no other work exists that extracts and classifies global scene context features for visual object tracking.

4.3 Scene Context-Based Tracker Selection

Different tracking algorithms have different strengths and weaknesses. Some cope well with different lighting conditions, some are particularly robust to object deformations or occlusions, but will fail in other conditions. Due to the compromise between invariance and discriminative power, despite recent progress in on-line tracking algorithms, it is hard to design models that perform well in very different environments and contexts. Although the fusion of different visual cues generally improves the performance in that regard, it remains difficult to decide on the importance or on the weight each modality should get when the overall scene context changes, especially *within* a given video.

In order to "measure" these changes and take them into account in the tracking procedure, we propose the Scene Context-Based Tracker (SCBT) selection framework. In this framework, we combine several independent and complementary trackers, each specialised on different image and scene conditions. The decision on which tracker to select is proposed by an offline trained classifier which, in turn, is based on general scene context features that are independent from the individual trackers. To summarise, we propose:

- a set of *general scene context features* that describe the global conditions (such as lighting, camera motion) that are relevant to track an object in a video,
- a framework that combines independent trackers by using a *classifier* that estimates at each frame the most suitable tracker for the given context, and by filtering the classifier responses using a temporal Hidden Markov Model (HMM).

It is also important to mention that this approach is generic, and in theory *any* on-line tracking algorithm can be integrated in this framework.

4.3.1 Overall Approach

The general procedure of the proposed selection framework SCBT is illustrated in figure 4.3. On a given video, N independent feature-based trackers $\{T_1, .., T_N\}$ run in parallel, and at each frame t, each tracker $T_n, (n \in 1..N)$ produces an estimation of the object's position. This is usually a bounding box B_t^n with an associated confidence value (or score) $c_{t,n}$. The objective is to select at each frame the best tracker, *i.e.* the one that outputs the bounding box that fits best the object to track, by using a discriminative model.

To do so, the scene features $\mathbf{f}_t$ are extracted simultaneously from the current image. Combining the scene features $\mathbf{f}_t$, the confidence values $\mathbf{c}_t = (c_{t,1}, .., c_{t,N})$ and the previously selected tracker s_{t-1}, a large feature vector $\mathbf{i}_t$ is formed. Using this input vector $\mathbf{i}_t$, the scene context classifier makes a prediction $\mathbf{y}_t$, representing the probabilities for each tracker to be the most suitable one. It is a N-class classifier, which has been trained off-line on annotated data to estimate the best tracker for the given scene context. The classifier's prediction $\mathbf{y}_t$ is subsequently filtered by a Hidden Markov Model (HMM) to ensure some temporal continuity of the tracker selection

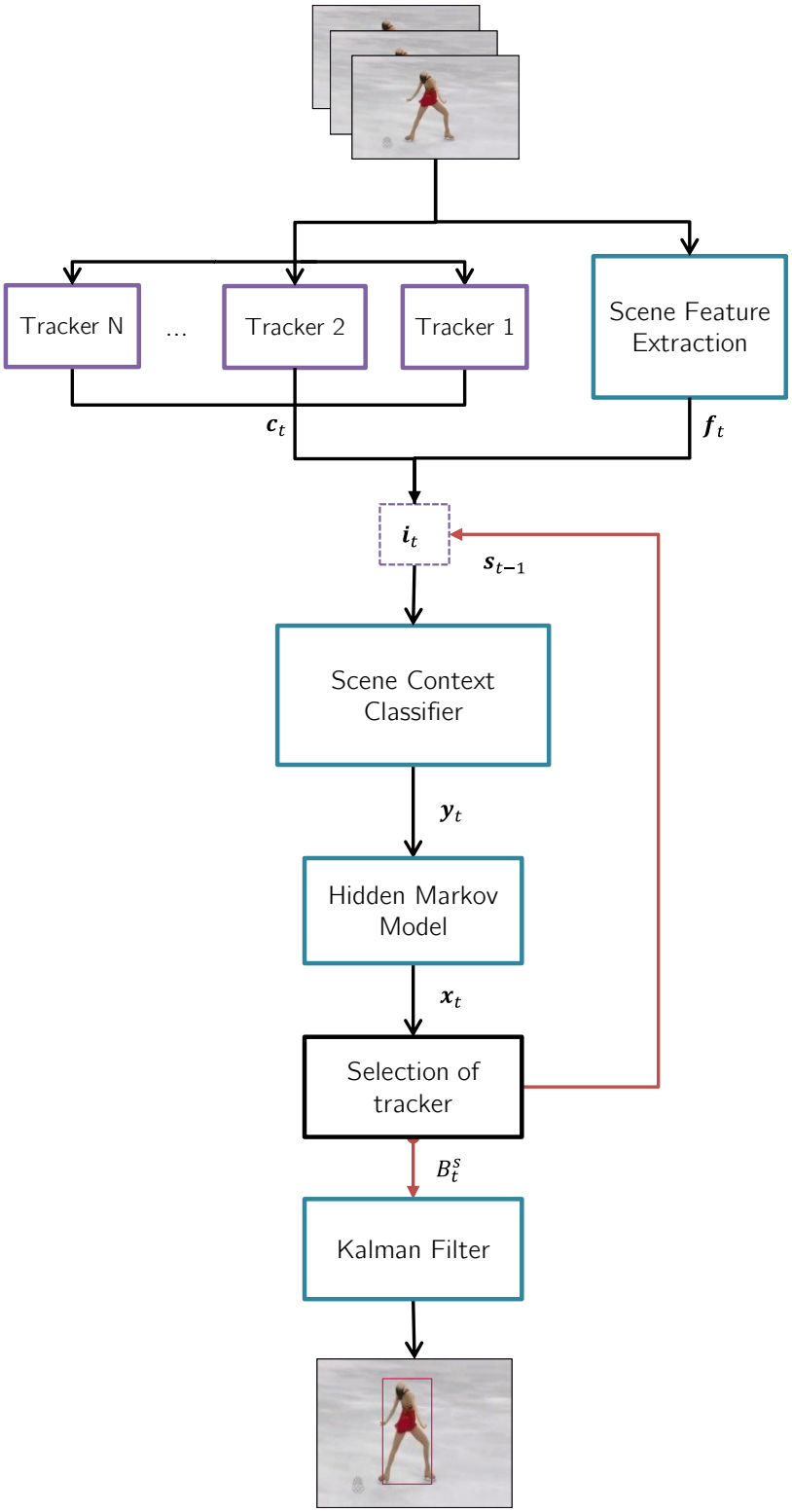


FIGURE 4.3: The overall framework of the proposed Scene Context-Based Tracker (SCBT).

and reject outliers. The tracker with the highest probability s_t, based on the filtered prediction $\mathbf{x}_t$, is selected for the current frame. Finally, to smooth the overall trajectory, the sequence of corresponding bounding boxes B_t^s is processed by a Kalman Filter as a post-processing step.

The result of the Kalman filter represents the final output of our tracking algorithm, and is further used to update the models of the individual trackers $T_n, (n \in 1..N)$. Apart from this last update step, all the trackers are completely independent and do not cooperate or interact with each other.

4.3.2 Scene Context Feature Extraction

In most existing computer vision algorithms, for example image classification or object recognition, some common visual features are used to concisely describe the content of an image or a region, or more specifically an object such as SIFT, HOG, Haar-like features, LBP, or more roughly using Gist descriptors, to cite a few. However, in our tracker combination framework, it is not of our interest to extract exact or specific information such as objects, or shapes from the scene. Our main goal is rather to globally characterise the scene through descriptors specifying its general state: is it dark? is it cluttered? is the motion homogeneous? *etc.*

To our knowledge, there are no established descriptors that capture global scene information regarding the overall environment or acquisition conditions, like lighting, camera motion, background uniformity *etc.* To quantify these phenomena is of particular interest for video analysis and object tracking algorithms. In the following, we propose a set of such descriptors that we call "*scene context features*" which are designed to help predicting the best tracker in a given environment.

Furthermore, these descriptors need to be correlated with the low-level features used in the individual trackers from our pool of trackers $\{T_1, .., T_N\}$. We thus designed the scene context features based on first and second order statistics of the main image characteristics (*e.g.* intensity, hue, motion vectors).

Let us define Ω as an image region of the input image and $f_{t,k}^{\Omega}$ as the scene feature k computed on the region Ω and frame t. To simplify, we omit the frame index t in this section.

Let p_i^{grey}, p_i^{hue}, p_i^{sat} and p_i^{flow} denote respectively the pixel value of the grey-scale image, the hue and saturation channels in HSV colour space, and the optical flow of the input image at the pixel position i. We chose the HSV colour space instead of RGB because it separates the image intensity component (which we also use for our context features) from the chroma information. This helps us design colour-based features which are invariant to intensity changes. We also define $\|\Omega\|$ as the number of pixels in the region Ω. We propose to use the following set of features, grouped into three categories and defined in equations 4.1-4.8:

Intensity and texture features

- *Average brightness* (f_1^{Ω}): the mean grey-scale pixel value over region Ω.

$$f_1^{\Omega} = \frac{\sum_{i \in \Omega} p_i^{grey}}{\|\Omega\|}. \tag{4.1}$$

- *Average contrast* (f_2^Ω): the mean squared value of the difference of each grey-scale pixel and the average brightness over Ω.

$$f_2^\Omega = \frac{1}{\|\Omega\|} \cdot \sum_{i \in \Omega} \left(p_i^{grey} - \frac{\sum_{i \in \Omega} p_i^{grey}}{\|\Omega\|} \right)^2. \tag{4.2}$$

Chromatic features

- *Average saturation* (f_3^Ω): the mean pixel value of the saturation channel in HSV colour space over the region Ω.

$$f_3^\Omega = \frac{\sum_{i \in \Omega} p_i^{sat}}{\|\Omega\|}. \tag{4.3}$$

- *Saturation variance* (f_4^Ω): the variance of saturation over Ω.

$$f_4^\Omega = \frac{\sum_{i \in \Omega} (p_i^{sat})^2}{\|\Omega\|} - \left(\frac{\sum_{i \in \Omega} p_i^{sat}}{\|\Omega\|} \right)^2. \tag{4.4}$$

- *Dominant hue* (f_5^Ω): the dominant colour of the region Ω extracted from a histogram of quantised hue pixel values in HSV colour space.

$$f_5^\Omega = \underset{i \in \Omega}{\operatorname{argmax}}(p_i^{hue}). \tag{4.5}$$

- *Hue variance* (f_6^Ω): the variance of the pixel values in the hue channel.

$$f_6^\Omega = \frac{\sum_{i \in \Omega} (p_i^{hue})^2}{\|\Omega\|} - \left(\frac{\sum_{i \in \Omega} p_i^{hue}}{\|\Omega\|} \right)^2. \tag{4.6}$$

Motion features

- *Average motion* (f_7^Ω): the mean of the norm of optical flow vectors densely computed over the region Ω.

$$f_7^\Omega = \frac{\sum_{i \in \Omega} p_i^{flow}}{\|\Omega\|}. \tag{4.7}$$

- *motion variance* (f_8^Ω): the variance of the norm of dense optical flow vectors over Ω.

$$f_8^\Omega = \frac{\sum_{i \in \Omega} (p_i^{flow})^2}{\|\Omega\|} - \left(\frac{\sum_{i \in \Omega} p_i^{flow}}{\|\Omega\|} \right)^2. \tag{4.8}$$

Using three different image regions Ω^L, Ω^G and Ω^B illustrated in figure 4.4, we define three values for each of features $f_k^\Omega, (k \in 1..8)$.

- The **global value** is defined as the feature computed on the whole image Ω^G: f_k^G.
- The **local value** is the feature computed on the Region Of Interest (ROI) Ω^L: f_k^L.

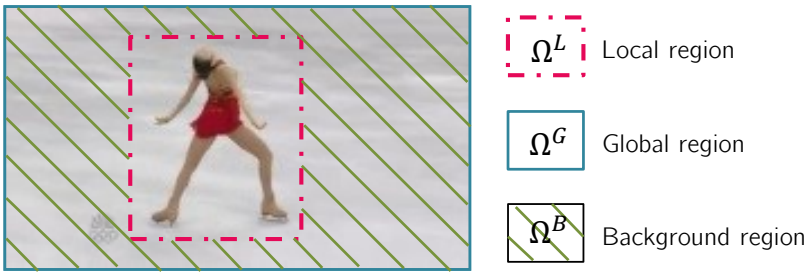


FIGURE 4.4: The different image regions used to compute scene features.

- The **differential value** as the difference between the feature computed on the foreground region (*i.e.* the ROI Ω^L) and the background region (*i.e.* the image not including the ROI, Ω^B): $f_k^D = f_k^L - f_k^B$.

Not every combination of feature and region is used as some of them have little semantic meaning. The concatenation of these features gives us:

$$\begin{aligned} \mathbf{f}^G &= \left\{ f_1^G, f_2^G, f_3^G, f_6^G, f_7^G, f_8^G \right\} \\ \mathbf{f}^L &= \left\{ f_1^L, f_2^L, f_3^L, f_4^L, f_5^L, f_6^L, f_7^L, f_8^L \right\} \\ \mathbf{f}^D &= \left\{ f_1^D, f_2^D, f_3^D, f_4^D, f_5^D, f_6^D, f_7^D \right\}. \end{aligned} \tag{4.9}$$

Finally, we obtain $M = 21$ scene context features $\mathbf{f}_t = \left\{ \mathbf{f}_t^G, \mathbf{f}_t^L, \mathbf{f}_t^D \right\}$ for the frame t of a given video.

The advantage of the proposed scene features is that they represent a very concise and generic description of a visual scene which can be computer very efficiently and can easily be pit in relation with the types of features used by most existing tracking algorithms.

4.3.3 Scene Context Classifier

The scene context classifier's goal is to learn the different patterns that show high correlation between the information extracted from the scene context (*i.e.* the scene features) and the performance of a tracker in the particular set of conditions. As multi-class classifier, we chose a fully connected Multi-Layer Perceptron (MLP) with one hidden layer, N output neurons and sigmoid activation functions. Any other algorithm could be used as well. In fact, a multi-class SVM showed equivalent performance in our experiments. However, it was relatively sensitive to the choice of hyper-parameters (*e.g.* the type of kernel).

As shown in figure 4.3, the classifier's input $\mathbf{i}_t$ at a frame t consists of several components: The *scene context features* $\mathbf{f}_t$ extracted from the input image characterising the scene, the *confidence* values of the N trackers $\mathbf{c}_t = (c_{t,1}..c_{t,N})$ providing the classifier with a measure of reliability of each tracker's result, and finally, the *identifier* s_{t-1} of the tracker that has been selected in the previous frame $t-1$. We will experimentally show that this recursion highly contributes to learning the correlation between the scene context features and the selected tracker in a given frame.

Furthermore, in order to give the classifier information on the evolution of the scene context over time, we additionally provide it with the features

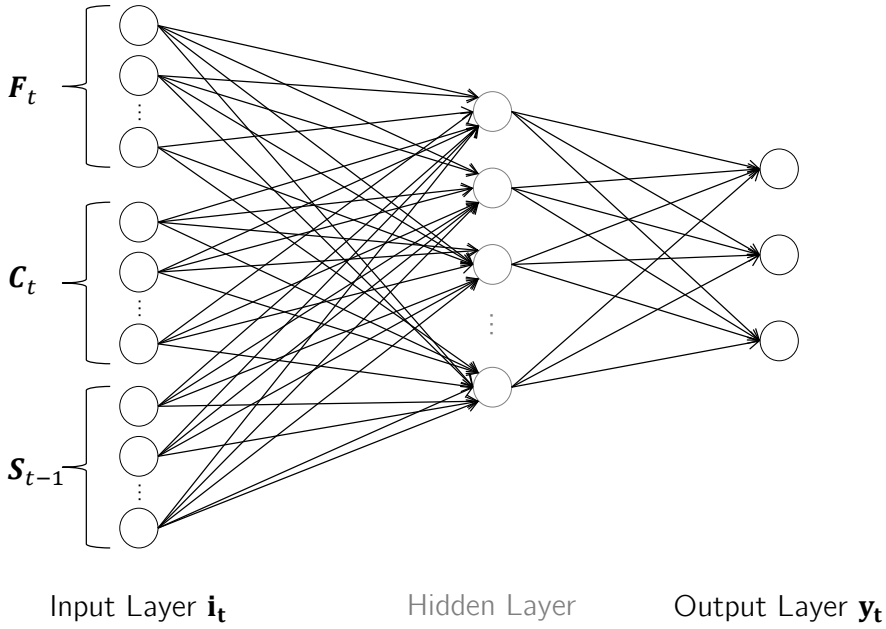


FIGURE 4.5: Representation of the Neural Network used as scene context classifier in our SCBT framework.

from the two previous frames $t-1$ and $t-2$. Incorporating the temporal aspect has proven to be effective, as shown in our experimental results. In fact, it provides the classifier with information over the evolution in time of the scene context.

We hence form a sliding window of 3 frames with the following vectors:

$$\begin{aligned} \mathbf{F}_t &= \{\mathbf{f}_t, \mathbf{f}_{t-1}, \mathbf{f}_{t-2}\}, \\ \mathbf{C}_t &= \{\mathbf{c}_t, \mathbf{c}_{t-1}, \mathbf{c}_{t-2}\}, \\ \mathbf{S}_{t-1} &= \{s_{t-1}, s_{t-2}, s_{t-3}\}. \end{aligned} \tag{4.10}$$

and the final feature vector given as input to the classifier is the following:

$$\mathbf{i}_t = \{\mathbf{F}_t, \mathbf{C}_t, \mathbf{S}_{t-1}\}. \tag{4.11}$$

The neural network classifier, illustrated in figure 4.5, is trained off-line on a dataset with annotated object bounding boxes. Let us consider a training sample $\left\{\mathbf{i}_j, o_j^*\right\}$, $(j \in 1..N_{train})$ where N_{train} is number of training samples, $\mathbf{i}_j$ is the input vector and o_j^* is the corresponding *label* for the sample j, described below.

In order to construct the classifier input vector $\mathbf{i}_j = \{\mathbf{F}_j, \mathbf{C}_j, \mathbf{S}_{j-1}\}$, we run the N trackers on each video of the training dataset, and at each frame (*i.e.* training sample j), we extract the scene context features $\mathbf{F}_j$ as well as the trackers' confidences $\mathbf{C}_j$ and save the identifier of best tracker to be used as the "previously selected tracker" $\mathbf{S}_{j-1}$ in the following frame. The most accurate tracker for each sample j, *i.e.* the label o_j^*, is determined by computing the F-scores (*c.f.* equation 3.11) $Fscore_j^n$ of each tracker T_n. The

tracker with the highest F-score is considered the label o_j^* of the sample:

$$o_j^* = \underset{n \in 1..N}{\operatorname{argmax}}(Fscore_j^n). \tag{4.12}$$

We optimise the neural network parameters with standard stochastic gradient descent by minimising the mean squared error between the network's response vector $\mathbf{y}_j = (y_{j,1}, .., y_{j,N})$ and the desired output vector $\mathbf{y}_j^*$. The usual output strategy (one-of-N encoding) is used, where $\mathbf{y}_j^* \in \{-1, +1\}^N$ with $+1$ for the component corresponding to the sample label o_j^* and -1 otherwise. The network's final class prediction is simply $o_j = \operatorname{argmax}_{n \in N} \mathbf{y}_{j,n}$. We perform early stopping using a separate validation set.

4.3.4 Tracking Procedure

The proposed algorithm uses N independent on-line trackers that are initialised with the bounding box of the object in the first video frame. Then, as illustrated in figure 4.3, the trackers and the context feature extraction operate in parallel providing at each frame N confidence values $\mathbf{C}_t$ and M scene context features $\mathbf{F}_t$ respectively.

At each video frame t, the scene context classifier estimates probabilities for each tracker $\mathbf{y}_t$ to perform best under the current scene context. Then, a Hidden Markov Model (HMM) and Kalman filter are employed subsequently.

Hidden Markov Model

To avoid frequent and unnecessary switching between different trackers, we filter the classifier responses in time using a Hidden Markov Model (HMM).

The HMM is used to estimate the discrete hidden variable $x_t \in \{1..N\}$ corresponding to the best tracker selection, and it receives the observations $\mathbf{y}_t$ being the output of the scene context classifier. Another observation variable $\mathbf{d}_t = (d_{t,1}..d_{t,N})$ is added and defined as the normalised distances of each tracker's resulting bounding box B_t^n to the previous estimated object position. Using the HMM, we want to estimate the posterior probability distribution of x_t, which we can compute recursively:

$$p(x_t|\mathbf{y}_t, \mathbf{d}_t) = p(\mathbf{y}_t|x_t) \int p(x_t|x_{t-1}, \mathbf{d}_t) p(x_{t-1}|\mathbf{y}_{t-1}, \mathbf{d}_{t-1}) \mathrm{d}x_{t-1} \,. \tag{4.13}$$

To simplify model parameter estimation, we assume that observations $\mathbf{d}_t$ and the hidden variable x_{t-1} are independent. This gives us:

$$p(x_t|\mathbf{y}_t, \mathbf{d}_t) = p(\mathbf{y}_t|x_t) p(x_t|\mathbf{d}_t) \int p(x_t|x_{t-1}) p(x_{t-1}|\mathbf{y}_{t-1}, \mathbf{d}_{t-1}) \mathrm{d}x_{t-1} \tag{4.14}$$

The likelihood function $p(\mathbf{y}_t|x_t)$ of the observed classifier responses and the probability $p(x_t|\mathbf{d}_t)$ of selecting a tracker given the distances to the previous bounding box are modelled by histograms computed on a separate training set. The transition probability $p(x_t|x_{t-1})$ is set empirically to achieve a reasonable continuity of the HMM responses. Then, the best tracker selection according to the HMM's filtered prediction is computed as the maximum a posteriori probability:

$$s_t = \underset{s'}{\operatorname{argmax}}\, p(x_t = s'|\mathbf{y}_t, \mathbf{d}_t). \tag{4.15}$$

Kalman

Because of the selection framework, switching from tracker to tracker can introduce small spacial jumps in the objects position through the video. To counter this phenomenon and provide a smooth trajectory, a Kalman Filter is employed. The sequence of bounding boxes B_t from the selected tracker T_{s_t} are filtered over time.

From the initial state, *i.e.* ground-truth bounding box from first frame, the Kalman filter makes a prediction (a bounding box, *i.e.* position and dimensions of the object) at each frame, which is corrected by the measurement B_t of the selected result. Thus, the Kalman filter helps to ensure the spatial continuity of the tracking.

General Update

The last step of the tracking procedure is the *general update* of the trackers with the filtered bounding box. The individual trackers train their models online. They use the ground truth bounding box of the first frame to initialise their models and update them every frame once a prediction is made using the selected bounding box B_t^s processed by the Kalman filter.

In the STC framework, we employed a selective update strategy which had its advantages. However, one limitation of this strategy is to not be able to reuse a tracker which would have drifted. If a tracker drifts in the first part of the video, but then a scene condition where this tracker would be the most appropriate comes, it wouldn't be possible to reuse this tracker. This motivates our choice of updating all the trackers with the resulting bounding box in order to avoid their respective drift.

4.4 Performance Evaluation

For our proposed tracker selection framework SCBT, we employed the same $N = 3$ On-line AdaBoost (OAB) trackers described previously in section 3.3.2, with the visual cues: Haar-like features (HAAR), Histograms of Oriented Gradients (HOG) and Histograms of Colour (HOC).

We evaluate the performance of the SCBT framework in two ways. First, we measured the classification rate of the proposed scene context classifier when trained on the different groups of features described in section 4.3.2. Secondly, we evaluated the overall tracking algorithm on a public benchmark analysing the contribution of the different components, *i.e.* scene context classifier, HMM, and Kalman filter. The experiments were conducted using the following datasets:

Training dataset. The context-based classifier is trained on the Princeton Tracking Benchmark Dataset, presented in section 2.5.2. It contains 100 RGB-D videos with a diverse set of object types, backgrounds, and changes in illumination, appearance and motion.

Evaluation dataset. The evaluation of context-based classifier, as well as for the overall tracking framework was conducted on the publicly available Visual Object Tracking (VOT2013) benchmark detailed in section 2.5.3. The VOT2013 dataset contains 16 image sequences collected from well-known tracking evaluations, they cover most of the challenging situations in object

tracking: scale variance, complex backgrounds, occlusions, object deformation *etc.*

4.4.1 Classifier Evaluation

In order to understand the relevance of the different scene context features, we conducted a series of experiments related to the scene context classifier. Using the same Neural Network architecture, *i.e.* one hidden layer with N neurones, we trained multiple classifiers with different sets of features.

We first separated the scene features into the three group presented in section 4.3.2: *local features* $\mathbf{f}_t^L$, *global features* $\mathbf{f}_t^G$ and *differential* $\mathbf{f}_t^D$.

Then we add the different features computed on three consecutive time steps (*i.e.* the sliding window) to form three vectors corresponding to the scene features $\mathbf{F}_t$, the confidences $\mathbf{C}_t$ and the previous tracker identifiers $\mathbf{S}_{t-1}$:

$$\mathbf{F}_t = \{\mathbf{f}_t, \mathbf{f}_{t-1}, \mathbf{f}_{t-2}\}, \tag{4.16}$$

$$\mathbf{C}_t = \{\mathbf{c}_t, \mathbf{c}_{t-1}, \mathbf{c}_{t-2}\}, \tag{4.17}$$

$$\mathbf{S}_{t-1} = \{s_{t-1}, s_{t-2}, s_{t-3}\}. \tag{4.18}$$

We also experimented with different numbers of time steps as we combined the different components (scene features, confidences and previously selected tracker) using only 1 frame: $\{\mathbf{F}_t^{1f}, \mathbf{C}_t^{1f}, S_{t-1}^{1f}\}$, where:

$$\mathbf{F}_t^{1f} = \{\mathbf{f}_t\}, \tag{4.19}$$

$$\mathbf{C}_t^{1f} = \{\mathbf{c}_t\}, \tag{4.20}$$

$$S_{t-1}^{1f} = \{s_{t-1}\}. \tag{4.21}$$

Then using the different components from 2 frames $\{\mathbf{F}_t^{2f}, \mathbf{C}_t^{2f}, \mathbf{S}_{t-1}^{2f}\}$ where:

$$\mathbf{F}_t^{2f} = \{\mathbf{f}_t, \mathbf{f}_{t-1}\}, \tag{4.22}$$

$$\mathbf{C}_t^{2f} = \{\mathbf{c}_t, \mathbf{c}_{t-1}\}, \tag{4.23}$$

$$\mathbf{S}_{t-1}^{2f} = \{s_{t-1}, s_{t-2}\}. \tag{4.24}$$

And finally the proposed 3-frame window $\{\mathbf{F}_t, \mathbf{C}_t, \mathbf{S}_{t-1}\}$.

For each scene features set, the classifier is trained offline on the Princeton dataset, and then evaluated on the VOT2013 dataset. The results for the different combinations are shown in table 4.1. The recognition rate represents the proportion of frames over the evaluation dataset, where the classifier has successfully predicted the best tracker, *i.e.* the tracker with the highest F-score, knowing the ground-truth.

First, we observe that the combination of *local*, *global* and *differential* scene features gives only a low recognition rate of 30.03%, however when introducing features over several time steps $\mathbf{F}_t$, we achieve a slightly higher rate of 35.80%. These results show that the scene context features alone are not sufficient to make a prediction.

However, by adding the confidence values $\mathbf{C}_t$ and the previous tracker identifier $\mathbf{S}_{t-1}$, the rate is considerably increased to 81.80%. Given the low recognition rates without $\mathbf{S}_{t-1}$, one might think that the classifier's decision

Classifier input $\mathbf{i}_t$	Frame Window Size	Recognition Rate
$\left\{\mathbf{f}_t^L\right\}$	1	26.28%
$\left\{\mathbf{f}_t^L, \mathbf{f}_t^G\right\}$	1	27.25%
$\left\{\mathbf{f}_t^L, \mathbf{f}_t^G, \mathbf{f}_t^D\right\} = \mathbf{f}_t$	1	30.03%
$\{\mathbf{F}_t\}$	3	35.80%
$\{\mathbf{F}_t, \mathbf{C}_t\}$	3	40.06%
$\{\mathbf{F}_t, \mathbf{C}_t, \mathbf{S}_{t-1}\}$	3	**81.80%**
$\left\{\mathbf{F}_t^{1f}, \mathbf{C}_t^{1f}, S_{t-1}^{1f}\right\}$	1	49.82%
$\left\{\mathbf{F}_t^{2f}, \mathbf{C}_t^{2f}, \mathbf{S}_{t-1}^{2f}\right\}$	2	74.05%
$\{\mathbf{F}_t, \mathbf{C}_t, \mathbf{S}_{t-1}\}$	3	**81.80%**

TABLE 4.1: Recognition results for different classifier inputs on the VOT2013 database.

Method	Accuracy	Failures
Best Confidence (BC)	0.559	3.513
Previous work (STC)	0.562	0.792
Context Classifier	0.540	1.617
Context Classifier + HMM	**0.574**	0.887
Context Classifier + HMM + Kalman Filter (SCBT)	0.553	**0.583**

TABLE 4.2: VOT2013 Benchmark results for the proposed method, the Best Confidence (BC) baseline and the Spatial and Temporal Coherence-based framework (STC) presented in chapter 3.

is mainly relying on this particular feature. But, when training solely on the features $\mathbf{S}_{t-1}$, the classifier does not converge to a viable solution as $\mathbf{S}_{t-1}$ alone does not allow for a good generalisation. It tends to just respond with the identifier of the previously selected tracker (*i.e.* the identity function). In fact, it is the association of both the context scene features and tracker features that enables the classifier to extract and learn the correlations between the scene information and the trackers' performance.

We can also observe the influence of the size of the sliding window over the recognition rate, as the rate doubles when using inputs from 3 frames rather than 1. This proves the importance of using the context from the previous frames as it allows the classifier of observing the evolution of the context.

For the remaining experiments, the input vector $\mathbf{i}_t = \{\mathbf{F}_t, \mathbf{C}_t, \mathbf{S}_{t-1}\}$ is used.

4.4.2 Tracking Evaluation

We further evaluated the performance of the tracking framework and its different components following the protocol of the VOT2013 benchmark, which allows us to compare our proposed framework to state-of-the-art tracking methods.

The tracking algorithm is initialised with the ground truth object bounding box in the videos' first frame and re-initialised whenever the target is lost. The benchmark provides two evaluation measures: *"accuracy"*, *i.e.* the average overlap with the ground truth, and *"failures"*, *i.e.* the robustness of the algorithm counting the number of times the tracking is lost (the measures are detailed in section 2.5.3). Here, we present the (sequence-based) average values for the whole dataset.

In table 4.2, the results for our proposed tracking method and its different components are presented along with a baseline method called "Best Confidence" (BC) that selects the best tracker only by the maximum confidence value, as well as the STC selection framework proposed in chapter 3. A classification based on the global scene context features considerably reduces the number of failures compared to using only the confidence values in BC. We can also see that both the HMM and the Kalman Filter greatly improve the robustness. However, when decreasing the number of failures the accuracy decreases slightly as well. These two additional components of the algorithm are important for the *continuity* of the tracking and, at the same time, ensure that a *different*, more suitable tracker can be selected whenever drastic scene changes occur.

We further compare the proposed framework SCBT (*i.e.* including HMM and Kalman Filter) with other state-of-the-art tracking algorithms, as well as the three individual OAB-based trackers. Figure 4.6 shows the ranking and Accuracy-Robustness plots. The proposed algorithm SCBT (marked Ours) shows to increase the robustness of our individual trackers HAAR, HOG and HOC. In fact, the SCBT framework ranks among the top trackers of the challenge in terms of robustness, outperforming for example EDFT [Fel13], Struck [HSHT11] and FoT [MV11] methods as well as the previously proposed framework STC. This proves that, using the same individual trackers, including the scene context into the selection framework increases the overall tracking robustness.

On the other hand, the accuracy of our method is directly linked and dependent on the accuracy of the individual trackers. Note that we fixed the scale of the trackers in our experiments as the robustness of OAB generally decreased when adapting to scale. Using more accurate trackers would, in theory, increase the general accuracy of our proposed method as well as the robustness. Nevertheless, we demonstrated that our framework is able to combine the strengths of each OAB tracker and enhance the overall robustness.

Finally, in figure 4.7, qualitative tracking results are presented. We can observe that for different scene condition throughout a video, the selection framework chooses the adequate tracker. In the "David" video, the HOG-based tracker is the selected one as the scene is dark lit and colour or intensity features would fail in this scene. However, once the object is in a brighter scene, the other two trackers can be used.

4.5 Conclusion

In this chapter, we proposed a novel tracker selection framework based on scene context, SCBT. We used a classifier to learn the patterns that relate the scene context information with the "suitability" of specific independent

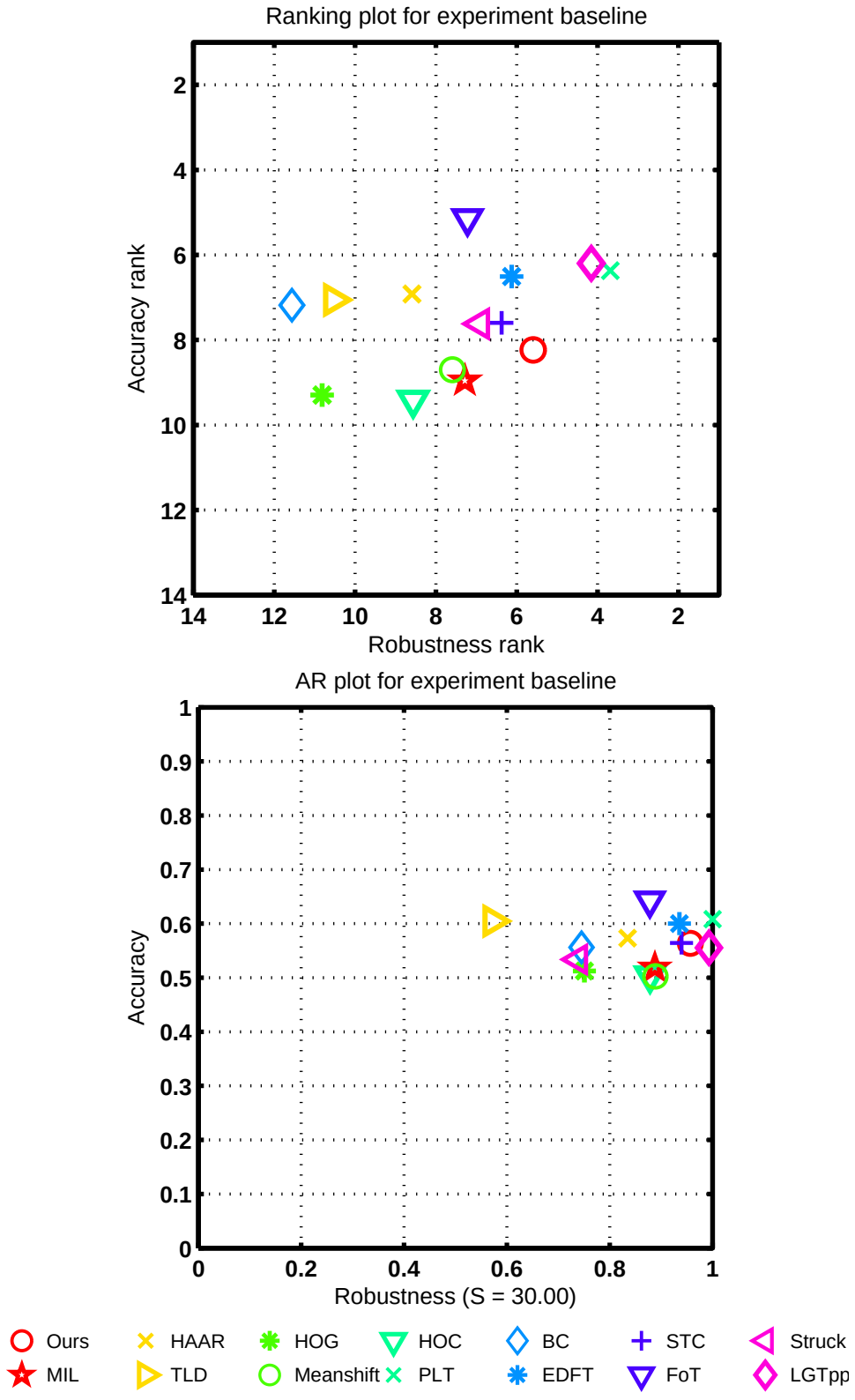


FIGURE 4.6: VOT2013 Benchmark ranking (1^{st}) and Accuracy-Robustness (2^{nd}) plots. Comparison of the proposed SCBT framework (Ours) with Online AdaBoost trackers HAAR, HOG, HOC ; baseline BC ; selection framework STC (*c.f.* chapter 3) and state of the art trackers: Struck [HSHT11], MIL [BYB11], TLD [KMM12], Meanshift [CRM03], PLT [Kri+13], EDFT [Fel13], FoT [MV11], LGT++ [CKL13].

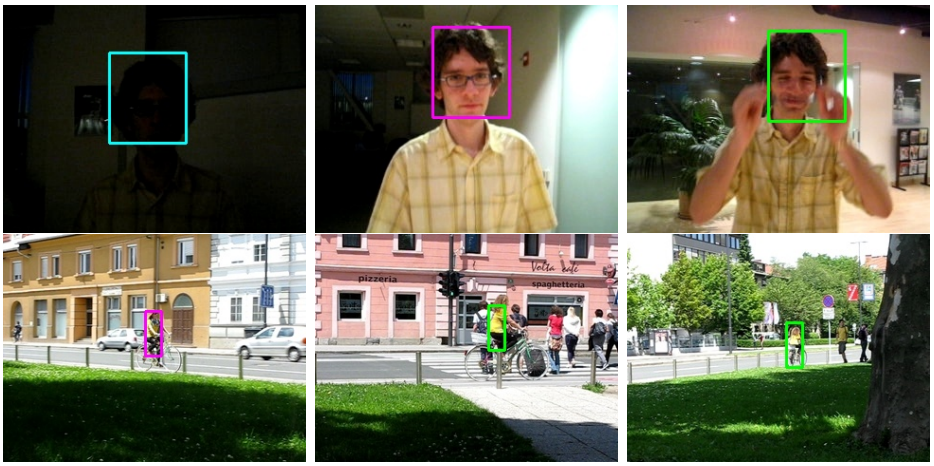


FIGURE 4.7: Illustration of our proposed framework's tracking results on the "David"(1^{st} row) and "Bicycle"(2^{nd} row) videos. Different scene context variations in lighting, texture or background are present throughout the videos. Our framework selects the most suitable tracker in each scenario (*pink*: OAB HAAR, *blue*: OAB HOG, *green*: OAB HOC).

trackers under the conditions at a given video frame. For that purpose, we designed scene context features to characterise the scene, rather than the commonly used features in tracking algorithms, such as Haar-like, HOG or SIFT features. In fact, extracting shapes or patterns for the scene is not of interest in our case, as they do not help characterise the scene in its entirety. In opposition, the designed context features extract information about the global state of the scene based on various visual cues. Moreover, the visual cues on which the context features are correlated to the complementary individual trackers employed in the framework. It is not important for the trackers to use all the possible feature domains, as long as they are complementary and not redundant. One needs to consider the processing time of the individual trackers as they need to run in parallel while keeping a real-time processing of tracking.

We also introduced an HMM to eliminate outliers and enforce the continuity in our tracking.

In our experiments with the VOT2013 benchmark, the proposed framework proved its efficiency and exceeded the individual trackers performance. Moreover, based on the same individual trackers, the SCBT framework outperformed the STC selection framework proposed in the previous chapter, showing the effectiveness of the use of context and specifically the proposed scene context classifier improving the tracking performance in challenging environments.

The SCBT framework also ranked among the top state-of-the-art trackers on the VOT2013 benchmark. With average performing in individual trackers, the selection framework is capable of improving the overall tracking performance and compete with state-of-the-art trackers.

The selection framework can be further improved, by first using more performing individual trackers, and also by optimising the training of scene context classifier.

Chapter 5

Extending the Scene Context Tracker Selection Framework

5.1 Introduction

In the Scene Context Tracker Selection (SCBT) framework, proposed in Chapter 4, we designed a preliminary selection framework using Online AdaBoost trackers with Haar-like, HOG (Histogram of Gradient) and HOC (Histogram of Colour) features.

Two components are important for the performance of the tracker selection framework: the individual trackers and the context classifier. Firstly, the trackers' individual performance directly impacts the framework as the overall tracking performance is limited by those of the different trackers. In fact, the OAB trackers perform only averagely and we assume that changing these trackers to better performing ones, while keeping the same feature domains, would make the selection framework much more robust. On the other hand, a more effective training of the scene context classifier can increase the tracking performance.

Along these lines, we proceeded in a thorough analysis and evaluation of the different components of the SCBT framework, and proposed an extended version of the Scene Context Tracker Selection framework, denoted SCBT+, with the following contributions:

- We first benchmark multiple feature-based trackers in the aim of choosing faster, more accurate and robust individual trackers to employ in the selection framework. With better performing trackers, the overall performance of the selection framework is greatly improved. We also show the importance of the choice of the individual trackers and their influence on the tracking performance of the framework.

- Using the new pool of trackers, we proceed in evaluating the scene context classifier over multiple aspects: training dataset, classifier inputs and output strategies in order to optimise the training and performance of the scene context classifier.

The results of our experimental evaluation show that the choice of trackers can be crucial in the tracking performance. Moreover, the optimised selection framework (SCBT+) proves to greatly improve the overall tracking results over multiple tracking benchmarks and compete with state-of-the-art trackers.

5.2 Extended Scene Context Tracker Selection

The extended framework SCBT+, illustrated in figure 5.1, has globally the same structure as the SCBT framework (detailed in section 4.3.1).

The proposed algorithm uses N independent online trackers which are initialised with the bounding box of the object in the first video frame. In the previous SCBT framework, Online AdaBoost trackers were used which are replaced by KCF (Kernelized Correlation Filter) trackers detailed in the following section.

Then, the trackers and the context feature extraction operate in parallel providing, at each frame t, the confidence values $\mathbf{C}_t$ and the scene context features $\mathbf{F}_t$ respectively. Using the previously selected tracker $\mathbf{S}_{t-1}$, the input vector $\mathbf{i}_t = \{\mathbf{F}_t, \mathbf{C}_t, \mathbf{S}_{t-1}\}$ is formed. Subsequently, the scene context classifier estimates probabilities for each tracker $\mathbf{y}_t$ to perform best under the current scene context, based on the input vector $\mathbf{i}_t$. The tracker with the highest probability is thus selected as the most suitable tracker $s_t = \text{argmax}_{n \in N}\, \mathbf{y}_{t,n}$. The bounding box B_t^s from the selected tracker T_t^s is then passed to a Kalman Filter to provide a smoother object trajectory. Because of the selection framework, switching from tracker to tracker can introduce small spacial jumps in the objects position through the video. The Kalman filter helps to ensure the spatial continuity of the tracking.

Finally, the last step is the *general update* of the trackers with the filtered bounding box. The individual trackers train their models online. They use the ground truth bounding box of the first frame to initialise their models and update them every frame once a prediction is made using the selected bounding box B_t^s processed by the Kalman filter.

In the previous version of the selection framework SCBT, we used a Hidden Markov Model (*c.f.* section 4.3.4) in order to avoid frequent and unnecessary switching between different trackers. However, we found that with the more robust state-of-the-art individual trackers and with our improved training procedure, the scene context classifier is more stable, and the HMM becomes unnecessary. In fact, we experimented the use of the HMM in the SCBT+ framework and found that it doesn't improve the tracking performance but the contrary. Because the classifier's training is more stable with the improvements, the predictions made are less noisy than noticed before. Thus, in this framework SCBT+, instead of minimising the excessive changes between trackers as in SCBT, the HMM cancels the correct changes from the classifier. Because of this, the HMM becomes obsolete and unnecessary in the framework.

5.2.1 Individual trackers: KCF

In the selection frameworks previously presented STC (*c.f.* chapter 3) and SCBT (*c.f.* chapter 4), three Online AdaBoost trackers [GGB06] with Haar, HOG (Histogram of Gradient), and HOC (Histogram of Colour) features were employed. In order to improve the overall tracking performance of the extended selection framework, we benchmarked multiple trackers, as we will see in the experiments section 5.3.2, and chose to use three KCF-based trackers (Kernelized Correlation Filter) [Hen+12; Hen+15].

Proposed by Henriques *et al.*, the KCF tracker learns a correlation filter by ridge regression to have a high response for the target object and

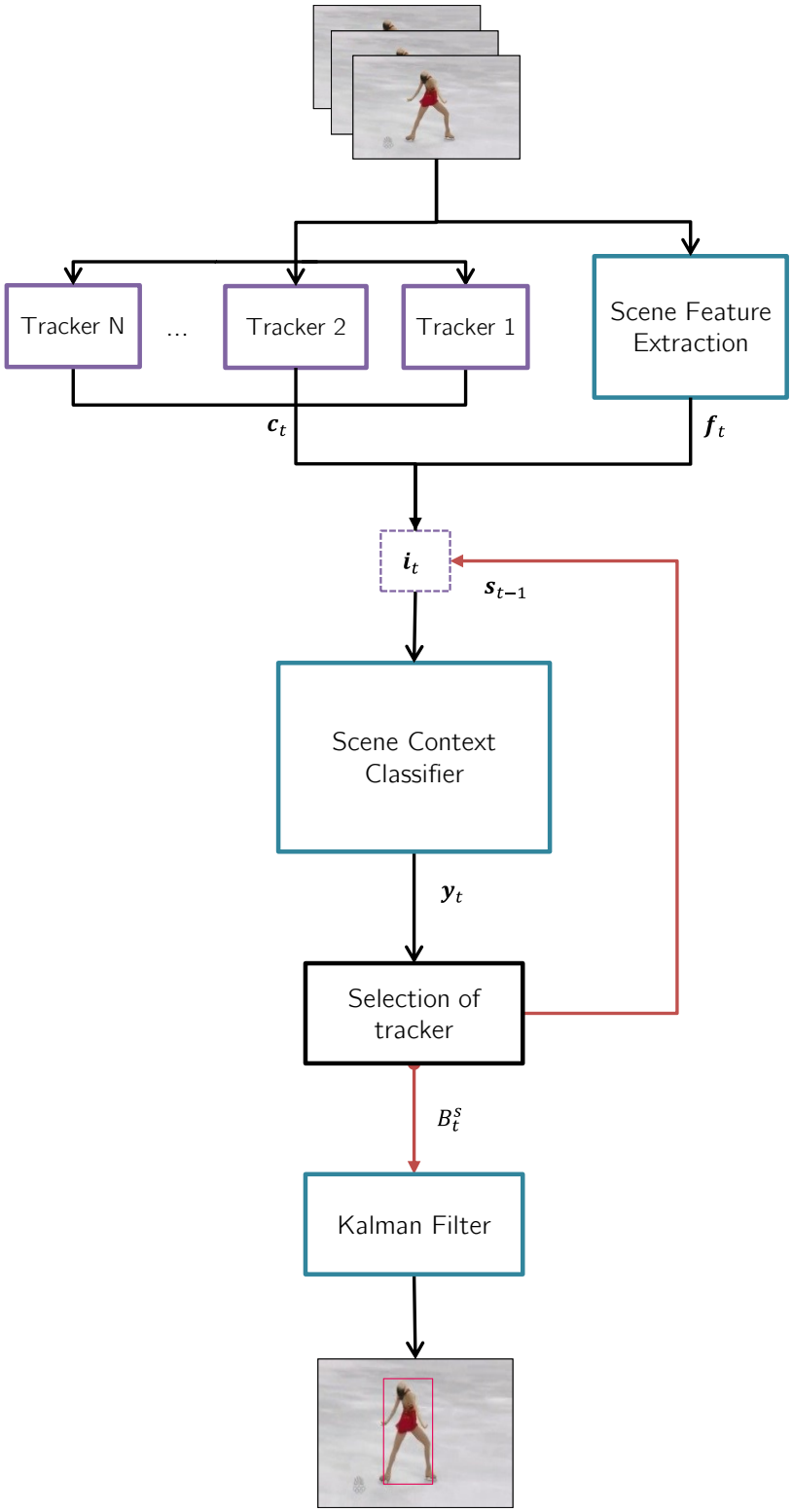


FIGURE 5.1: The overall framework of the extended Scene Context-Based Tracker (SCBT+).

a low response on the background. The correlation is done in the Fourier domain which allows a very efficient implementation. Correlation Filter-based trackers have recently achieved excellent performance, showing great robustness in challenging situations, for example with motion blur and large illumination changes. In addition to their robustness, KCF is computationally efficient allowing us to combine three trackers in our framework while preserving real-time tracking. In fact, to provide some memory, the model is integrated by linearly interpolating the new parameters with the ones from the previous frame.

The KCF tracker follows the pipeline presented in Algorithm 2. For the matter of clarity, the following equations are presented with one-dimensional vectors with a single feature (*i.e.* the pixel value of an image) allowing a simpler notation. However, they are readily transferable to the case of 2D images with multiple channels.

First, a window of fixed size (*i.e.* ROI) is cropped from the input image $\mathbf{z}$, at the estimated target location. The target is located by evaluating equation 5.1 and finding the maximum response. For the single input $\mathbf{z}$, the classifier response is evaluated using circulant structure to compute responses simultaneously instead of using a sliding-window over the sub-windows. Thus, the vector with the responses at all positions is given by:

$$\hat{\mathbf{y}} = \mathcal{F}^{-1}\left(\mathcal{F}(\bar{\mathbf{k}}) \odot \mathcal{F}(\alpha)\right), \tag{5.1}$$

where $\odot$ denotes the element-wise product, while $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote the Fourier transform and its inverse, respectively. The vector $\bar{\mathbf{k}}$ is the vector with elements $\bar{k}_i = \kappa(\mathbf{z}, P^i\mathbf{x})$, κ being the Gaussian kernel. P is the permutation matrix that cyclically shifts $\mathbf{x}$ by one element, P^i corresponding to i permutations resulting in i cyclic shifts. The training samples $\mathbf{x}_i = P^i\mathbf{x}$ are collected through the dense sampling of a given image $\mathbf{x}$.

The vector α, containing the α_i coefficients, is the simple closed form solution to the Regularised Least Squares (RLS) with Kernels (KRLS). And by applying the propriety that the matrix K with elements $K_{ij} = \kappa(P^i\mathbf{x}, P^i\mathbf{x})$ is circulant, it allows the creation of efficient learning algorithm applied to KLRS. Thus to train a new model (α and $\mathbf{x}$), Equation 5.2 is used.

$$\alpha = \mathcal{F}^{-1}\left(\frac{\mathcal{F}(y)}{\mathcal{F}(k) + \lambda}\right), \tag{5.2}$$

where the division is performed element-wise and the λ parameter controls the amount of regularisation. This closed-form solution is very efficient as it uses only Fast Fourier Transform (FFT) and element-wise operations.

As a confidence measure, we chose to use the peak value of the Gaussian filter function, *i.e.* the coefficient of the highest response in equation 5.1.

Low-level Features

We employed three single-scale KCF trackers operating on the same feature domains as the previous OAB trackers: Raw features, HOG features and LAB features.

The KCF RAW tracker uses raw grey-scale pixels as features, thus making the filter single-channel. No feature extraction is performed, other than

Algorithm 2 Matlab code for KCF tracker [Hen+12].

Input: x: training image patch, $m \times n \times c$
Input: y: regression target, Gaussian shaped, $m \times n$
Input: z: test image patch $m \times n \times c$

Output: *responses*: detection score for each location, $m \times n$

```
function TRAIN( x, y, sigma, lambda )                    ▷ Eq. 5.2
    k = kernel-correlation( x, x, sigma );
    alphaf = fft2 ( y ) ./ ( fft2( k ) + lambda );
    return alphaf
end function

function DETECT( alphaf, x, z, sigma )                   ▷ Eq. 5.1
    k = kernel-correlation( x, z, sigma );
    responses = real( ifft2( alphaf .* fft2( k ) ) );
    return responses
end function

function KERNEL-CORRELATION( x1, x2, sigma )
    c = ifft2( sum( conj( fft2( x1 ) ).* fft2( x2, 3 ) );
    d = x1( : )' * x1( : ) + x2( : )' * x2( : ) - 2 * c;
    k = exp( -1 / sigma ∧ 2 * abs( d ) / numel( x1 ) );
    return k
end function
```

a cosine window on the raw pixel values in equation 5.3. The tracker can operate directly on the pixel values, with no feature extraction. However, since the Fourier transform is periodic, it does not respect the image boundaries. The large discontinuity between opposite edges of a non-periodic image will result in a noisy Fourier representation. A common solution is to weight the original $n \times n$ image (x^{raw}) with a cosine (or sine) window:

$$x_{ij} = (x_{ij}^{raw} - 0.5)\sin(\pi i/n)\sin(\pi j/n), \quad \forall i, j = 0, ..., n-1. \tag{5.3}$$

Values near the borders are set to zero, eliminating discontinuities.

The tracker KCF HOG (named Dual Correlation Filter DCF in [Hen+15]) employs Histogram of Gradient (HOG) features from [Fel+10], instead of raw pixels. Via a linear kernel, a linear multi-channel filter is employed. Multiple channels are allowed (such as the orientation bins of a HOG descriptor) by simply summing over them in the Fourier domain. The HOG descriptors consist of 4×4 cells and constitute a feature map with 31-dimensional vectors $G(i, j)$, with 27 dimensions corresponding to different orientation channels (9 contrast insensitive and 18 contrast sensitive) and 4 dimensions capturing the overall gradient energy in square blocks of four cells around (i, j).

The KCF LAB tracker employs LAB features computed by converting the input image into CIE-Lab channels and quantising the colours into 15 centroids. The lab features also consist of 4×4 cells. The LAB, and HOG features, are also weighted by a cosine window, which smoothly removes discontinuities at the image boundaries caused by the cyclic assumption.

5.2.2 Classifier training strategies

As detailed in the previous chapter (*c.f.* section 4.3.3), the scene context classifier makes use of the information extracted from the scene (*i.e.* input vector $\mathbf{i}_t$) in order to predict the performance of a tracker (*i.e.* output vector $\mathbf{y}_t$). To learn the different patterns that show high correlation between the information extracted from the scene context and the performance of a tracker in the particular set of conditions, a fully connected Multi-Layer Perceptron (MLP) with one hidden layer, N output neurons and sigmoid activation functions is employed.

The training of the classifier is executed offline on a dataset with annotated object bounding boxes. Let us consider a training sample $\{\mathbf{i}_j, o_j^*\}$, $(j \in 1..N_{train})$ where N_{train} is the number of training samples, $\mathbf{i}_j$ is the input vector and o_j^* is the *label* for the sample j, described below. In order to construct the classifier input vector $\mathbf{i}_j = \{\mathbf{F}_j, \mathbf{C}_j, \mathbf{S}_{j-1}\}$, the trackers are run on each video, and at each frame (*i.e.* training sample j), we extract the scene context features $\mathbf{F}_j$, the trackers' confidences $\mathbf{C}_j$ and save the identifier of best tracker to be used as the "previously selected tracker" $\mathbf{S}_{j-1}$ in the following frame.

The best tracker for each sample j, *i.e.* the label o_j^*, is determined by computing the F-scores of each tracker. The tracker with the highest F-score is considered as the label o_j^* of the sample.

We optimise the neural network parameters with standard stochastic gradient descent algorithm by minimising the mean squared error between the network's response vector $\mathbf{y}_j = (y_{j,1}..y_{j,N})$ and the desired output vector $\mathbf{y}_j^*$. The usual output strategy (one-of-N encoding) is $\mathbf{y}_j^* \in \{-1, +1\}^N$ with $+1$ for the component corresponding to the sample label o_j^* and -1 otherwise, which was also employed in the SCBT framework. However, in section 5.3.1, we will evaluate different output strategies as well as different network architecture to optimise the training of the classifier.

The network's final class prediction is simply $o_j = \text{argmax}_{n \in N}\, \mathbf{y}_{j,n}$.

5.3 Performance Evaluation

In order to understand the added value of our extended selection framework SCBT+, we proceed in evaluating the performance of each component. First, using the KCF trackers, we will review the training of the context classifier by examining the recognition rate for different inputs and training strategies and compare the results to the previous framework SCBT. Subsequently, we will analyse the impact of the choice of the individual trackers as we perform classifier and tracking performance evaluation on different sets of trackers. Finally, we will evaluate the overall tracking performance of the proposed framework on multiple public benchmarks and compare it to state-of-the-art tracking algorithms.

5.3.1 Scene Context Classifier Evaluation

In order to improve the training and learning of the scene context classifier, we evaluate the prediction performance of the classifier by varying different parameters. Using the KCF trackers previously introduced, we train and

Classifier training dataset	Training rate	Testing rate
Princeton	85.39%	77.79%
Princeton+ILSVRC	**89.56** %	**79.56** %

TABLE 5.1: Correct classification rates for the scene context classifier trained on different datasets and tested on the VOT2013 dataset.

evaluate multiple classifiers using different: training datasets, input vectors and the classifier output strategies.

Training dataset

As opposed to our previous framework SCBT where we only used the Princeton dataset (*c.f.* section 2.5.2) to train the scene context classifier, we augmented the training database to give more diverse training samples to our context classifier. A larger database facilitates the learning and convergence of the neural network. To do so, we combined the Princeton Dataset containing 100 videos, with a sub-set of the ILSVRC2015 Dataset (*c.f.* section 2.5.6). With a total of 397 videos (106203 training samples and 12700 validation samples), our training dataset represents a diverse set of object types, background and scene conditions.

To experimentally show the influence of the training dataset on the learning process, we used KCF trackers and trained neural network classifiers with the same architecture (one hidden layer of 50 neurones) on the two different datasets: one on the Princeton Dataset and one on the new augmented Princeton+ILSVRC15 dataset as shown in table 5.1. The classifiers are evaluated on the VOT2013 dataset.

The classification rates on the training dataset as well as the testing dataset have increased, as expected, when using the augmented training dataset. Thus, for the remaining experiments, all the classifiers were trained on the Princeton+ILSVRC15 dataset with KCF trackers.

Classifier input

In section 4.3.3, we described the different components of the scene context classifier's input vector $\mathbf{i}_t$. In order to understand the contribution of each part of this vector, we trained multiple classifiers using different inputs. Moreover, to be able to compare the KCF-based classifier with the OAB-based classifiers from the SCBT framework (*c.f.* section 4.4.1), we trained the classifiers using the same network architecture.

The results are presented in table 5.2, with classification rates from the different classifiers evaluated on the VOT2013 test dataset.

We first separate the scene features into the three categories presented in section 4.3.2: *local* $\mathbf{f}_t^L$, *global* $\mathbf{f}_t^G$ and *differential* $\mathbf{f}_t^D$ scene features. Then, we add the 3 frames time window $\mathbf{F}_t$, the confidence values $\mathbf{C}_t$ and the previous tracker $\mathbf{S}_{t-1}$.

We can observe that KCF-based classifiers globally behave the same way as the OAB-based classifiers. As already observed in chapter 4, the scene features alone are not sufficient to make a prediction as the recognition rate

Classifier input $\mathbf{i}_t$	Test rate (OAB)	Test rate (KCF)
$\{\mathbf{f}_t^L\}$	26.28 %	21.19 %
$\{\mathbf{f}_t^L, \mathbf{f}_t^G\}$	27.25 %	36.45 %
$\{\mathbf{f}_t^L, \mathbf{f}_t^G, \mathbf{f}_t^D\}$	30.03 %	31.73 %
$\{\mathbf{F}_t\}$	35.80 %	35.03 %
$\{\mathbf{F}_t, \mathbf{C}_t\}$	40.06 %	41.91 %
$\{\mathbf{F}_t, \mathbf{C}_t, \mathbf{S}_{t-1}\}$	81.80 %	**82.77** %

TABLE 5.2: Correct classification rates for different classifiers using different set of inputs, tested on the VOT2013 database. We compare the classifiers trained using OAB trackers and KCF trackers.

Classifier output $\mathbf{y}_t^*$	Train rate	Test rate	Accuracy	Failures
One-of-N	89.56 %	79.56 %	0.583	1.313
Threshold	63.77 %	51.76 %	0,600	1,440
Ranking	88.95 %	77.82 %	**0.607**	**0.563**
Regression	69.80 %	62.24 %	0,590	1,130

TABLE 5.3: Correct classification rates on train (Princeton+ILSVRC15) and test (VOT2013) datasets for context classifiers trained using different output $\mathbf{y}_t^*$ strategies and the corresponding VOT2013 benchmark tracking results.

for the combination of local, global and differential features obtain only 31.73%. Using different time steps helps increase the recognition rate to 35.03%. And it is the combination of scene features $\mathbf{F}_t$, confidence values $\mathbf{C}_t$ and the previously selected tracker $\mathbf{S}_{t-1}$ that shows a high prediction rate of 82.77%.

Moreover, the prediction rates are generally increased when using the KCF trackers rather than the OAB trackers. In fact, with better performing and more "stable" trackers, the scene context classifier training improves.

Training procedure

As discussed in section 5.2.2, the classifier's training relies on the error measured between the desired output vector $\mathbf{y}_t^*$ and the actual networks response vector $\mathbf{y}_t$. The output vector $\mathbf{y}_t^*$ is computed based on the F-scores of the trackers (Eq.3.11). We experimented with different ways of computing $\mathbf{y}_t^*$:

- **One-of-N**: $\mathbf{y}_t^* \in \{-1, +1\}^N$ It is the usual output strategy in classification where $+1$ is assigned to the class label o_t^* and -1 otherwise.
- **Threshold**: $\mathbf{y}_t^* \in \{-1, 0, +1\}^N$ We threshold the F-score of each class and assign $+1$ if the F-score is higher than the threshold, 0 if it is lower, and -1 if the F-score is null (*i.e.* the tracker is completely lost). Here, the threshold value was set to 0.6, determined empirically.
- **Ranking**: $\mathbf{y}_t^* \in \{-1, -0.3, +0.3, +1\}^N$ We rank the F-scores of our classes, and assign respectively $+1$, $+0.3$ and -0.3 from highest to

Tracking Algorithm	Intensity	Texture	Colour	Motion
OAB	HAAR	HOG	HOC	-
STK	HAAR	HOG	HOC	-
PT	-	HOG + HOC		-
KCF	RAW	HOG	LAB	-
KLT	-	-	-	OF

TABLE 5.4: Summary of individual trackers benchmarked with corresponding features.

lowest F-score. If the F-score value is 0, than -1 is assigned to the corresponding class.

- **Regression**: $\mathbf{y}_t^* = \{Fscore_t^1, .., Fscore_t^N\}$ We directly use the F-score values, so the classifier trains to predict these values.

We trained multiple classifiers using the different output strategies, with the same architecture (one hidden layer of 50 neurones), then evaluated the tracking performance of each classifier (*i.e.* the selection framework uses only the classifier's output to select a tracker) on the VOT2013 benchmark as shown in Tab.5.3.

The *Regression* and *Threshold* strategies have the lowest classification rates. In fact, it is a difficult task to learn and predict the F-scores of the trackers explaining the relatively low rate of *Regression*. *Thresholding* the F-scores also proves to be inefficient as we lose the information of the best tracker and can have several labels for the same sample.
On the other hand, *One-of-N* encoding and *Ranking* give the best classification results. With *One-of-N* encoding, we consider that only one tracker is correct at a point of time. However in *Ranking*, the classifier predicts the order from the best expected tracker to the least. This approach can be considered a mix of *One-of-N* classification and *Regression* as we still keep the information of the best tracker at a each frame, but the remaining trackers are ranked instead of considered as 'failed'. The *Ranking* approach proved its efficiency and superiority in the tracking results as it gives the best accuracy and robustness.
The scene context classifier used in the remaining experiments is therefore trained on the augmented dataset, with the input $\mathbf{i}_t = \{\mathbf{F}_t, \mathbf{C}_t, \mathbf{S}_{t-1}\}$ and *Ranking* output strategy.

5.3.2 Individual Trackers Performance Analysis

Before choosing the KCF trackers to be used as our individual trackers $\{T_n\}, (n \in 1..N)$ in the extended framework SCBT+, we reviewed multiple tracking algorithms using the same domain features as the OAB trackers employed in the SCBT framework.

Reviewed trackers

From the state-of-the-art, we gathered a few trackers with available open-source code and that respect our criteria: online and low-level feature-based trackers. From the original tracking algorithms, we extended some of them

to other feature domains as summarised in Table 5.4. The benchmarked trackers are the following:

- **Online AdaBoost (OAB)**: originally introduced with Haar-like features [GGB06], it is a feature selection with boosting based tracker. We extended the method to HOG (Histogram Of Gradient) features and HOC (Histogram Of Colour) features using the HSV colour channels (*c.f.* section 3.3.2).
- **Struck (STK)**: it is a Support Vector Machine (SVM) based tracker [HSHT11] with a Kernelized structured output SVM learned on Haar or HOG features. We extended it to HOC features the same way as the OAB algorithm.
- **PixelTrack (PT)**: Fast tracker using generalised Hough transform on pixel-based descriptors and a probabilistic segmentation [DG13]. Uses a mix of HOG and Colour features. Its extention PixelTrack+ (PTp) was also evaluated.
- **Kernelized Correlation Filter (KCF)**: tracker proposed by [Hen+15] based on a Kernalized correlation filter with RAW (raw intensity pixels), HOG and LAB (raw pixel values from LAB colour space) features (*c.f.* section 5.2.1).
- **Kanade-Lucas Tracker (KLT)**: the KLT (Kanade-Lucas-Tomasi) tracker [ST94] iteratively computes the translation of the target object region its new location in the following frame using optical flow.

Apart from the KLT tracker which is feature-point based, the remaining trackers are tracking-by-detection algorithms with easily implementable low-level features, which motivates our choice of benchmarking these trackers. And in order to represent the motion domain in our pool of trackers, we added the KLT tracker to the benchmark. In fact, since it is not our goal to implement new individual trackers, but rather use accessible state-of-the-art online trackers, we did not extend the Struck, KCF and OAB trackers to the motion model as it is a difficult task to implement motion features in tracking-by-detection algorithms.

Benchmark performance analysis

We run each of the trackers presented in table 5.4 on the latest VOT2015 benchmark (*c.f.* section 2.5.5) in the aim of evaluating, comparing and choosing which trackers to use in the SCBT+ framework. Table 5.5 presents the rankings of each tracker regarding the *accuracy* measure and *robustness* measure.

The Struck and KCF trackers show globally the best results among the different trackers, as KCF HOG, KCF LAB, STK HOC and STK HAAR rank among the three best overall either in accuracy or in robustness. On the other hand, the OAB trackers rank last among the tracking-by-detection algorithms. The motion-based KLT tracker also performs only averagely. We can also observe a strong tendency for colour-based trackers to outperform the rest of the feature domains. However, using only a a colour-based tracker is insufficient and the goal of our selection framework is to surpass the performance of the individual trackers used in it.

Tracker	Accuracy	Robustness
OAB HAAR	5.92	6.43
OAB HOG	6.20	7.45
OAB HOC	5.55	5.30
STK HAAR	4.98	4.03
STK HOG	4.90	4.48
STK HOC	3.82	3.07
KCF RAW	5.20	4.52
KCF HOG	3.00	3.67
KCF LAB	4.18	5.17
PT	6.17	6.58
PTp	5.87	5.32
KLT	6.35	5.47

TABLE 5.5: Performance of the different trackers benchmarked on VOT2015. The values presented for the accuracy and Robustness measures are in form of weighted ranks. For each measure, the first second and third tracker in ranking is coloured in red, blue and green respectively.

Classifier based on different sets of trackers

Subsequently to the review of the individual trackers, we trained different scene context classifiers and evaluated the tracking performance of the SCBT+ framework using different "sets" of trackers. We want to study the influence of the choice of the individual trackers on the selection framework. Thus we define the following sets:

$$\begin{array}{lllll} \text{Set A} & = \{\text{STK Haar}, & \text{KCF HOG}, & \text{STK HOC}, & \text{KLT}\} \\ \text{Set B} & = \{\text{KCF RAW}, & \text{KCF HOG}, & \text{KCF HOC}, & \text{KLT}\} \\ \text{Set C} & = \{\text{STK Haar}, & \text{KCF HOG} & \text{STK HOC} & \} \\ \text{Set D} & = \{\text{KCF RAW}, & \text{KCF HOG} & \text{KCF HOC} & \} \end{array}$$

In Set A, we chose the best performing tracker per visual cue based on the results from the VOT2015 benchmark presented above, constituting a 4 trackers set. On the other hand, for Set B, we chose the same tracking-by-detection algorithm for the different visual cues. Here we employed KCF for their performance and computational efficiency compared to the Struck trackers. Thus constituting a trackers set with 3 KCF trackers and KLT to represent the motion domain. For both of those sets, because of the average performance of the KLT tracker, we also evaluated the sets without the KLT tracker giving Set C and Set D.

For each of these sets of individual trackers, we trained a scene context classifier, and evaluated its tracking performance (*i.e.* the selection framework using only the classifier's output to select a tracker) on the VOT2013 benchmark, presented in table 5.6. We also show the individual trackers performance for each set on the VOT2013 benchmark in table 5.7

We can first observe that classifiers based on Set A and Set B have the lowest tracking results in table 5.6. Moreover, these sets show higher failures rates than theirs respective individual trackers. Set A achieves a failure rate of 1.150, while the most robust tracker from its pool achieves 0.313. The same occurs for set C having 0.938 failures while its most robust tracker achieves 0.875. The set B also shows the same behaviour.

Trackers Set	Train rate	Test rate	Accuracy	Failures
Set A	87.97 %	84.84 %	0.540	1.150
Set B	86.43 %	74.26 %	0.573	1.313
Set C	89.33 %	83.83 %	0.597	0.938
Set D	88.95 %	77.82 %	**0.607**	**0.563**

TABLE 5.6: Correct classification rates on train (Princeton+ILSVRC15) and test (VOT2013) datasets for context classifiers trained using sets of trackers and the corresponding VOT2013 benchmark tracking results.

Set	Individual Trackers	Accuracy	Failures
Set A	STK Haar	0.580	1.250
	KCF HOG	0.590	0.875
	STK HOC	0.617	0.313
	KLT	0.489	2.625
Set B	KCF RAW	0.522	1.688
	KCF HOG	0.590	0.875
	KCF HOC	0.568	0.938
	KLT	0.489	2.625

TABLE 5.7: Tracking performance of individual trackers from the different sets, on the VOT2013 benchmark.

For these cases A, B and C, while the classifiers show high rates of training and prediction rates, the overall tracking performance of the classifier is worse than the performance of its individual trackers. Meaning that the selection framework based on this classifier fails to make the proper selection.

This shows that the training and testing rates of the classifier do not necessarily reflect the tracking performance. In fact the testing rate shows at what rate a classifier is able to predict the correct tracker over the dataset, taking each frame as a separate example. Thus for equal testing rates, two classifiers can have different tracking performances, as the frames where the classifiers fail can be different. In some cases, while failing in predicting the best tracker for a frame, the "falsely" predicted tracker can still be performing well. Hence, it is important to evaluate both the classifier's testing rate and tracking performance.

We can first explain the failure of sets A, B, and C because of the uneven performance of the individual trackers. In both sets A and B, KLT's failures are too high in comparison to the rest of the trackers from the set. The same applies for set C where the STK HOC outperforms by a large margin the rest of the trackers. The unevenness of performance of the individual trackers in a set can threaten the balance between the trackers. While mis-classifications can be managed when the trackers perform equally, in the case of unbalance trackers, prediction errors from the classifier would considerably lower the tracking performance of the minimum performance to achieve. If we take the example of set A, with the tracker STK HOC having a rate of 0.313

Method	Accuracy	Failures
KCF RAW	0.522	1.688
KCF HOG	0.590	0.875
KCF LAB	0.568	0.938
Context classifier	**0.607**	0.563
Context classifier + gen.Update	0.606	0.438
Context classifier + gen.Update + Kalman F. (SCBT+)	0.599	**0.375**
Previous work (STC)	0.562	0.792
Previous work (SCBT)	0.553	0.583

TABLE 5.8: VOT2013 Benchmark[Kri+13] accuracy and failure results for: the KCF trackers, the extended selection framework SCBT+ and previous frameworks STC (Chapter 3) and SCBT (Chapter 4).

failures, in order to be relevant the classifier need to achieve a lower failures rate. However, with the tracker KLT having a failures rate of 2.625, the mis-classifications have a higher probability of selecting a failed tracker.

On the other hand, when using the same tracking algorithm on different domains in set D, the classifier succeeds in outperforming its individual trackers, motivating our choice of using KCF trackers.

We can conclude that the choice of the individual trackers in this scene context-based selection framework is important and crucial.

5.3.3 Tracking Evaluation

To evaluate our proposed selection framework, we used three editions of tracking benchmarks VOT (*c.f.* section 2.5,). The VOT benchmarks provide the visual tracking community with a precisely defined and repeatable way of comparing short-term trackers. These benchmarks allow us to compare our proposed frameworks with state-of-the-art tracking algorithms.

VOT2013 benchmark

We first used the VOT2013 benchmark to evaluate our selection framework SCBT+ and its different components, compare it to the individual trackers used (KCF) and the frameworks SCBT and STC from the previous chapters. The results are presented in Table 5.8.

The context classifier demonstrates its efficiency as it reduces the failures of the KCF trackers and improves the general accuracy. The *general update* of the trackers and Kalman filter both helps greatly increase the robustness of the tracking with a slight, negligible decrease of accuracy. Note that it is common to lose accuracy when gaining robustness in the VOT benchmarks as, at each failure from the framework, it is reinitialised with the ground truth.

Moreover, we can compare the extended framework SCBT+ to the SCBT selection frameworks based on OAB tracker, and see that the optimisation of the scene context classifier's training as well as the change in tracker greatly improves the overall tracking robustness.

VOT2014 benchmark

In figure 5.2 we compare the proposed selection framework SCBT+ to state-of-the-art trackers that participated in the VOT2014 challenge [Kri+14].

We can observe that our extented selection framework SCBT+ ranks among the top in both accuracy and robustness, outperforming methods such as EDFT[SL12], FoT[VM11] or Struck[HSHT11], proving that using different tracker and a simple scene context classifier greatly improves the accuracy and robustness of the overall tracking. We also outperform HMMTxD[VMN16] in terms of robustness.

However a few tracking methods such as DSST[Dan+14a] or PLT[Kri+13] outperform our selection framework.

VOT2015 benchmark

We further evaluate our selection framework on the VOT2015 benchmark[Kri+15].. In figure 5.3 we can see that our framework SCBT+ performs on the same level as most state-of-the-art methods but does not outperform the top ones. The winning methods, such as MDNet[NH15] or DeepSRDCF[Dan+15] which are based on Convolutional Neural Networks, are complex and computationally expensive compared to our framework incorporating relatively simple individual trackers and an MLP with one hidden layer. Moreover, the difficulty of the VOT2015 dataset and the single scale nature of the KCF trackers limits the selection framework. However, we can see a big improvement from SCBT to SCBT+ ranks in the benchmark.

5.4 Conclusion

In this chapter, we proposed to extend our previous selection framework, which exploits scene context information in order to learn and predict the most suitable tracker. Using state-of-the-art KCF trackers, we optimised the training of the scene context classifier by exploring multiple output strategies to select the most adapted one to our framework. With a few improvement in the SCBT framework, the SCBT+ proved to considerably increase the robustness of the overall tracking performance.

We further evaluated the proposed framework on multiple benchmarks proving the efficiency of the scene features and scene context classifier and showing state-of-the-art tracking performance. However, on the VOT2015 benchmark, the selection framework showed its limits compared to the state-of-the-art methods. In fact, the tracking performance of the framework is limited by the performance of the individual trackers that are used, as well as the performance of the scene context classifier.

The proposed selection framework could be further improved by using more complex individual tracker at the expense of the computational efficiency and speed of the framework. Employing multi-scale adaptation to handle the possible size variations of the target object would also increase the overall accuracy. Additionally, providing new 'semantic' scene features to the classifier that would characterise the type of object or type of scene would be an interesting future research direction. Another direction would be to learn how to extract automatically the scene features needed by the

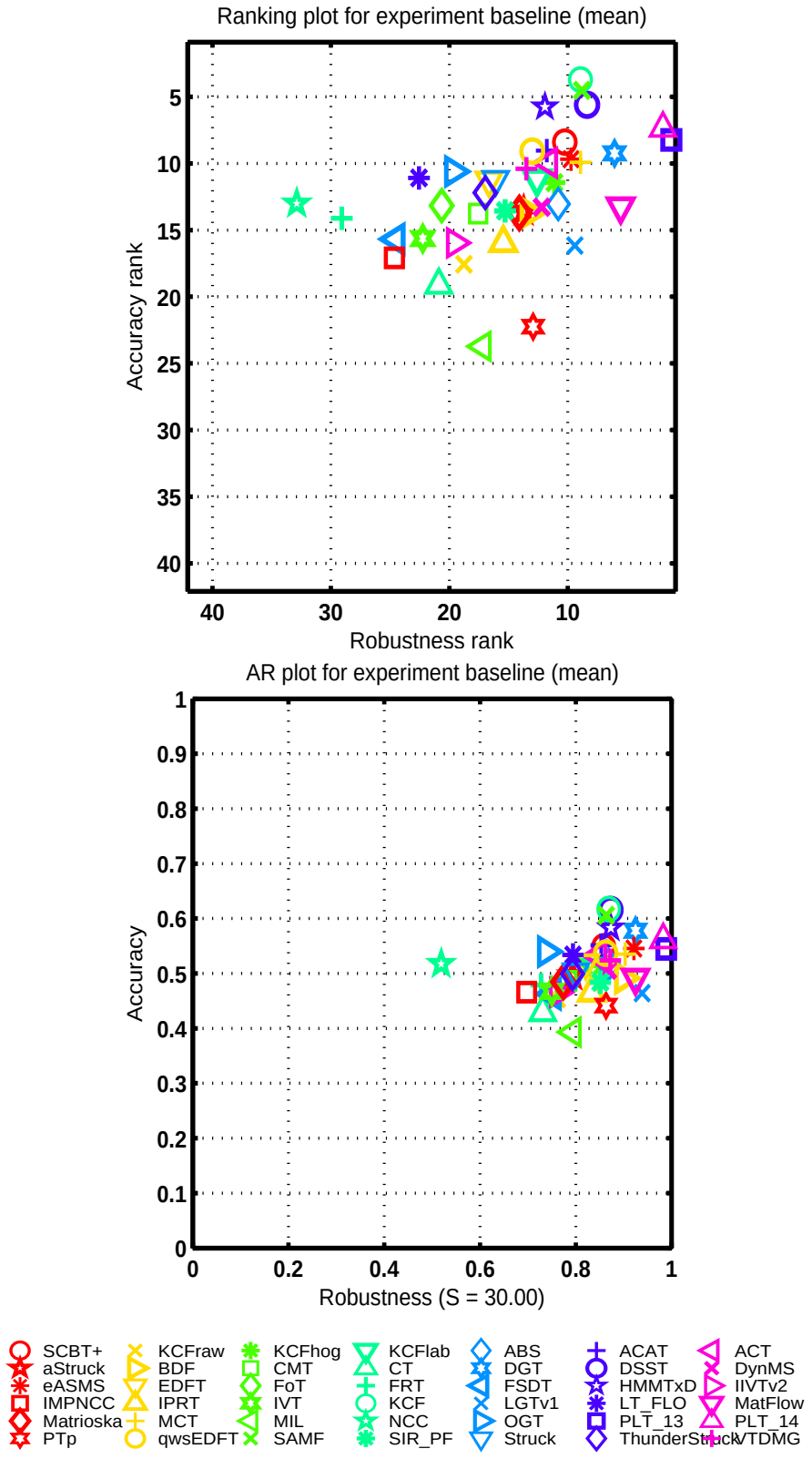


FIGURE 5.2: VOT2014 Benchmark [Kri+14] ranking (1^{st}) and Accuracy-Robustness (2^{nd}) plots. Comparison of our proposed framework (SCBT+), the individual tracker used KCFraw, KCFhog, KCFhoc and state-of-the-art trackers from the VOT2014 Challenge: ABS[Kri+13], ACAT[Kri+14], ACT[Dan+14b], aStruck[Kri+14], BDF[MP14], CMT[NP14], CT[ZZY12], DGT[Cai+13], DSST[Dan+14a], DynMS[CM02], eASMS[Kri+13], EDFT[SL12], FoT[VM11], FRT[ARS06], FSDT[Kri+14], HMMTxD[VMN16], IIVTv2[Kri+14], IMP-NCC[Kri+14], IPRT[Kri+14], IVT[Ros+08], KCF[Hen+15], LGTv1[CKL13], LT-FLO[Leb+13], MatFlow[Kri+14], Matrioska[MP13], MCT[DG14], MIL[BYB11], NCC[BH01], OGT[NHH14], PLT-13 [Kri+13], PLT-14[Kri+13], PT+[DG13], qwsEDFT[Kri+14], SAMF[Kri+14], SIR-PF[Kri+14], Struck[HSHT11], Thunder-Struck[Kri+14], VTDMG[Yi+12].

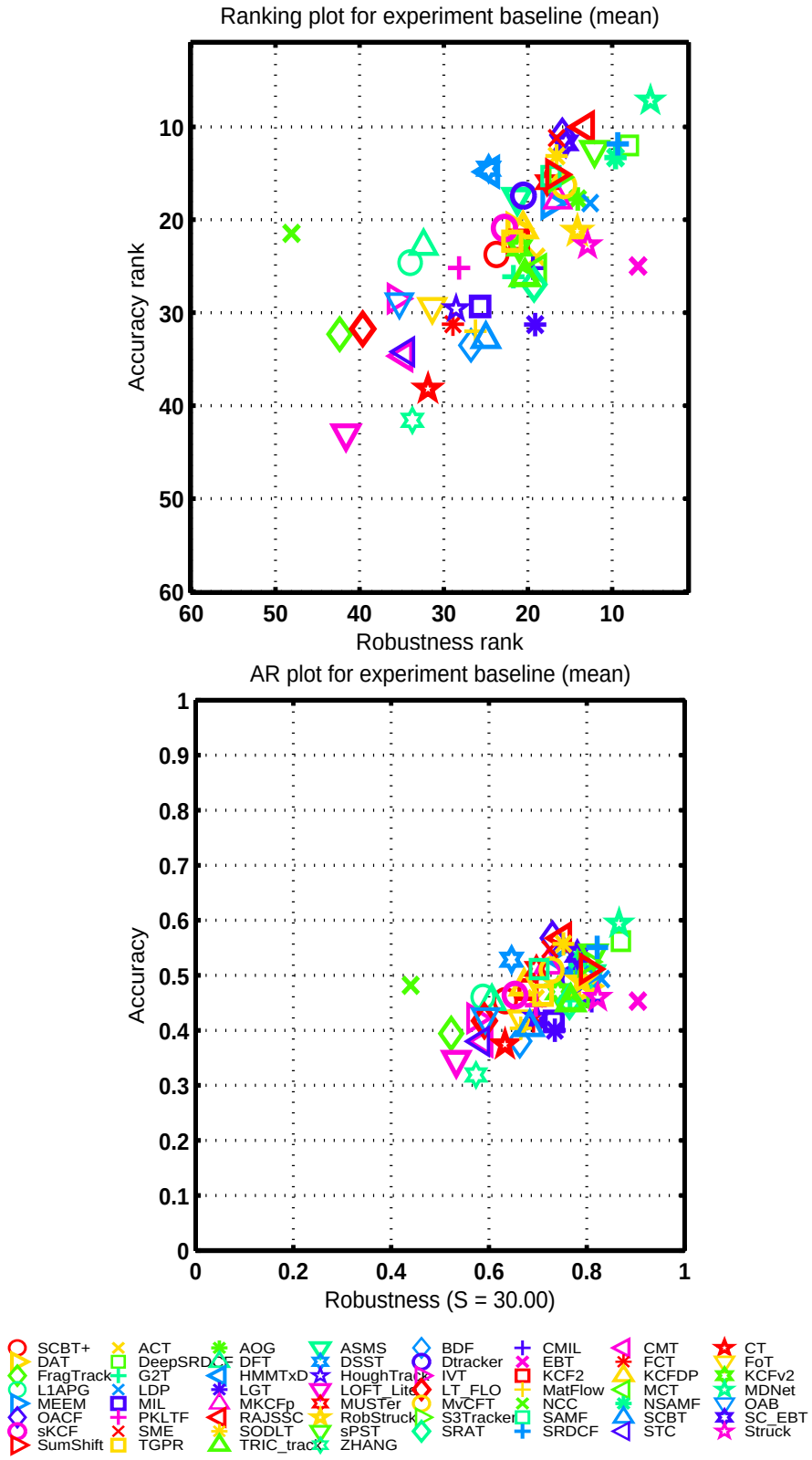


FIGURE 5.3: VOT2015 Benchmark [Kri+15] ranking (1^{st}) and Accuracy-Robustness (2^{nd}) plots. Comparison of the extended SCBT+, previous SCBT framework and state-of-the-art trackers from the VOT2015 Challenge: ACT[Dan+14b], AOG[Kri+15], ASMS[VNM14], BDF[MP14], CMIL[Kri+15], CMT[NP14], CT[ZZY12], DAT[PMB15], DeepSRDCF[Dan+15], DFT[Kri+15], DSST[Dan+14a], Dtracker[Kri+15], EBT[ZPL15], FCT[Kri+15], FoT[VM11], FragTrack[Kri+15], G2T[Kri+15], HMMTxD[VMN16], HoughTrack[GRB13], IVT[Ros+08], KCF2[Kri+15], KCFDP[Kri+15], KCFv2[Kri+15], L1APG[Bao+12], LDP[Kri+15], LGT[CKL13], LOFT-Lite[Kri+15], LT-FLO[Leb+13], MatFlow[Kri+15], MCT[DG14], MDNet[NH15], MEEM[ZMS14], MIL[BYB11], MKCFp[Kri+15], MUSTer[Hon+15] , MvCFT [Kri+15], NCC[DH01], NSAMF [Kri+15], OAB[GGB06], OACF[Kri+15], PKLTF[Kri+15], RAJSSC[Kri+15], RobStruck[Kri+15], S3Tracker[Kri+15], SAMF[Kri+14], SCBT[MDB15a], SC-EBT[WY14], sKCF[Kri+15], SME[Kri+15],

classifier. In fact, instead of manually designing the scene features, a Convolutional Neural Network can be employed to combine the tasks of scene feature extraction and tracker prediction.

Chapter 6

Deep Learning-Based Tracker Selection

6.1 Introduction

In the past few years, deep learning and more specifically convolutional neural network (CNN) models became quite popular in the computer vision community and have been used successfully, giving state-of-the-art results for many different tasks, including image classification [KSH12] and speech recognition [Hin+12].

Image representation is essential in vision domains. In fact, in the case of tracking, many methods use existing image features (such as Haar features, histogram features, or local binary patterns) to represent the target objects, while concentrating on improving the classification task. With features that are hand-crafted offline, the tracking model may not be adapted to the tracking problem or to the tracked object. The introduction of CNNs and deep architectures can solve this problem as these networks learn richer invariant features via multiple non-linear transformations. As opposed to using hand-crafted features, it has been demonstrated that the features learned by a large-scale CNN can achieve far superior performance in some high-level vision tasks.

However, the immediate adoption of CNN for online visual tracking is not straightforward. First of all, CNN requires a large number of training samples, which is often not available in visual tracking. Moreover, CNN tends to easily over-fit to the most recent observation, which may result in a drifting problem. Moreover, CNN training is computationally intensive for online visual tracking. Due to these difficulties, CNN has been employed only as an offline feature learning step on predefined datasets for tracking applications so far.

In the context of a tracker selection framework, one interesting idea would be to use a CNN to both extract global context features from the scene and learn to predict the correct tracker to use at each video frame. Thus, in this chapter, we explore the possibility of using a convolutional network to execute and learn both of these tasks in a single network.

6.2 State-of-the-art of Deep Learning in Tracking

Deep learning, and more specifically Convolutional Neural Networks (CNN), attracted a lot of attention in the computer vision community in the recent years due to its performance and results for different tasks. We can cite the works by Krizhevsky *et al.* for object classification [KSH12], Girshick *et al.*

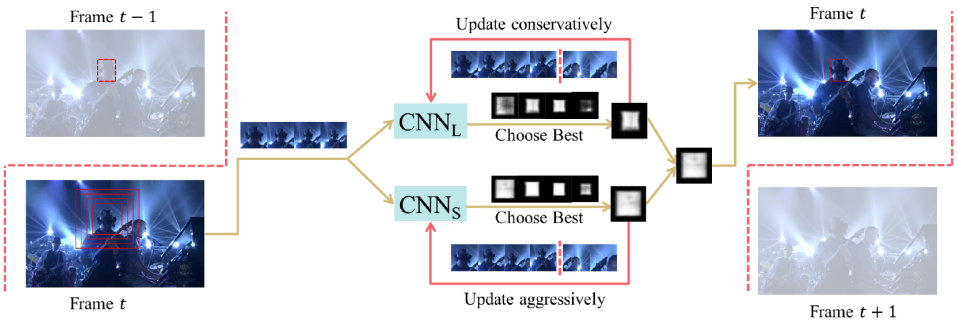


FIGURE 6.1: Pipeline of the SO-DLT framework. Image courtesy of [Wan+15b].

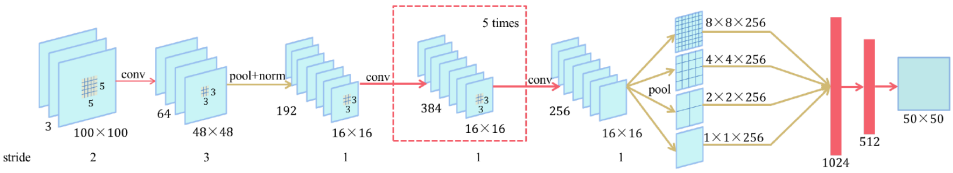


FIGURE 6.2: Architecture of the structured output CNN used in the SO-DLT framework. Image courtesy of [Wan+15b].

for object detection [Gir+14], Taigman *et al.* for face verification [Tai+14], and Garcia and Delakis for face detection [GD04]. CNNs can be used in different ways when applied to the task of visual tracking. Pre-trained networks for object recognition can provide features for visual representation of the object to be tracked. Another possibility is training from scratch or fine-tuning a pre-trained network in an online manner in order to better adapt it to the appearance of the target and the scene.

Inspired by the advances in deep learning architectures, Wang and Yeung [WY13] introduced the first deep learning tracker reported in literature, the Deep Learning Tracker (DLT) in which a Stacked Denoising Auto-Encoder (SDAE), in the role of unsupervised image feature extractor, is trained offline to learn generic image features from a large image dataset as auxiliary data. Then, the features learned are transferred to the online tracking process. Tracking is performed with a classification neural network constructed with the trained auto-encoder and an additional classification layer. Moreover, for more adaptivity, both the feature extractor and the classifier can be further tuned to adapt to appearance changes of the moving object. This approach, however, is limited by the auto-encoder which is not suitable for the tracking task as reconstructing the full image (rather than the object) is not optimal.

Following the success of Wang and Yeung's work [WY13], Zhou *et al.* [Zho+14b] proposed to incorporate a deep learning architecture with an online Ada-Boost framework. An offline unsupervised generic image feature extractor is built using an SDAE to learn multi-level feature descriptors. Each layer of the SDAE serves as a different level of feature space, which is subsequently transformed to a discriminative deep neural network classifier by adding an additional classification layer. Finally, in an boosting feature selection framework, these layered classifiers are combined for to localise the target.

To address the limitations of the DLT framework [WY13], Wang *et al.*

[Wan+15b] later proposed the a CNN-based Structured Output Deep Learning tracker (SO-DLT). In this framework, the feature extraction is performed by a pre-trained CNN with the goal of distinguishing the objects from non-objects, *i.e.* it learn from examples the concept of *objectness*, instead of reconstructing the input or performing categorical classification. The CNN parameters learned offline are not fixed as they are fine-tuned during online tracking in order to adapt the CNN to the target object. The CNN, presented in figure 6.2, outputs a pixel-wise probability map, instead of a simple class label, indicating the probability that each pixel belongs to the object. Moreover, two CNNs are run concurrently during online tracking with different fine-tuning strategies as to make one CNN account for short-term appearance changes while the other accounts for long-term changes. The CNNs work collaboratively as illustrated in figure 6.1. The final tracking result is determined by the CNN with the higher confidence.

Rather than training a CNN offline, Li *et al.* [LLP14b] introduced an online CNN, *i.e.* updated online and only trained with the first frame at the initialization, for learning effective feature representations of the target object in an online tracking-by-detection strategy. The three-layer CNN, built with three local-contrast normalized images and a gradient image as cues, employs a truncated structural loss function. This loss function helps obtain and maintains a large number of training samples and reduces the risk of tracking error accumulation by accommodating the uncertainty of the model output. The Stochastic Gradient Descent approach is also enhanced in CNN training with a temporal selection mechanism, which generates positive and negative samples within different time periods and increases the speed of the training process. The CNN model is updated in a "lazy" style to speed-up the training stage, where the network is updated only when a significant appearance change occurs on the object, without sacrificing tracking accuracy.

Li *et al.* [LLP14a] further proposed to use multiple online CNNs, instead of one, as a data-drive model of different instances of the target object, in a tracking framework named DeepTrack. Through the use of a pool of CNNs, the authors aimed to capture and adapt to object changes without overfitting to data. Each CNN maintains a specific set of kernels discriminating the object from its surrounding using the same previously proposed low-level cues. Given a frame, a subset of CNNs from the pool are selected to evaluate patches from the ROI. The best matching CNN is determined using the k-Nearest Neighbours (k-NN) and the matching score is assigned to the corresponding patch. The patch with the highest score is selected as the tracking result for the frame and the corresponding CNN is retrained using a warm-start back-propagation scheme. Finally, an iterative training procedure is applied to each cue independently, and then the fully-connected layers are jointly trained.

More recently, Li *et al.* [LLP16] improved the DeepTrack framework [LLP14a], illustrated in figure 6.3, by introducing a more complex CNN model with less image cues (*c.f.* figure 6.4) instead of using the pool of CNNs. The four-layer CNN model generates scores for all possible hypotheses of the object locations (object states) in a given frame. The hypothesis with the highest score is then selected as the prediction of the object state in the current frame.

Instead of treating CNN as a black-box feature extractor, Wang *et al.*

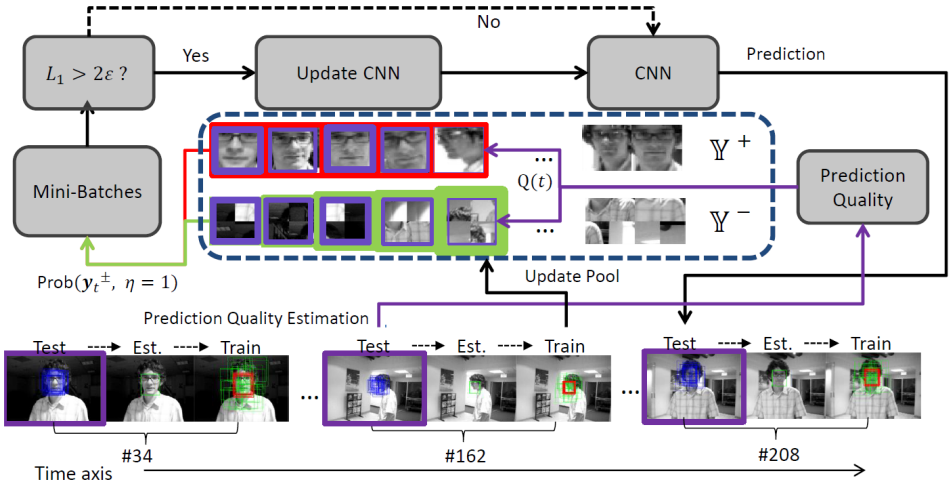


FIGURE 6.3: Work flow of DeepTrack algorithm. The bottom row shows the test, estimation and training stages on a frame. The dashed block covers the positive sample pool Y^+ (red) and negative sample pool Y^- (green). In each pool, the edges of the sample patches indicate their sampling importance. Image courtesy of [LLP16].

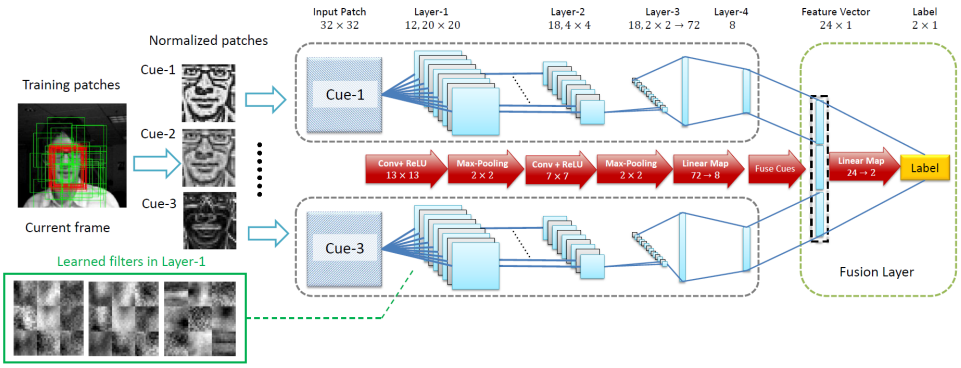


FIGURE 6.4: Architecture of the CNN employing the DeepTrack framework. Image courtesy of [LLP16].

[Wan+15a] conducted a study on the properties of CNN features pre-trained offline for classification tasks on the ImageNet dataset, and utilized the representations learned in the intermediate convolutional layers. They found that a top layer serves as a generic category detector as it encodes more semantic features, while a lower layer encodes a specific-instance detector with more discriminative information. Thus, the Fully Convolutional Network-based Tracker (FCNT) is proposed, which jointly considers two convolutional layers of different levels to complement each other in handling drastic appearance change and distinguishing the target object from similar objects. The tracking framework follows the pipeline illustrated in figure 6.5. It automatically selects discriminative feature maps from the lower and higher layers of the pre-trained VGG network [SZ14]. A general network GNet, capturing the category information of the target, is built on top of the selected feature maps of the higher layer, while a specific network SNet, discriminating the target from background, is built on top of the selected feature maps of the

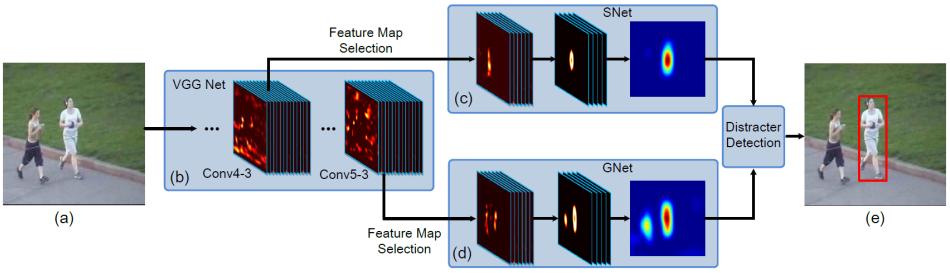


FIGURE 6.5: Pipeline of the FCNT [Wan+15a] algorithm. Image courtesy of [Wan+15a].

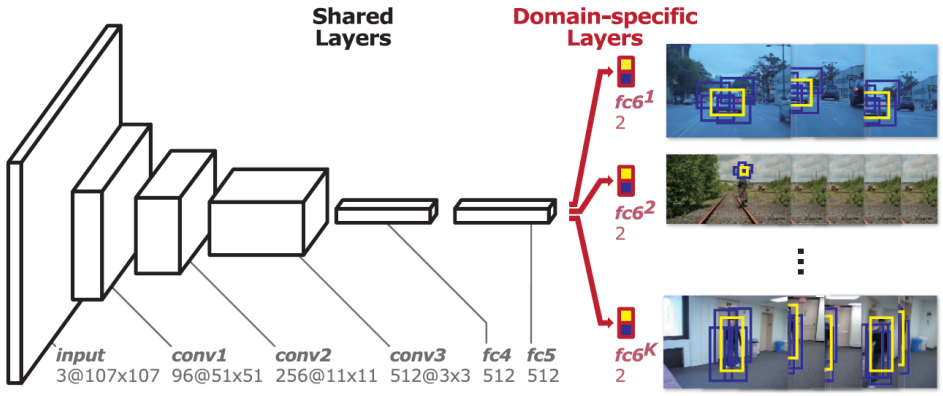


FIGURE 6.6: MDNet [NH15] architecture, consisting of shared layers and K branches of domain-specific layers. Yellow and blue bounding boxes denote the positive and negative samples in each domain, respectively. Image courtesy of [NH15].

lower layer. At every frame, the ROI is propagated through the fully convolutional network and generates two foreground heat maps by GNet and SNet. The final object localization is determined by a distracter detection scheme that decides which heat map to use.

In a similar way, Ma *et al.* [Ma+15] exploited the features extracted from the pre-trained VGG network convolutional layers, as the last layers encode semantic information of targets which is robust to appearance variations, while the earlier layers provide more precise localization but are less invariant to appearance changes. Three correlation filter-based trackers are applied on the different convolutional layers and a coarse-to-fine search on the multi-level correlation response maps infers the location of targets.

Rather than a single frame as input, Fan *et al.* [Fan+10] proposed to use image pairs of two adjacent frames, as well as local and global information, to train a CNN and learn spatial and temporal features in order to estimate the location and scale of the target object The discriminative CNN model extracts discriminant spatial and temporal features for specific object class tracking, where the features are learned from a parametric feature pool. The model also fuses local and global information with multiple path ways in the CNN. This shift-variant architecture is designed so as to alleviate the drift problem when the distracting objects are similar to the target in cluttered environment.

More recently, Nam and Han [NH15] proposed to learn representations

for visual tracking based on a discriminatively trained CNN, named the Multi-Domain Network (MDNet). In an offline phase, the CNN is pre-trained using a large set of tracking videos to obtain a generic target representation. The network, presented in figure 6.6, is composed of shared layers and multiple branches of domain-specific layers, where domains correspond to individual training sequences and each branch is responsible for binary classification to identify the target in each domain. By distinguishing domain-independent and domain-specific information, the shared layers of MDNet updated in every iteration, with all the training video sequences, are able to learn a generic feature representation. The online tracking of a new sequence is executed by constructing a new network combining the shared layers in the pre-trained CNN with a new binary classification layer, *i.e.* removing the the domain-specific layers, which is updated online. The MDNet achieved outstanding tracking performance and ranked first in the VOT2015 [Kri+15] challenge.

Convolutional networks have been greatly used for feature extraction in the aim of localising the objects to track and showed great performance. However, our aim in using CNN is *not* to extract features that represents the object, but rather features which characterise the global scene and its context in the aim of predicting the suitable tracker to this context. It is a new problematic to confront to convolutional networks.

6.3 CNN-based Tracker Selection

The main idea is to use a CNN to automatically extract global features representing the scene directly from input image frames and learn to predict the most suitable tracker, from the pool of N trackers, depending on this context, at each point in time. From the available deep learning framework, we chose to use the well-known and widely used Caffe library [Jia+14] to implement the network as it is the most compatible to our working environment.

6.3.1 Network architecture

We implemented a CNN architecture, illustrated in figure 6.7. The network is composed of three parallel streams of two convolutional layers for the three channels of the input data (Hue, Saturation and Value) and three fully connected layers. The three parallel streams are constructed in the same way for each cue.

Each convolution layer is followed by an average 3×3 patch pooling operation which reduces the obtained feature map to a lower dimension and then a ReLU (Rectified Linear Units) layer which increases the nonlinear properties of the decision functions. The first convolution employs 4 kernels with size 5×5, while the second uses 16 kernels with size 3×3. Then, a fully connected layer and sigmoid function are applied to each stream before concatenating them, adding the non-image data and applying another fully connected layer sigmoid function before the final N neurones.

Finally, the loss is computed based on a Euclidian loss function which computes the sum of squares of differences between the last fully connected

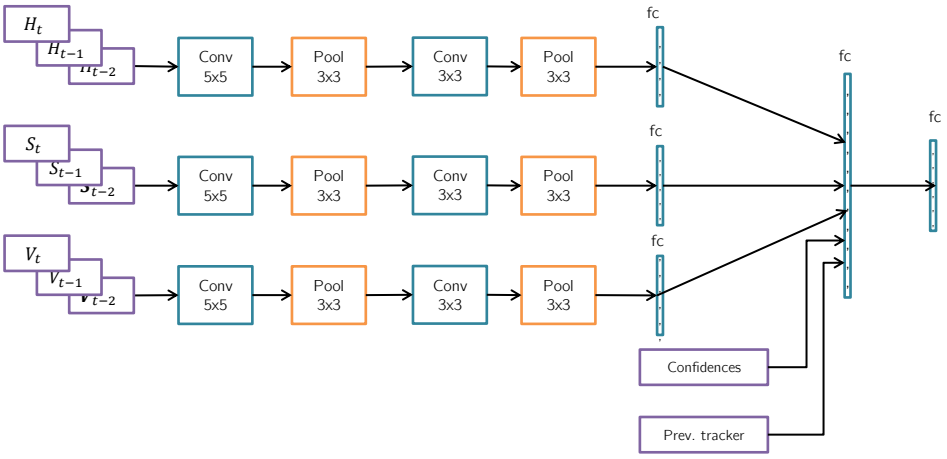


FIGURE 6.7: CNN architecture used for CNN-based tracker selection. For simplicity, the RelU following the pooling layers (pool) and the sigmoid layers following the fully-connected layers (fc) are not shown.

layer of N neurones (after the sigmoid function is applied) and the "ranking" ground-truth vector computed using the F-scores of the trackers, as previously presented in section 5.3.1.

The implemented architecture is straightforward, employing three streams of convolutions confining the images cues to let the network learn different features from each cue, while keeping the number of filters relatively low in the attempt of avoiding the over-fit of the network.

6.3.2 Data Inputs

In the same way as in our previously presented frameworks SCBT and SCBT+, we use three different data inputs. The image frames, individual trackers' confidences and the identifier of the previously selected tracker, from a time window of 3 frames, are employed in the presented CNN.

Firstly, the main inputs of the CNN are the video images from the frames t, $t-1$ and $t-2$. The images I_t, I_{t-1} and I_{t-2} are converted from RGB to HSV cue domains:

$$I_t = \{H_t, S_t, V_t\} \tag{6.1}$$
$$I_{t-1} = \{H_{t-1}, S_{t-1}, V_{t-1}\} \tag{6.2}$$
$$I_{t-2} = \{H_{t-2}, S_{t-2}, V_{t-2}\}. \tag{6.3}$$

As we did for the scene context features in the SCBT framework, we employed the HSV colour space in the aim of separating intensity information from colour.

Then, the channels in each image are isolated in order to group them by cue, thus giving three images H, S and V for the three parallel convolutional paths. They are 3-dimensional images, where the 3^{rd} dimension is time frames:

Iterations		Validation rate	Test rate
500.000	class 0	0 %	0%
	class 1	69.65%	82.39%
	class 2	69.52%	85.65%
	total	**43.80%**	**51.95%**
760.000	class 0	75.49 %	76.23%
	class 1	51.26 %	68.99%
	class 2	74.28 %	84.43%
	total	**65.60 %**	**74.90%**
2.580.000	class 0	80.40 %	81.98%
	class 1	81.60 %	86.96%
	class 2	71.02 %	81.38%
	total	**78.73 %**	**83.97%**

TABLE 6.1: Evolution of the validation and correct classification rates for the proposed CNN, trained on the Princeton dataset and tested on the VOT2013 dataset at different iteration steps. Each class correspond to an individual tracker, 0: KCF RAW, 1: KCF HOG, 2: KCF LAB.

$$data_H = \{H_t, H_{t-1}, H_{t-2}\} \tag{6.4}$$
$$data_S = \{S_t, S_{t-1}, S_{t-2}\} \tag{6.5}$$
$$data_V = \{V_t, V_{t-1}, V_{t-2}\}. \tag{6.6}$$

With this scheme, the CNN can learn to extract different features from each cue, while maintaining the temporal knowledge. The input images are also resized to a size of 60×80 in order capture the global features of the scene rather than the details of the objects present in it.

The confidence values from the individual trackers are also integrated into the CNN. Because they are numerical data, the vector of $3 \times N$ values (for N trackers and 3 frames) is added after the first fully connected layer. Finally, the previously selected tracker vector of 3 values (for 3 frames) is integrated the same way as the confidence values.

6.4 Preliminary Experimental Results

To collect the necessary data, the individual trackers (KCF trackers presented in 5) are ran over the Princeton and VOT2013 datasets to collect the confidence values, the F-scores and the label, *i.e.* the best performing tracker, for the current frame. Then, the convolutional network is trained on the Princeton dataset. During the tracking, the network is evaluated at frequent intervals on the validation dataset (a subset of Princeton) to follow the evolution of the training. We evaluated a few key snapshot networks from different iterations on the VOT2013 dataset. The validation (and classification (per-class and total) rates are presented in table 6.1.

We can first observe that during the first 700.000 iterations, the network only responds 2 classes out of 3. We can assume that the network is stuck in a

local minimum, where it over-fit to predicting only 2 classes, which increases the global the total classification rate. Once the network succeeds in leaving the local minimum, the network stabilises at validation rates between 55% and 65%. The testing rate at iteration 760.000 achieves a score of 74.90%.

After more iterations, the network validation rates reach 78.73% and a high testing rate of 83.97%. However, when using this network in a tracking framework, the network tends to always respond the same tracker as the previously picked. This means that the CNN over-fitted and mainly uses the information from the "previously selected tracker" neurones.

On the other hand, the network at iteration 760.000, doesn't show the same severity of over-fit. In a tracking framework, the network shows to be able to switch from a tracker to another. However, the over-fit is still present as the network tends to keep responding the same previous tracker, while the changes from one tracker to the next "best performing" come with a slight delay when changes occurs in a video. Because of this, the failures in tracking increase and the framework looses robustness.

6.5 Conclusion

In this chapter, we experimented one of the perspectives and possible improvements of the context-based tracker selection framework by introducing a Convolutional Neural Network. The goal of the CNN is to automatically learn to extract effective "scene context features" from the input images and predict the most suitable tracker depending on the input scene for each video frame. Thus, the CNN would replace the two steps of extracting handcrafted scene features and learning to predict the correct tracker into one single network.

The preliminary experiments show on that the network is able to learn some information from the context but tends to over-fit. Despite the significant classification rate, the training of the network is lacking.

We can explain this because of the complexity of the problem to be learned by the CNN. In fact, the employed CNN architecture is not sufficient for the task of extracting meaningful features to characterise the scene, rather than features to describe specific object, as usually done in literature. Moreover, this problem of learning to extract the features associated a specific tracker is very abstract and of semantic nature, thus making the task too complex for the CNN and resulting in the over-fitting of the network. Another inconvenient in the use of a CNN is the need for long training and large datasets.

The results are still promising, as such a type of convolutional network has not been discussed in literature. While we experimented with a few possible architectures which did not give as promising results as the presented one, other architectures and training strategies need to be explored to better adapt the CNN to the given problem. On possibility would be to use recurrent network such as Long short-term memory (LSTM) network to better solve the temporal constraints, especially regarding the previously selected tracker component.

Chapter 7

Conclusion and Perspectives

In this thesis, we concentrated on studying the problem of online object tracking. In unconstrained environments, this computer vision task faces many challenges due to the unknown nature of the object and all the possible changes that can occur to this object or the scene itself. To be able to deal with the various changes, many different tracking strategies and classification models exist in literature. Broad categories of tracking models were reviewed in chapter 2, and more specific tracking frameworks were presented in the relevant chapters 3, 4 and 6.

We particularity focused on the approach of ensemble tracking, a strategy in which multiple trackers or tracking models are used jointly, either by fusion or selection, in order to improve the overall tracking performance. In fact, in unconstrained environments, the tracking task becomes difficult to generalise to all the possible scenarios. In order to face these challenges, one intuitive approach is to combine the strengths of multiples complementary trackers. Considering that visual tracking algorithms are generally specialised on different image conditions, a combination of complementary tracking models would enable the coverage of a larger set of scene conditions. Based on this strategy, we proposed multiple novel frameworks throughout this thesis in the aim of improving the tracking performance.

The first system we introduced, in chapter 3, presented a simple and efficient combination of trackers. This framework, named the Spatial and Temporal Coherence tracker selection framework (STC), recurrently selects the most suitable tracker from a pool of independent trackers. The individual trackers run in parallel, each providing the framework with an object position (bounding box) and a confidence value in this result. At each point in time, a spatio-temporal criterion is enforced to eliminate temporarily the trackers considered having drifted. Then, using the normalised confidence values, a tracker is selected for the current frame and its resulting bounding box is considered as the object position. Moreover, a selective update is applied in which only the selected tracker is updated for the current frame with its corresponding result, in the aim of minimising the possible drift of the trackers considered "lost".

In our experiments, three Online AdaBoost trackers were implemented with different features (Haar, HOG and HOC). We started by evaluation these individual trackers to understand the behaviour of the trackers and their combination. We observed that although the tracking is based on the same learning model, with different features the performance of the trackers will depend on the environment and nature of the object. Furthermore, computing the "best theoretical combination" of the OAB trackers showed us the maximum performance that can be obtained from combining and selecting

the correct tracker at each frame. The overall tracking performance of the combination is greatly increased in comparison to the individual trackers, which justifies our approach of using combination of trackers.

Then, the tracking performance of the STC framework was evaluated and compared to other baseline tracker combination methods such as the confidence based selection, the low-level fusion of features, or high level fusion of bounding boxes. The proposed framework proves to outperform all the baselines. In fact, in contrast to the other methods, the spatio-temporal criterion helps the selection framework to avoid possible drifting trackers, while being able to switch from one tracker to a more confident one. Moreover the tracker update strategy keeps the trackers independent which also minimises the risk of drift.

This selection framework has the benefit of being simple and adding very little computational complexity over the individual trackers. Using only spatial and temporal information and the confidence value, the framework succeeds in improving the overall tracking performance. Although efficient, the framework does not take into account information from the scene context, as the changes in the scene are directly correlated with changes in the performance of the trackers. Introducing the context into the selection framework would help improve the selection and prediction rate and further increase the tracking performance.

In the subsequent chapter, we considered the use of scene context directly in the process of tracker selection. In fact, it is intuitive to use context-based knowledge to direct the selection of the trackers in order to to be able to cope with the variations in scene conditions. We thus proposed a novel framework, the Scene Context-Based Tracker selection framework (SCBT), in which context information is extracted from the scene and used to select the most suitable tracker at each video frame. To represent the context information, we designed a set of "scene features" based on first and second order statistics of main image characteristics (*e.g.* intensity, hue, motion vectors).

These scene features are extracted from the global input image and a local region (*i.e.* region of interest of the target object). Combining the scene features with the confidences values, as well as the identifier of the previously selected tracker, from a sliding window of three frames, a classifier is trained to predict the most suitable tracker depending the input context. During tracking, the scene context classifier makes a prediction representing the probabilities for each tracker to be the most suitable one which is subsequently filtered by a Hidden Markov Model (HMM) to ensure some temporal continuity of the tracker selection and reject outliers. The tracker with the highest probability is selected for the current frame. Finally, to smooth the overall trajectory, the sequence of selected bounding boxes is processed by a Kalman Filter as a post-processing step.

We employed the same previous OAB trackers to evaluate the SCBT framework. We evaluated the scene context classifier by measuring the classification rate of the classifier when trained with different groups of inputs. We observed that the extracted scene features alone are not sufficient, but with the addition of the confidence values, and the previously selected tracker at several times steps, we are able to obtain a high classification rate. In fact, it is the association of both the context scene features and tracker features

that enables the classifier to extract and learn the correlations between the scene information and the trackers' performance.

The SCBT framework and its different components was further evaluated on a tracking benchmark. We observed that the classifier is able to outperform the individual trackers in accuracy and robustness and shows the efficiency of the HMM and Kalman filter as they increase the tracking robustness in comparison to using the classifier alone. We also compared the SCBT to the previous STC framework and showed the effectiveness of the introduction of context in the tracker selection process. Moreover, the SCBT framework ranks among the top trackers in the VOT2013 benchmark in terms of robustness. However in terms of accuracy, the framework is performing worse compared to the benchmark methods as it is limited by the accuracy of the individual trackers used.

To overcome the limitations encountered by the SCBT framework, we proceeded in a thorough analysis and evaluation of the different components of the framework in chapter 5. In fact, two components are important for the performance of the tracker selection framework SCBT: the individual trackers and the context classifier. We thus proposed an extended version of our framework, called SCBT+. Having the same structure as SCBT, the framework is improved by first benchmarking multiple feature-based trackers in order to choose faster, more accurate and robust individual trackers. And to understand the influence of the choice of the individual trackers on the selection framework, the scene context classifiers were also evaluated using different sets of trackers. We observed that the unevenness of the performance of the individual trackers in a set can have a negative impact on the balance between the trackers. While mis-classifications can be managed when the trackers perform equally, in the case of unbalance trackers, prediction errors from the classifier considerably lower the tracking performance of the minimum performance to achieve. However, when using the same tracking algorithm on different domains, for example KCF trackers, the classifier succeeds in outperforming its individual trackers.

We started with the desire to design tracker combination frameworks that can be generic and use any kind of trackers (not necessarily highly robust) available to improve their performance. Thus we used with standard OAB trackers in STC and SCBT. Then to overcome the limitations in accuracy of the frameworks, we experimented with other trackers. While the STC framework is very generic and only needs complementary trackers, we found through our experiments that the SCBT and SCBT+ framework needs a more careful choice of trackers with the properties cited above. The SCBT and SCBT+ are less generic than STC as they need more attention in this choice. Nonetheless, these frameworks proved to improve their performance with the new KCF trackers.

In addition to changing the trackers, the training dataset is also enlarged and multiple training strategies are studied. We found that the classic one-of-N and regression classification can be outperformed by a "ranking" strategy which combines the two and gives the classifier the dual information of the best tracker and simplified information about the performance of the remaining trackers.

With the different improvements, the SCBT+ succeeds in outperforming the STC and SCBT frameworks, showing great results in robustness. On

the VOT2013 and VOT2014 benchmarks, SCBT+ showed state-of-the-art tracking performance. However, in VOT2015, the framework showed its limits compared to the participating methods.

One possible perspective for context-based tracker selection would be to learn how to extract automatically the scene features needed by the classifier. In fact, instead of manually designing the scene features which might not be adapted to the various possible representations of the scene context, a Convolutional Neural Network can be employed to solve this problem as these networks learn richer invariant features via multiple non-linear transformations

Thus, in chapter 6, we started to explore the perspective of using a convolutional network to execute the dual tasks automatically extracting global features representing the scene directly from input image frames and learning to predict the most suitable tracker, from the pool of N trackers, depending on this context, at each point in time. We implemented a CNN architecture taking as inputs image frames from a sliding window and separating the cues as to be able to learn to extract different features from each cue. The confidence values the previously selected tracker are also provided to the network.

The preliminary experiments show that the network is able to learn some information from the context but tends to over-fit. In fact, the problem of learning to extract the features associated a specific tracker is very abstract, thus creating a too large semantic gap for the CNN and resulting in the over-fitting of the network. The results are still promising, as such a type of convolutional network has not been discussed in literature. Other architectures need to be explored to adapt the CNN to the present problem as it would be the key to improving the network's learning.

To summarise, combining multiple trackers is an effective way of improving the tracking performance using simple and available tracking algorithms. The use of context in the combination can further help the selection process to adapt to the scene conditions. However, the choice of trackers to use is important as they must be complementary and balanced in terms of performance. The selection frameworks could be further improved by using more complex individual tracker at the expense of computational efficiency and speed of the framework. Employing multi-scale adaptation to handle the possible size variations of the target object would also increase the overall accuracy.

Additionally, providing new 'semantic' scene features to the classifier that would characterise the type of object or type of scene would be an interesting future research direction.

One of the most promising directions is the integration of a convolutional network. Although the results of our preliminary experiments are encouraging, further studies need to be undertaken. Also, improving the ability of CNNs to process multi-dimensional heterogeneous, temporal data is a very interesting perspective.

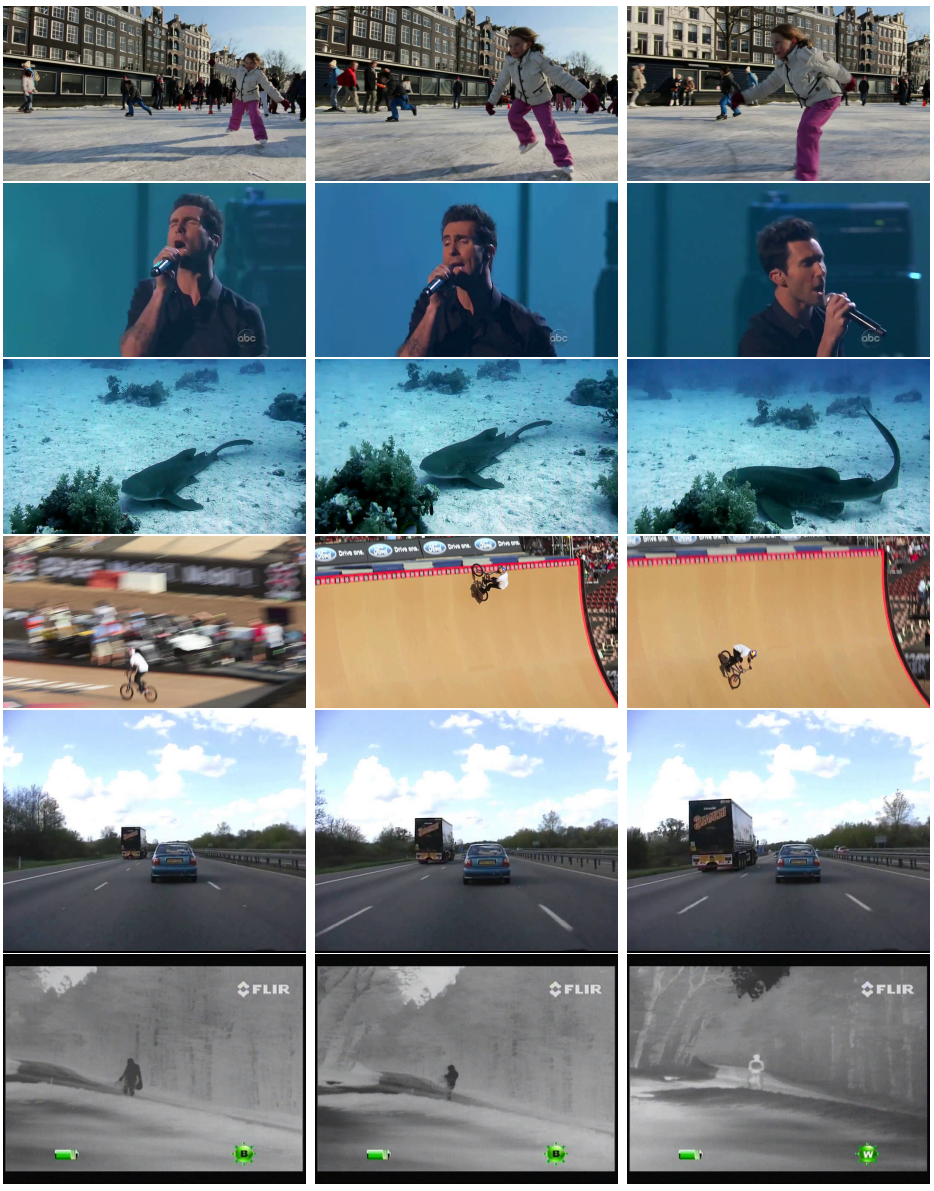
However, one drawback of the context-based selection is the need of offline training which is specific to a particular set of individual trackers. And because of this offline step, training data needs to be collected from large datasets which can be cumbersome. One possible solution would be

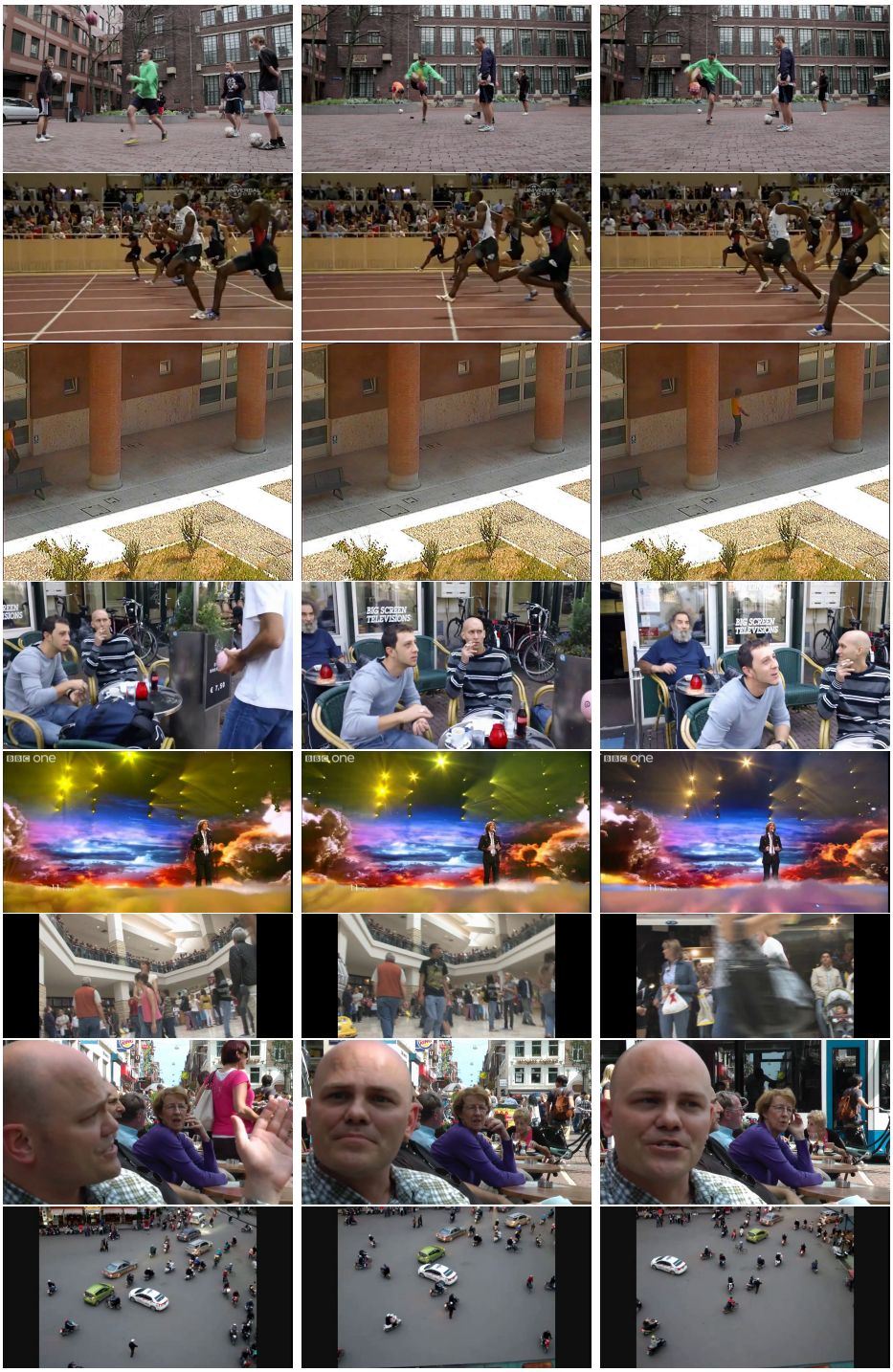
employing transfer learning: for example, importing weights from an pre-trained network and fine-tuning online to adapt the network to the current scenario.

Appendix A

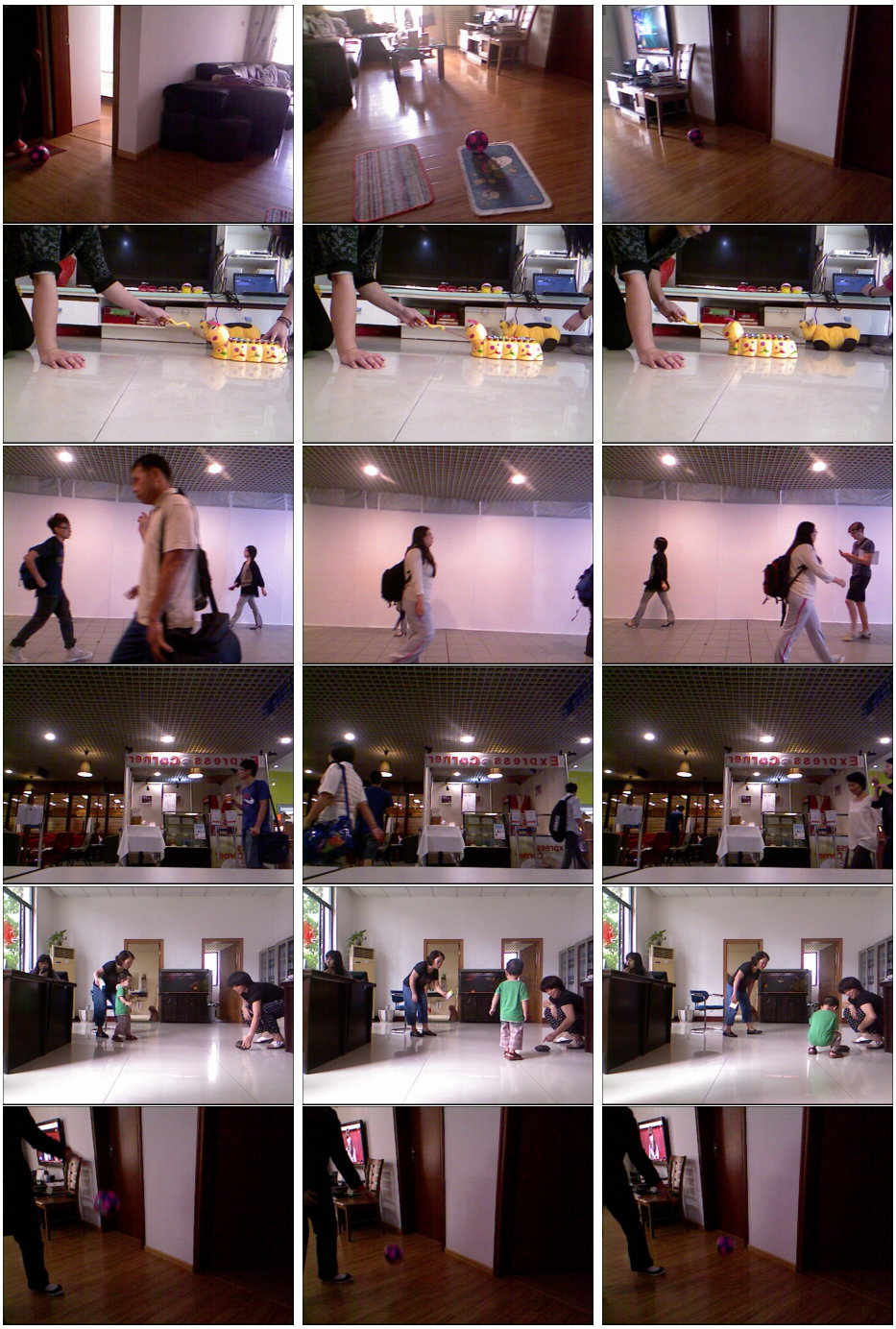
Snippets From Used Tracking Datasets

A.1 ALOV++



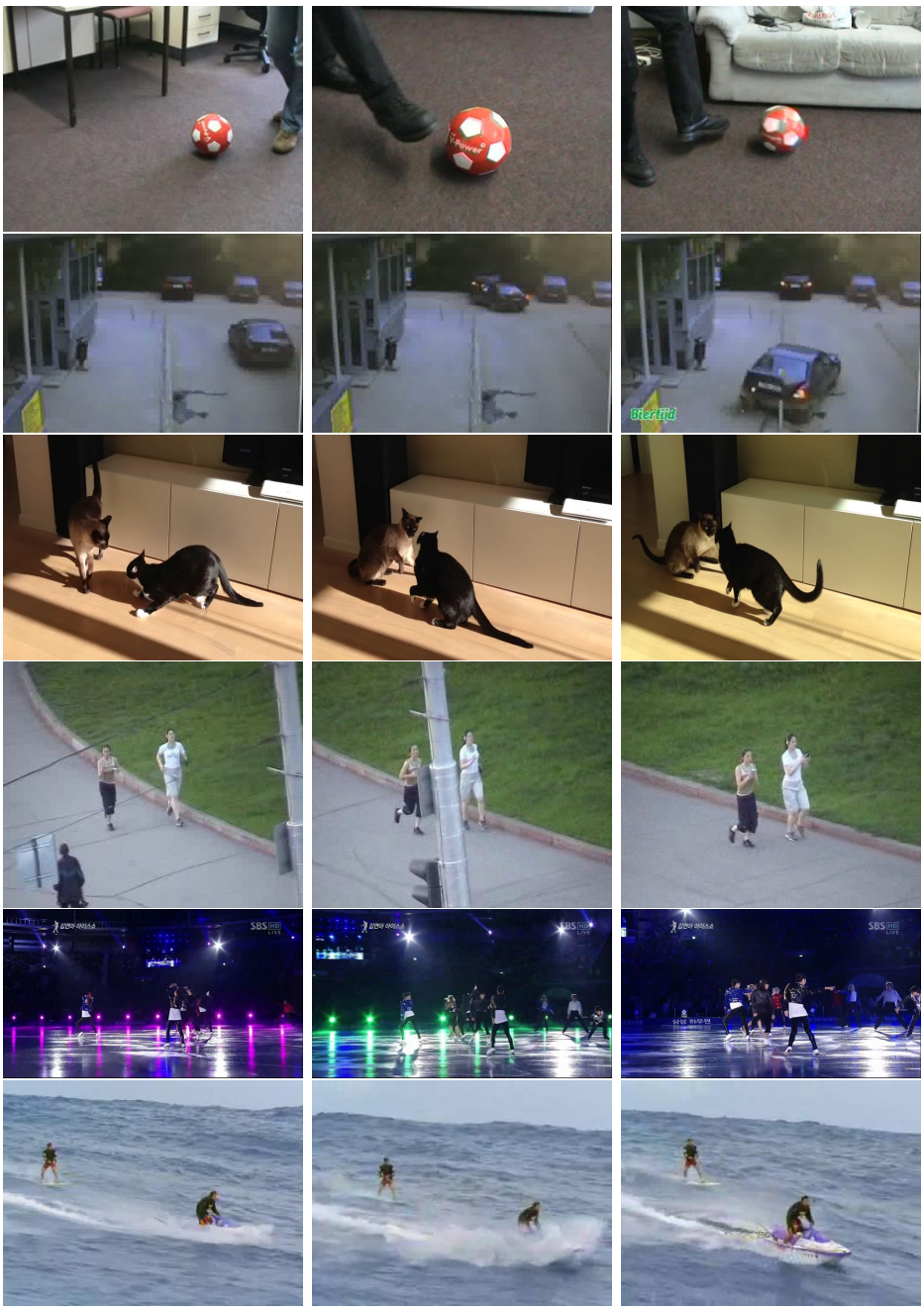


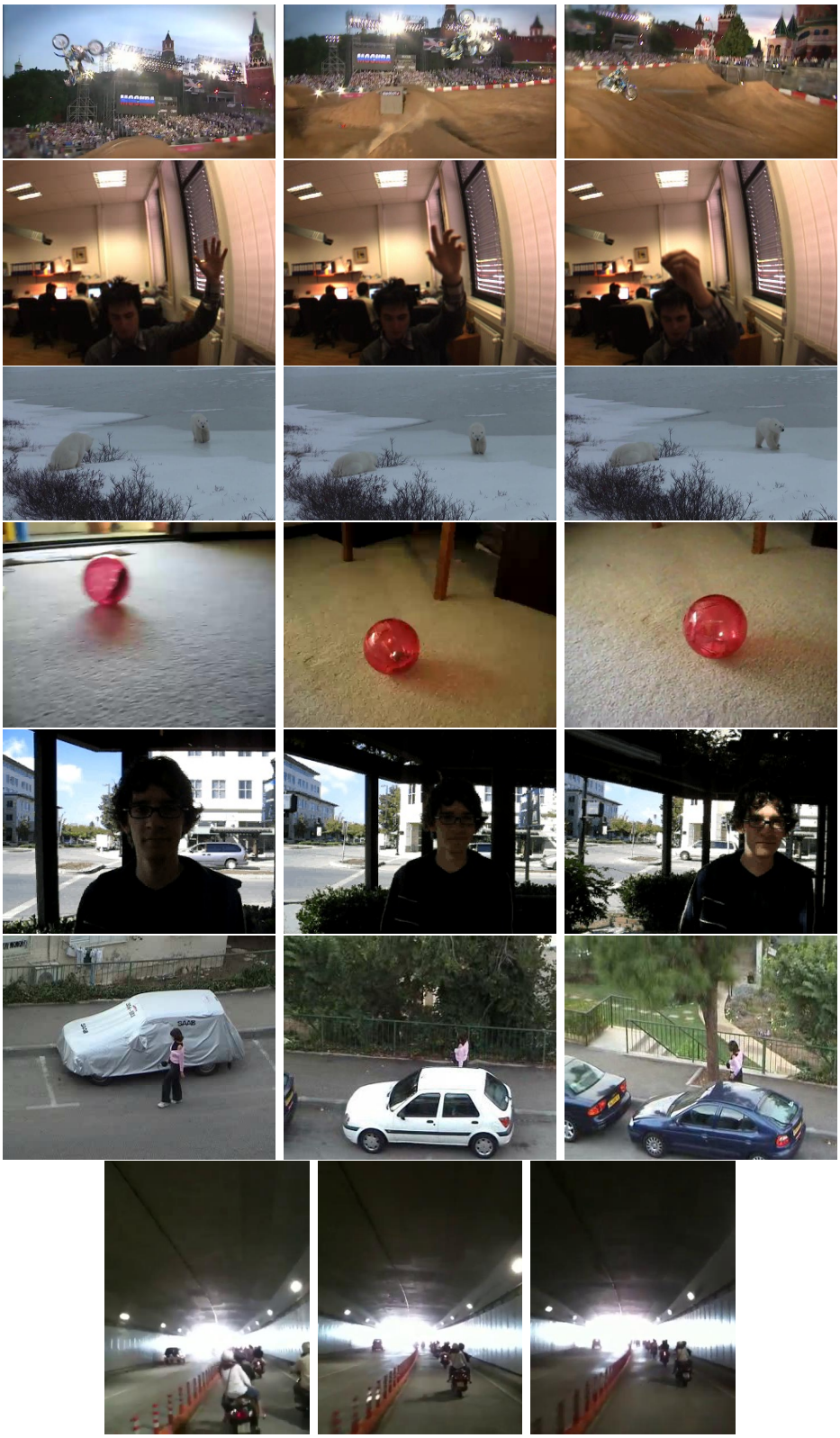
A.2 Princeton



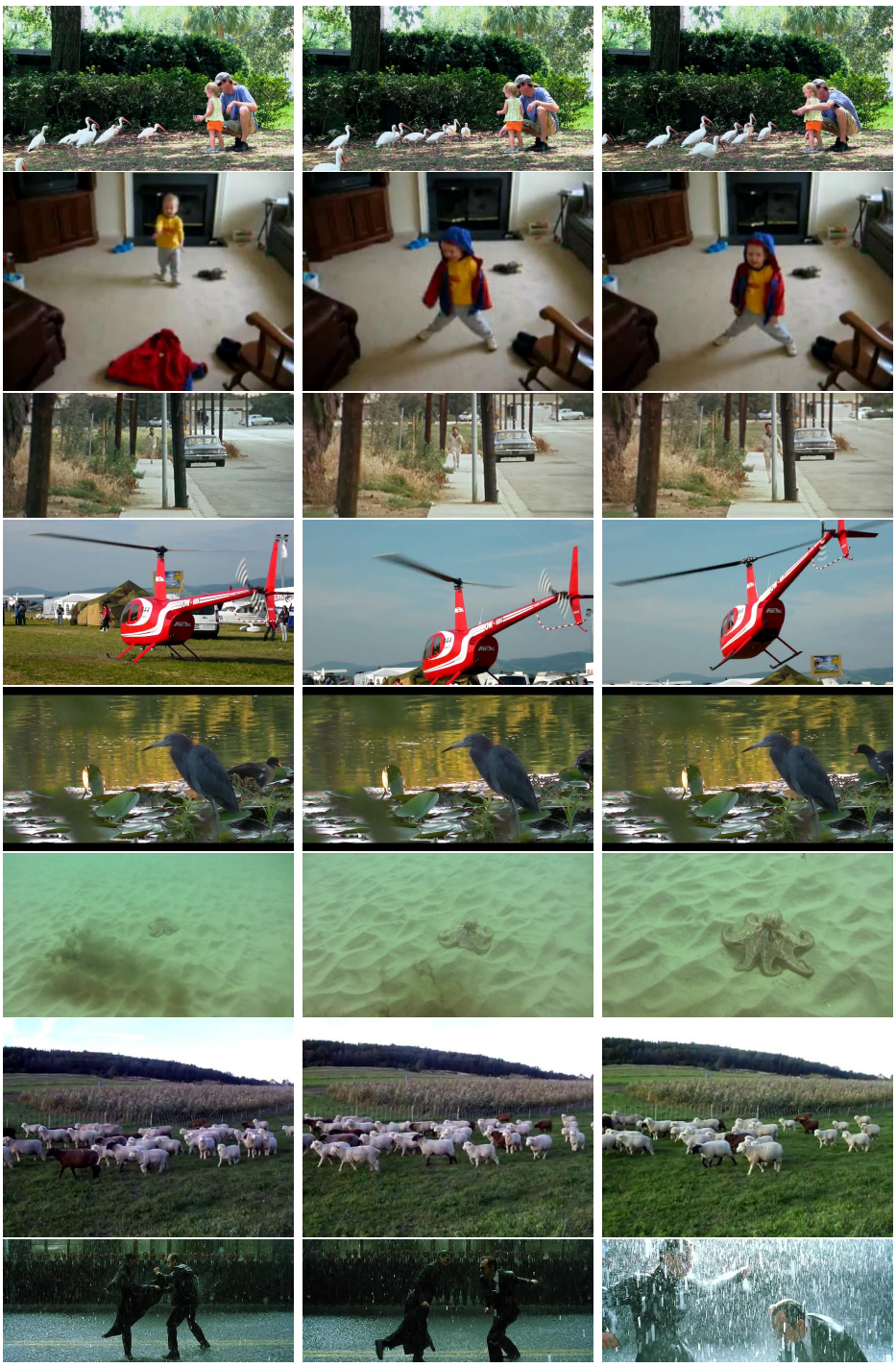
A.3 VOT2013

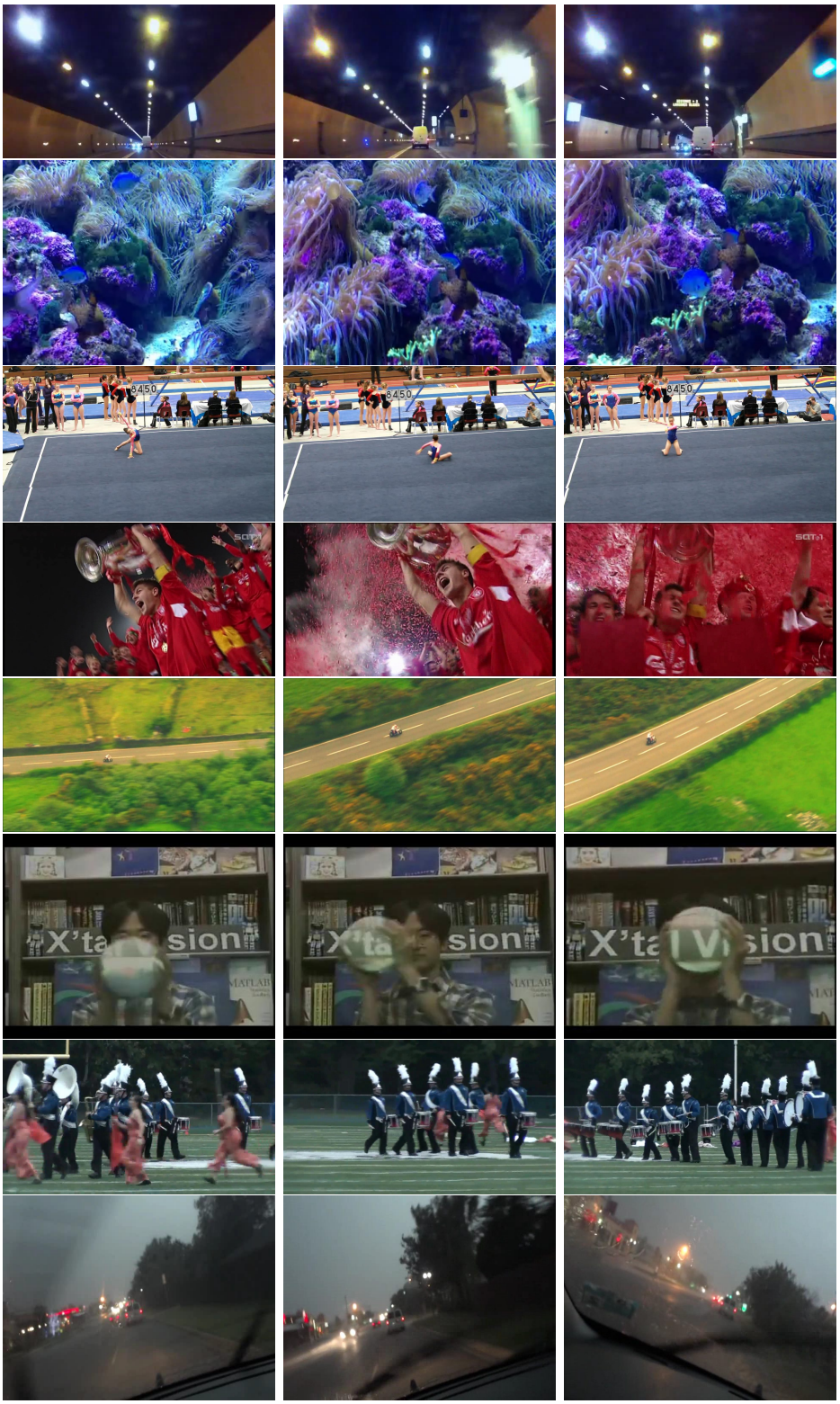
A.4 VOT2014



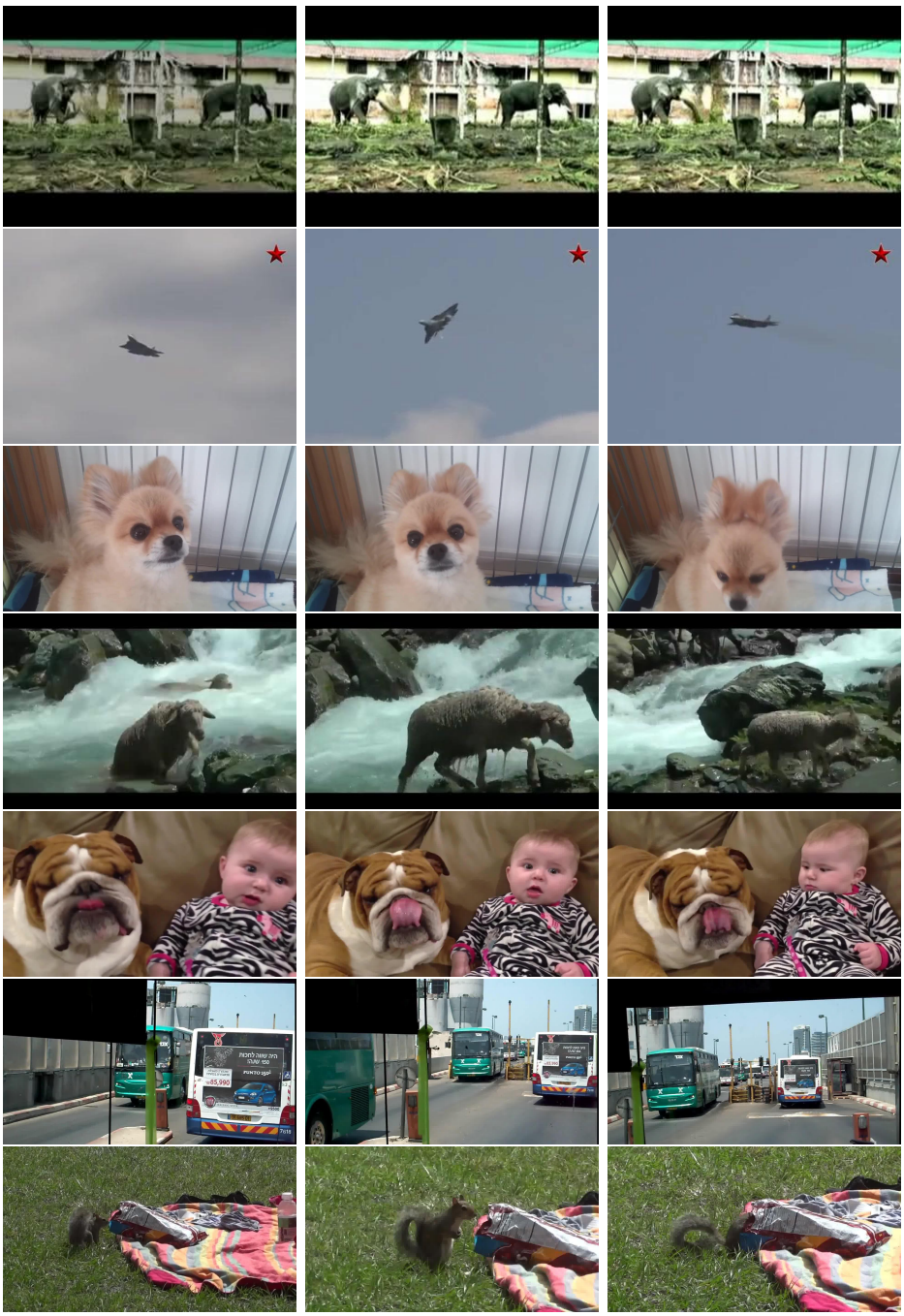


A.5 VOT2015





A.6 ILSVRC15



Contributions

[Kri+15] Matej Kristan et al. “The Visual Object Tracking VOT2015 challenge results”. In: *Proceedings of the International Conference on Computer Vision (Workshops)*. 2015.

[MDB15a] S. Moujtahid, S. Duffner, and A. Baskurt. “Classifying global scene context for On-line Multiple tracker Selection”. In: *Proceedings of British Machine Vision Conference*. 2015, pp. 163.1–163.12.

[MDB15b] Salma Moujtahid, Stefan Duffner, and Atilla Baskurt. “Coherent selection of independent trackers for real-time object tracking”. In: *International Conference on Computer Vision Theory and Applications*. 2015, pp. 584–592.

[Mou+16] Salma Moujtahid et al. “On-line tracker selection using visual scene context”. In: *Submitted to Journal of Computer Vision and Image Understanding*. 2016.

Bibliography

[ARS06] Amit Adam, Ehud Rivlin, and Ilan Shimshoni. "Robust Fragments-based Tracking using the Integral Histogram". In: *Proceedings of the Conference on Computer Vision and Pattern Recognition*. 2006, pp. 798–805.

[Aga+04] Aseem Agarwala et al. "Keyframe-based Tracking for Rotoscoping and Animation". In: *ACM SIGGRAPH 2004 Papers*. SIGGRAPH '04. 2004, pp. 584–591.

[Aru+02] M Sanjeev Arulampalam et al. "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking". In: *IEEE Transactions on signal processing* 50.2 (2002), pp. 174–188.

[Avi07] S. Avidan. "Ensemble Tracking". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.2 (2007), pp. 261–271.

[Avi04] Shai Avidan. "Support vector tracking". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.8 (2004), pp. 1064–1072.

[Avi05] Shai Avidan. "Ensemble Tracking". In: *Proceedings of the Conference on Computer Vision and Pattern Recognition*. 2005, pp. 494–501.

[BYB09] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. "Visual tracking with online multiple instance learning". In: *Proceedings of the Conference on Computer Vision and Pattern Recognition*. 2009, pp. 983–990.

[BYB11] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. "Robust Object Tracking with Online Multiple Instance Learning". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.8 (2011), pp. 1619–1632.

[Bac+12] Yoram Bachrach et al. "How to grade a test without knowing the answers - A Bayesian graphical model for adaptive crowdsourcing and aptitude testing". In: *Proceedings of the International Machine Learning Conference* (2012), pp. 1175–1182.

[Bad+07] Vijay Badrinarayanan et al. "Probabilistic color and adaptive multi-feature tracking with dynamically switched priority between cues". In: *Proceedings of the International Conference on Computer Vision*. IEEE. 2007, pp. 1–8.

[BY14] S. H. Bae and K. J. Yoon. "Robust Online Multi-object Tracking Based on Tracklet Confidence and Online Discriminative Appearance Learning". In: *Proceedings of the Conference on Computer Vision and Pattern Recognition*. 2014, pp. 1218–1225.

[BPS14] Christian Bailer, Alain Pagani, and Didier Stricker. "A Superior Tracking Approach: Building a Strong Tracker through Fusion". In: *Proceedings of the European Conference on Computer Vision*. 2014, pp. 170–185.

[BM04] Simon Baker and Iain Matthews. "Lucas-kanade 20 years on: A unifying framework". In: *International Journal of Computer Vision* 56.3 (2004), pp. 221–255.

[Bao+12] Chenglong Bao et al. "Real time robust L1 tracker using accelerated proximal gradient approach." In: *Proceedings of the Conference on Computer Vision and Pattern Recognition*. 2012, pp. 1830–1837.

[BR11] Ben Benfold and Ian Reid. "Stable multi-target tracking in real-time surveillance video". In: *Proceedings of the Conference on Computer Vision and Pattern Recognition*. IEEE. 2011, pp. 3457–3464.

[Bir98] S Birchfield. "Elliptical head tracking using intensity gradients and color histograms". In: *Proceedings of the Conference on Computer Vision and Pattern Recognition*. 1998, pp. 232–237.

[BJ98] Michael J Black and Allan D Jepson. "Eigentracking: Robust matching and tracking of articulated objects using a view-based representation". In: *International Journal of Computer Vision* 26.1 (1998), pp. 63–84.

[BL08] Matthew B Blaschko and Christoph H Lampert. "Learning to localize objects with structured output regression". In: *Proceedings of the European Conference on Computer Vision*. 2008, pp. 2–15.

[Bol+09] David S Bolme et al. "Simple real-time human detection using a single correlation filter". In: *Performance Evaluation of Tracking and Surveillance (PETS-Winter), 2009 Twelfth IEEE International Workshop on*. 2009, pp. 1–8.

[Bol+10] David S Bolme et al. "Visual object tracking using adaptive correlation filters". In: *Proceedings of the Conference on Computer Vision and Pattern Recognition*. 2010, pp. 2544–2550.

[BUB08] Antoine Bordes, Nicolas Usunier, and Léon Bottou. "Sequence labelling SVMs trained in one pass". In: *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. 2008, pp. 146–161.

[Bor+07] Antoine Bordes et al. "Solving multiclass support vector machines with LaRank". In: *Proceedings of the International Conference on Machine Learning*. 2007, pp. 89–96.

[Bou01] Jean-Yves Bouguet. "Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm". In: *Intel Corporation* 5.1-10 (2001), p. 4.

[BH00] Yuri Boykov and Daniel P Huttenlocher. "Adaptive Bayesian recognition in tracking rigid objects". In: *Proceedings of the Conference on Computer Vision and Pattern Recognition*. Vol. 2. 2000, pp. 697–704.

[BH01] K. Briechle and U. D. Hanebeck. "Template Matching Using Fast Normalized Cross Correlation". In: *Proceedings of SPIE: Optical Pattern Recognition XII*. Vol. 4387. 2001, pp. 95–102.

[BC86] Ted J Broida and Rama Chellappa. "Estimation of object motion parameters from noisy images". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 1 (1986), pp. 90–99.

[BC13] Asad A. Butt and Robert T. Collins. "Multi-target Tracking by Lagrangian Relaxation to Min-cost Network Flow." In: *Proceedings of the Conference on Computer Vision and Pattern Recognition*. 2013, pp. 1846–1853.

[Cai+13] Zhaowei Cai et al. "Structured Visual Tracking with Dynamic Graph". In: *Proceedings of the Asian Conference on Computer Vision*. 2013, pp. 86–97.

[Cav] *CAVIAR (Context Aware Vision using Image-based Active Recognition) dataset*. http://homepages.inf.ed.ac.uk/rbf/CAVIAR/. 2002.

[CKL13] Luka Cehovin, Matej Kristan, and Ales Leonardis. "Robust Visual Tracking Using an Adaptive Coupled-Layer Visual Model". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.4 (2013), pp. 941–953.

[Ceh] *Cehovin dataset*. http://www.vicos.si/User:Lukacu/Research/Tracking. 2009.

[CHT15] Zhe Chen, Zhibin Hong, and Dacheng Tao. "An Experimental Survey on Correlation Filter-based Tracking". In: *arXiv preprint arXiv:1509.05520* (2015).

[CS10] Dung M Chu and Arnold WM Smeulders. "Thirteen hard cases in visual tracking". In: *Proceedings of the IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. 2010, pp. 103–110.

[Cif+12] Cristina García Cifuentes et al. "Motion models that only work sometimes". In: *Proceedings of British Machine Vision Conference*. 2012, 12–pages.

[Col+01] R. T. Collins et al. "Algorithms for cooperative multisensor surveillance". In: *Proceedings of the IEEE* 89.10 (2001), pp. 1456–1477.

[CL05] Robert T. Collins and Yanxi Liu. "On-Line Selection of Discriminative Tracking Features". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.10 (2005), pp. 1631–1643.

[CM02] Dorin Comaniciu and Peter Meer. “Mean Shift: A Robust Approach Toward Feature Space Analysis”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.5 (May 2002), pp. 603–619.

[CRM03] Dorin Comaniciu, Visvanathan Ramesh, and P. Meer. “Kernel-based object tracking”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25.5 (2003), pp. 564–577.

[CRM00] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. “Real-time tracking of non-rigid objects using mean shift”. In: *Proceedings of Conference on Computer Vision and Pattern Recognition*. Vol. 2. 2000, pp. 142–149.

[DT05] Navneet Dalal and Bill Triggs. “Histograms of Oriented Gradients for Human Detection”. In: *Proceedings of Conference on Computer Vision and Pattern Recognition*. 2005, pp. 886–893.

[Dan+14a] Martin Danelljan et al. “Accurate Scale Estimation for Robust Visual Tracking”. In: *Proceedings of British Machine Vision Conference*. 2014.

[Dan+14b] Martin Danelljan et al. “Adaptive Color Attributes for Real-Time Visual Tracking”. In: *Proceedings of Conference on Computer Vision and Pattern Recognition*. 2014, pp. 1090–1097.

[Dan+15] Martin Danelljan et al. “Learning Spatially Regularized Correlation Filters for Visual Tracking”. In: *Proceedings of International Conference on Computer Vision*. 2015, pp. 4310–4318.

[DLLP97] Thomas G Dietterich, Richard H Lathrop, and Tomás Lozano-Pérez. “Solving the multiple instance problem with axis-parallel rectangles”. In: *Artificial intelligence* 89.1 (1997), pp. 31–71.

[DVM11] TB Dinh, Nam Vo, and G Medioni. “Context tracker: Exploring supporters and distracters in unconstrained environments”. In: *Proceedings of British Machine Vision Conference*. 2011, pp. 1177–1184.

[DG16] Séverine Dubuisson and Christophe Gonzales. “A survey of datasets for visual tracking”. In: *Machine Vision and Applications* 27.1 (2016), pp. 23–52.

[DG13] Stefan Dufffner and Christophe Garcia. “PixelTrack: a fast adaptive algorithm for tracking non-rigid objects”. In: *Proceedings of the International Conference on Computer Vision*. 2013, pp. 2480–2487.

[DG14] Stefan Dufffner and Christophe Garcia. “Exploiting contextual motion cues for visual object tracking”. In: *Proceedings of the European Conference on Computer Vision*. 2014, pp. 232–243.

[ESF09] ANNA Ellis, A Shahrokni, and JAMES Ferryman. “Overall evaluation of the pets2009 results”. In: *Proceedings of the IEEE International Workshop on PETS*. 2009, pp. 117–124.

[Fan+10] Jialue Fan et al. “Human tracking using convolutional neural networks”. In: *IEEE Transactions on Neural Networks* 21.10 (2010), pp. 1610–1623.

[Fel13] Michael Felsberg. “Enhanced Distribution Field Tracking Using Channel Representations”. In: *Proceedings of the International Conference on Computer Vision (Workshops)*. 2013.

[Fel+10] Pedro F Felzenszwalb et al. “Object detection with discriminatively trained part-based models”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.9 (2010), pp. 1627–1645.

[Fra] *FRAGtrack dataset*. http://www.cs.technion.ac.il/~amita/fragtrack/fragtrack.htm. 2006.

[FS97] Y. Freund and R. Schapire. “A decision-theoretic generalization of on-line learning and an application to boosting.” In: *Journal of Computer and System Sciences* 55.1 (1997), pp. 119–139.

[FS95] Yoav Freund and Robert E Schapire. “A desicion-theoretic generalization of on-line learning and an application to boosting”. In: *European Conference on Computational Learning Theory*. 1995, pp. 23–37.

[FH75] Keinosuke Fukunaga and Larry Hostetler. “The estimation of the gradient of a density function, with applications in pattern recognition”. In: *IEEE Transactions on information theory* 21.1 (1975), pp. 32–40.

[Gal+11] Juergen Gall et al. “Hough forests for object detection, tracking, and action recognition”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.11 (2011), pp. 2188–2202.

[Gao+14] Jin Gao et al. “Transfer learning based visual tracking with gaussian processes regression”. In: *Proceedings of European Conference on Computer Vision*. 2014, pp. 188–203.

[GD04] Christophe Garcia and Manolis Delakis. “Convolutional face finder: A neural architecture for fast and robust face detection”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.11 (2004), pp. 1408–1423.

[Gir+14] Ross Girshick et al. “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: *Proceedings of Conference on Computer Vision and Pattern Recognition*. 2014, pp. 580–587.

[GRB13] Martin Godec, Peter M Roth, and Horst Bischof. “Hough-based tracking of non-rigid objects”. In: *Computer Vision and Image Understanding* 117.10 (2013), pp. 1245–1256.

[Goo] *Google Self-Driving Car Project*. https://www.google.com/selfdrivingcar/. 2002.

[GB06] H. Grabner and H. Bischof. “On-line Boosting and Vision”. In: *Proceedings of Conference on Computer Vision and Pattern Recognition*. Vol. 1. 2006, pp. 260–267.

[GGB06] Helmut Grabner, Michael Grabner, and Horst Bischof. "Real-Time Tracking via On-line Boosting". In: *Proceedings of British Machine Vision Conference*. 2006, pp. 47–56.

[GLB08] Helmut Grabner, Christian Leistner, and Horst Bischof. "Semi-supervised on-line boosting for robust tracking". In: *Proceedings of European Conference on Computer Vision*. 2008, pp. 234–247.

[Gra+10] Helmut Grabner et al. "Tracking the invisible : Learning where the object might be". In: *Proceedings of Conference on Computer Vision and Pattern Recognition*. 2010, pp. 1285–1292.

[GZT11] S. Gu, Y. Zheng, and C. Tomasi. "Linear time offline tracking and lower envelope algorithms". In: *Proceedings of International Conference on Computer Vision*. 2011, pp. 1840–1846.

[HSHT11] Sam Hare, Amir Saffari, and Philip H.S. Torr. "Struck: structured output tracking with kernels". In: *Proceedings of the International Conference on Computer Vision*. 2011, pp. 263–270.

[HHD00] I. Haritaoglu, D. Harwood, and L. S. Davis. "W4: real-time surveillance of people and their activities". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.8 (2000), pp. 809–830.

[Hen+12] João F. Henriques et al. "Exploiting the Circulant Structure of Tracking-by-detection with Kernels". In: *Proceedings of the European Conference on Computer Vision*. 2012, pp. 702–715.

[Hen+15] João F. Henriques et al. "High-Speed Tracking with Kernelized Correlation Filters". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.3 (2015), pp. 583–596.

[Hin+12] Geoffrey Hinton et al. "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups". In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 82–97.

[HMT12] Zhibin Hong, Xue Mei, and Dacheng Tao. "Dual-force metric learning for robust distracter-resistant tracker". In: *Proceedings of the European Conference on Computer Vision*. 2012, pp. 513–527.

[Hon+15] Zhibin Hong et al. "Multi-store tracker (muster): A cognitive psychology inspired approach to object tracking". In: *Proceedings of the Conference on Computer Vision and Pattern Recognition*. 2015, pp. 749–758.

[HAS15] Yang Hua, Karteek Alahari, and Cordelia Schmid. "Online object tracking with proposal selection". In: *Proceedings of the International Conference on Computer Vision*. 2015, pp. 3092–3100.

[IB98a] Michael Isard and Andrew Blake. "Condensation—conditional density propagation for visual tracking". In: *International Journal of Computer Vision* 29.1 (1998), pp. 5–28.

[IB98b] Michael Isard and Andrew Blake. "ICONDENSATION: Unifying low-level and high-level tracking in a stochastic framework". In: *Proceedings of European Conference on Computer Vision*. Springer. 1998, pp. 893–908.

[Jia+14] Yangqing Jia et al. "Caffe: Convolutional Architecture for Fast Feature Embedding". In: *arXiv preprint arXiv:1408.5093* (2014).

[KMM10a] Zdenek Kalal, Jiri Matas, and Krystian Mikolajczyk. "Pn learning: Bootstrapping binary classifiers by structural constraints". In: *Proceedings of Conference on Computer Vision and Pattern Recognition*. IEEE. 2010, pp. 49–56.

[KMM10b] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. "Forward-backward error: Automatic detection of tracking failures". In: *Proceedings of International Conference on Pattern Recognition*. 2010, pp. 2756–2759.

[KMM12] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. "Tracking-Learning-Detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.7 (2012), pp. 1409–1422.

[KSC16] ObaidUllah Khalid, J.C. SanMiguel, and Andrea Cavallero. "Multi-Tracker Partition Fusion". In: *IEEE on Circuits and Systems for Video Technology* (Jan. 2016), pp. –.

[KT04] Mathias Kolsch and Matthew Turk. "Fast 2d hand tracking with flocks of features and multi-cue integration". In: *Proceedings of the Conference on Computer Vision and Pattern Recognition (Workshops)*. 2004, pp. 158–158.

[Kri+14] M. Kristan et al. "The Visual Object Tracking VOT2014 challenge results". In: *Proceedings of the European Conference on Computer Vision (Workshops)*. 2014, pp. 191–217.

[Kri+13] Matej Kristan et al. "The Visual Object Tracking VOT2013 challenge results". In: *Proceedings of the International Conference on Computer Vision (Workshops)*. 2013.

[Kri+15] Matej Kristan et al. "The Visual Object Tracking VOT2015 challenge results". In: *Proceedings of the International Conference on Computer Vision (Workshops)*. 2015.

[Kri+16] Matej Kristan et al. "A novel performance evaluation methodology for single-target trackers". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (preprint)* (2016).

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in Neural Information Processing Systems*. 2012, pp. 1097–1105.

[KXT03] BVK Vijaya Kumar, Chunyan Xie, and Jason Thornton. "Iris verification using correlation filters". In: *Proceedings of International Conference on Audio-and Video-Based Biometric Person Authentication*. 2003, pp. 697–705.

[KL11] J. Kwon and K.M. Lee. “Tracking by sampling trackers”. In: *Proceedings of the International Conference on Computer Vision*. 2011, pp. 1195–1202.

[KL10] Junseok Kwon and K.M. Lee. “Visual tracking decomposition”. In: *Proceedings of Conference on Computer Vision and Pattern Recognition*. 2010, pp. 1269–1276.

[LT+14] Laura Leal-Taixé et al. “Learning an image-based motion context for multiple people tracking”. In: *Proceedings of Conference on Computer Vision and Pattern Recognition* (2014).

[Leb+13] Karel Lebeda et al. “Long-Term Tracking Through Failure Cases”. In: *Proceedings of the International Conference on Computer Vision (Workshops)*. 2013, pp. 153 –160.

[LY11] Jae-Yeong Lee and Wonpil Yu. “Visual tracking by partition-based histogram backprojection and maximum support criteria”. In: *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on*. IEEE. 2011, pp. 2860–2865.

[LRS00] L. Lee, R. Romano, and G. Stein. “Monitoring activities from multiple video streams: establishing a common coordinate frame”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.8 (2000), pp. 758–767.

[LSS05] B. Leibe, E. Seemann, and B. Schiele. “Pedestrian detection in crowded scenes”. In: *Proceedings of the Conference on Computer Vision and Pattern Recognition*. Vol. 1. 2005, pp. 878–885.

[LLR06] Ido Leichter, Michael Lindenbaum, and Ehud Rivlin. “A General Framework for Combining Visual Trackers – “Black Boxes” Approach”. In: *International Journal of Computer Vision* 67.3 (2006), pp. 343–363.

[LGB08] Christian Leistner, Helmut Grabner, and Horst Bischof. “Semi-supervised boosting using visual similarity learning”. In: *Proceedings of Conference on Computer Vision and Pattern Recognition*. 2008, pp. 1–8.

[LLP16] H. Li, Yi Li, and F. Porikli. “Deeptrack: Learning discriminative feature representations online for robust visual tracking”. In: *IEEE Transactions on Image Processing* 25.4 (2016), pp. 1834–1848.

[LLP14a] H. Li, Yi Li, and Fatih Porikli. “DeepTrack: Learning Discriminative Feature Representations by Convolutional Neural Networks for Visual Tracking”. In: *Proceedings of the British Machine Vision Conference*. 2014.

[LLP14b] Hanxi Li, Yi Li, and Fatih Porikli. “Robust online visual tracking with a single convolutional neural network”. In: *Proceedings of the Asian Conference on Computer Vision*. 2014, pp. 194–209.

[Li+12] Quannan Li et al. “Disagreement-Based Multi-system Tracking”. In: *Proceedings of the Asian Conference on Computer Vision*. Vol. 7729. 2012, pp. 320–334.

[LZ14] Yang Li and Jianke Zhu. "A scale adaptive kernel correlation filter tracker with feature integration". In: *European Conference on Computer Vision*. 2014, pp. 254–265.

[Lim+04] Jongwoo Lim et al. "Incremental learning for visual tracking". In: *Advances in Neural Information Processing Systems*. 2004, pp. 793–800.

[LK81] Bruce D. Lucas and Takeo Kanade. "An Iterative Image Registration Technique with an Application to Stereo Vision". In: *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*. 1981, pp. 674–679.

[LZK14] Wenhan Luo, Xiaowei Zhao, and Tae-Kyun Kim. "Multiple Object Tracking: A Review". In: *CoRR* (2014). URL: http://arxiv.org/abs/1409.7618.

[Ma+15] Chao Ma et al. "Hierarchical convolutional features for visual tracking". In: *Proceedings of International Conference on Computer Vision*. 2015, pp. 3074–3082.

[MC09] Emilio Maggio and Andrea Cavallaro. "Learning Scene Context for Multiple Object Tracking". In: *IEEE Transactions on Image Processing* 18.8 (2009), pp. 1873–1884.

[MP14] Mario Edoardo Maresca and Alfredo Petrosino. "Clustering Local Motion Estimates for Robust and Efficient Object Tracking". In: *Proceedings of the European Conference on Computer Vision (Workshops)*. 2014, pp. 244–253.

[MP13] MarioEdoardo Maresca and Alfredo Petrosino. "MATRIOSKA: A Multi-level Approach to Fast Tracking by Learning". In: *Proceedings of Int. Conf on Image Analysis and Processing*. Vol. 8157. 2013, pp. 419–428.

[MV11] Jiri Matas and Tomas Vojir. "Robustifying the Flock of Trackers". In: *Comp. Vis. Winter Workshop*. 2011.

[May82] Peter S Maybeck. *Stochastic models, estimation, and control*. Vol. 3. Academic press, 1982.

[Mil] *MILtrack dataset*. http://vision.ucsd.edu/~bbabenko/miltrack.shtml. 2009.

[Alo] *MILtrack dataset*. http://www.alov300.org/. 2014.

[MD03] Anurag Mittal and Larry S. Davis. "M2Tracker: A Multi-View Approach to Segmenting and Tracking People in a Cluttered Scene". In: *International Journal of Computer Vision* 51.3 (2003), pp. 189–203.

[MDB15a] S. Moujtahid, S. Duffner, and A. Baskurt. "Classifying global scene context for On-line Multiple tracker Selection". In: *Proceedings of British Machine Vision Conference*. 2015, pp. 163.1–163.12.

[MDB15b] Salma Moujtahid, Stefan Duffner, and Atilla Baskurt. "Coherent selection of independent trackers for real-time object tracking". In: *International Conference on Computer Vision Theory and Applications*. 2015, pp. 584–592.

[Mou+16] Salma Moujtahid et al. “On-line tracker selection using visual scene context”. In: *Submitted to Journal of Computer Vision and Image Understanding*. 2016.

[NH15] Hyeonseob Nam and Bohyung Han. “Learning Multi-Domain Convolutional Neural Networks for Visual Tracking”. In: *Computing Research Repository (CoRR)* (2015), pp. –. arXiv: 1510.07945v2 [cs.CV].

[NHH14] Hyeonseob Nam, Seunghoon Hong, and Bohyung Han. “Online Graph-Based Tracking”. In: *Proceedings of the European Conference on Computer Vision*. 2014, pp. 112–126.

[NP14] Georg Nebehay and Roman Pflugfelder. “Consensus-based Matching and Tracking of Keypoints for Object Tracking”. In: *Winter Conference on Applications of Computer Vision*. 2014, pp. 862–869.

[NHY08] SM Shahed Nejhum, Jeffrey Ho, and Ming-Hsuan Yang. “Visual tracking with histograms and articulating blocks”. In: *Proceedings of the Conference on Computer Vision and Pattern Recognition*. 2008, pp. 1–8.

[NBT16] T.L.A. Nguyen, F. Bremond, and J. Trojanova. “Multi-Object Tracking of Pedestrian Driven by Context”. In: *In Proceedings of the 13th IEEE International Conference on Advanced Video and Signal-Based Surveillance* (2016).

[NKMVG03] Katja Nummiaro, Esther Koller-Meier, and Luc Van Gool. “An adaptive color-based particle filter”. In: *Image and Vision Computing* 21.1 (2003), pp. 99–110.

[Oku+04] Kenji Okuma et al. “A Boosted Particle Filter: Multitarget Detection and Tracking”. In: *Proceedings of the European Conference on Computer Vision*. 2004, pp. 28–39.

[OT01] A. Oliva and A. Torralba. “Modeling the shape of the scene: A holistic representation of the spatial envelope”. In: *International Journal of Computer Vision* 42.3 (2001), pp. 145–175.

[POP98] Constantine P Papageorgiou, Michael Oren, and Tomaso Poggio. “A general framework for object detection”. In: *Proceedings of the International Conference on Computer Vision*. 1998, pp. 555–562.

[Pen+13] Thomas Penne et al. “Markov chain monte carlo modular ensemble tracking”. In: *Image and Vision Computing* 31.6 (2013), pp. 434–447.

[PVB04] P. Perez, J. Vermaak, and A. Blake. “Data Fusion for Visual Tracking With Particles”. In: *Proceedings of IEEE* 92.3 (2004), pp. 495–513.

[Pér+02] Patrick Pérez et al. “Color-based probabilistic tracking”. In: *Proceedings of the European Conference on Computer Vision*. 2002, pp. 661–675.

[PDB14] Federico Pernici and Alberto Del Bimbo. "Object tracking by oversampling local features". In: *IEEE transactions on pattern analysis and machine intelligence* 36.12 (2014), pp. 2538–2551.

[PMB15] Horst Possegger, Thomas Mauthner, and Horst Bischof. "In Defense of Color-based Model-free Tracking". In: *Proceedings of Conference on Computer Vision and Pattern Recognition*. 2015, pp. 2113–2120.

[Pro] *PROST dataset*. http://gpu4vision.icg.tugraz.at/index.php?content=subsites/prost/prost.php. 2010.

[RS99] Romer Rosales and Stan Sclaroff. "3D trajectory recovery for tracking multiple objects and trajectory guided recognition of actions". In: *Proceedings of Conference on Computer Vision and Pattern Recognition*. Vol. 2. 1999.

[Ros+08] David A. Ross et al. "Incremental Learning for Robust Visual Tracking". In: *International Journal of Computer Vision* 77.1-3 (2008), pp. 125–141.

[Rus+15] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision* 115.3 (2015), pp. 211–252.

[San+10] Jakob Santner et al. "Prost: Parallel robust online simple tracking". In: *Proceedings of Conference on Computer Vision and Pattern Recognition*. 2010, pp. 723–730.

[SKK02] Marios Savvides, BVK Vijaya Kumar, and Pradeep Khosla. "Face verification using correlation filters". In: *IEEE Automatic Identification Advanced Technologies* (2002), pp. 56–61.

[SBW02] Haim Schweitzer, JW Bell, and Feng Wu. "Very fast template matching". In: *Proceedings of European Conference on Computer Vision*. 2002, pp. 358–372.

[SL12] Laura Sevilla-Lara. "Distribution Fields for Tracking". In: *Proceedings of Conference on Computer Vision and Pattern Recognition*. 2012, pp. 1910–1917.

[Sha+00] Amanda JC Sharkey et al. "The "test and select" approach to ensemble combination". In: *Proceedings of International Workshop on Multiple Classifier Systems*. 2000, pp. 30–44.

[ST94] Jianbo Shi and C. Tomasi. "Good features to track". In: *Proceedings of Conference on Computer Vision and Pattern Recognition*. 1994, pp. 593–600.

[SI07] C. Siagian and L. Itti. "Rapid biologically-inspired scene classification using features shared with visual attention". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.2 (2007), pp. 300–312.

[SZ14] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[Sme+14] Arnold WM Smeulders et al. “Visual tracking: An experimental survey”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.7 (2014), pp. 1442–1468.

[SX13] Shuran Song and Jianxiong Xiao. “Tracking revisited using RGBD Camera: Unified Benchmark and Baselines”. In: *Proceedings of the International Conference on Computer Vision*. 2013.

[SGG09] Severin Stalder, Helmut Grabner, and L Van Gool. “Beyond semi-supervised tracking: Tracking should be as simple as detection, but not simpler than recognition”. In: *Proceedings of the International Conference on Computer Vision (Workshops)*. 2009, pp. 1409–1416.

[SWC09] Björn Stenger, Thomas Woodley, and Roberto Cipolla. “Learning to Track with Multiple Observers”. In: *Proceedings of Conference on Computer Vision and Pattern Recognition*. 2009, pp. 2647–2654.

[Sun+05] Jian Sun et al. “Bidirectional tracking using trajectory segment analysis”. In: *Proceedings of International Conference on Computer Vision*. Vol. 1. 2005, 717–724 Vol. 1.

[Tai+14] Yaniv Taigman et al. “Deepface: Closing the gap to human-level performance in face verification”. In: *Proceedings of Conference on Computer Vision and Pattern Recognition*. 2014, pp. 1701–1708.

[Tan+07] Feng Tang et al. “Co-tracking using semi-supervised support vector machines”. In: *Proceedings of International Conference on Computer Vision*. 2007, pp. 1–8.

[Bir] *The birchfield dataset*. http://www.ces.clemson.edu/~stb/research/headtracker/seq/. 1998.

[TLH05] Ying-Li Tian, M. Lu, and A. Hampapur. “Robust and efficient foreground analysis for real-time video surveillance”. In: *Proceedings of Conference on Computer Vision and Pattern Recognition*. Vol. 1. 2005, pp. 1182–1187.

[Tor+03] A. Torralba et al. “Context-based vision system for place and object recognition”. In: *Proceedings of the International Conference on Computer Vision*. 2003, pp. 1023–1029.

[TM01] J. Triesch and Christoph v. d. Malsburg. “Democratic Integration: Self-Organized Integration of Adaptive Cues”. In: *Neural Computation* 13.9 (2001), pp. 2049–2074.

[Tso+05] Ioannis Tsochantaridis et al. “Large margin methods for structured and interdependent output variables”. In: *Journal of Machine Learning Research* 6.Sep (2005), pp. 1453–1484.

[VCC03] Namrata Vaswani, A Roy Chowdhury, and Rama Chellappa. “Activity recognition using the dynamics of the configuration of interacting objects”. In: *Proceedings of the Conference on Computer Vision and Pattern Recognition*. Vol. 2. 2003, pp. II–633.

[Ved+09] Andrea Vedaldi et al. "Multiple kernels for object detection". In: *Proceedings of the International Conference on Computer Vision*. 2009, pp. 606–613.

[VJ01] Paul Viola and Michael Jones. "Rapid object detection using a boosted cascade of simple features". In: *Proceedings of the Conference on Computer Vision and Pattern Recognition*. Vol. 1. 2001, pp. I–511.

[VJ04] Paul Viola and Michael J. Jones. "Robust Real-Time Face Detection". In: *International Journal of Computer Vision* 57.2 (2004), pp. 137–154.

[VM11] Tomáš Vojíř and Jiří Matas. "Robustifying the Flock of Trackers". In: *Proceedings of Computer Vision Winter Workshop*. 2011, pp. 91–97.

[VMN16] Tomas Vojir, Jiri Matas, and Jana Noskova. "Online Adaptive Hidden Markov Model for Multi-Tracker Fusion". In: (2016), pp. –. arXiv: 1504.06103v2 [cs.CV].

[VNM14] Tomas Vojir, Jana Noskova, and Jiri Matas. "Robust Scale-adaptive Mean-shift for Tracking". In: *Pattern Recognition Letters* 49.C (Nov. 2014), pp. 250–258.

[Wan+15a] Lijun Wang et al. "Visual tracking with fully convolutional networks". In: *Proceedings of the International Conference on Computer Vision*. 2015, pp. 3119–3127.

[WY13] Naiyan Wang and Dit-Yan Yeung. "Learning a deep compact image representation for visual tracking". In: *Advances in Neural Information Processing Systems*. 2013, pp. 809–817.

[WY14] Naiyan Wang and Dit Yan Yeung. "Ensemble-based tracking: Aggregating crowdsourced structured time series data". In: *Proceedings of International Conference on Machine Learning*. Vol. 32. 2014, pp. 1107–1115.

[Wan+15b] Naiyan Wang et al. "Transferring rich feature hierarchies for robust visual tracking". In: *arXiv preprint arXiv:1501.04587* (2015).

[Wan+15c] X. Wang et al. "TRIC-track: Tracking by Regression with Incrementally Learned Cascades". In: *Proceedings of the International Conference on Computer Vision*. 2015, pp. 4337–4345.

[WB95] Greg Welch and Gary Bishop. *An Introduction to the Kalman Filter*. Tech. rep. Chapel Hill, NC, USA, 1995.

[Wen+14] Longyin Wen et al. "Robust Online Learned Spatio-Temporal Context Model for Visual Tracking". In: *IEEE Transactions on Image Processing* 23.2 (2014), pp. 785–796.

[Whi+09] Jacob Whitehill et al. "Whose vote should count more: Optimal integration of labels from labelers of unknown expertise". In: *Advances in neural information processing systems*. 2009, pp. 2035–2043.

[WLY13] Y. Wu, J. Lim, and M. H. Yang. "Online Object Tracking: A Benchmark". In: *Proceedings of Conference on Computer Vision and Pattern Recognition*. 2013, pp. 2411–2418.

[XAS15] Yu Xiang, Alexandre Alahi, and Silvio Savarese. "Learning to Track: Online Multi-Object Tracking by Decision Making". In: *Proceedings of the International Conference on Computer Vision*. 2015, pp. 4705–4713.

[YDD05a] Changjiang Yang, Ramani Duraiswami, and Larry Davis. "Efficient mean-shift tracking via a new similarity measure". In: *Proceedings of the Conference on Computer Vision and Pattern Recognition*. Vol. 1. 2005, pp. 176–183.

[YDD05b] Changjiang Yang, Ramani Duraiswami, and Larry Davis. "Fast multiple object tracking via a hierarchical particle filter". In: *Proceedings of the International Conference on Computer Vision*. Vol. 1. 2005, pp. 212–219.

[YWH09] Ming Yang, Ying Wu, and Gang Hua. "Context-aware visual tracking". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.7 (July 2009), pp. 1195–1209.

[Yi+12] Kwang Moo Yi et al. "Visual Tracking with Dual Modeling". In: *Proceedings of the 27th Conference on Image and Vision Computing New Zealand*. IVCNZ '12. 2012, pp. 25–30.

[YJS06] A. Yilmaz, O. Javed, and M. Shah. "Object tracking: A survey". In: *ACM Computing Surveys (CSUR)* 38 (4 Dec. 2006).

[YLS04] Alper Yilmaz, Xin Li, and Mubarak Shah. "Object contour tracking using level sets". In: *Proceedings of the Asian Conference on Computer Vision*. 2004.

[YPC08] Zhaozheng Yin, Fatih Porikli, and Robert T. Collins. "Likelihood Map Fusion for Visual Object Tracking". In: *IEEE Workshop on Applications of Computer Vision*. Jan. 2008, pp. 1–7.

[YDM08] Qian Yu, Thang Ba Dinh, and Gérard Medioni. "Online tracking and reacquisition using co-trained generative and discriminative trackers". In: *Proceedings of the European Conference on Computer Vision*. 2008, pp. 678–691.

[ZMS14] Jianming Zhang, Shugao Ma, and Stan Sclaroff. "MEEM: Robust Tracking via Multiple Experts Using Entropy Minimization". In: *Proceedings of the European Conference on Computer Vision*. Vol. 8694. 2014, pp. 188–203.

[ZZY12] Kaihua Zhang, Lei Zhang, and Ming-Hsuan Yang. "Real-time Compressive Tracking". In: *Proceedings of the European Conference on Computer Vision*. 2012, pp. 864–877.

[Zha+14] Kaihua Zhang et al. "Fast Visual Tracking via Dense Spatio-temporal Context Learning". In: *Proceedings of the European Conference on Computer Vision*. Vol. 8693. 2014, pp. 127–141.

[Zho+14a] Bineng Zhong et al. "Visual tracking via weakly supervised learning from multiple imperfect oracles". In: *Pattern Recognition* 47.3 (2014), pp. 1395–1410.

[ZCM03] Shaohua Zhou, Rama Chellappa, and Baback Moghaddam. "Adaptive visual tracking and recognition using particle filters". In: *Proceedings of the IEEE International Conference on Multimedia and Expo*. Vol. 2. 2003, pp. II–349.

[Zho+14b] Xiangzeng Zhou et al. "An ensemble of deep neural networks for object tracking". In: *Proceedings of the International Conference on Image Processing*. 2014, pp. 843–847.

[ZPL15] Gao Zhu, Fatih Porikli, and Hongdong Li. "Tracking Randomly Moving Objects on Edge Box Proposals". In: *CoRR* (2015).

FOLIO ADMINISTRATIF

THESE DE L'UNIVERSITE DE LYON OPEREE AU SEIN DE L'INSA LYON

NOM : MOUJTAHID DATE de SOUTENANCE : 03/11/2016

Prénoms : Salma

TITRE : « Exploitation du contexte de scène pour le suivi d'objet en ligne dans des environnements non contraints »

NATURE : Doctorat Numéro d'ordre : 2016LYSEI110

Ecole doctorale : INFOMATHS

Spécialité : Informatique

RESUME :

With the increasing need for automated video analysis, visual object tracking became an important task in computer vision. A tracking algorithm in unconstrained environments faces multiple challenges such as changes in object shape, background, lighting, motion, and others.
In this thesis, we start from the observation that different tracking algorithms have different strengths and weaknesses depending on the environments and context. To overcome the varying challenges, we show that combining multiple modalities and tracking algorithms can considerably improve the overall tracking performance in unconstrained environments.
More concretely, we first introduced a new tracker selection framework using a spatial and temporal coherence criterion. In this algorithm, multiple independent trackers are combined in a parallel manner, each of them using low-level features based on different complementary visual aspects like colour, texture and shape. By recurrently selecting the most suitable tracker, the overall system can switch rapidly between different tracking algorithms with specific appearance models depending on the changes in the video.
In the second contribution, the scene context is used to the tracker selection. We designed effective visual features, extracted from the scene context to characterise the different image conditions and variations. At each point in time, a classifier (Neural Network) is trained based on these features to predict the tracker that will perform best under the given scene conditions.
Finally, instead of using manually designed scene context features, we used a Convolutional Neural Network to automatically learn to extract these scene features directly from the input image and predict the most suitable tracker.
The evaluation of the proposed methods has been performed on multiple public benchmarks, and showed that exploiting scene context in this way improves the overall tracking performance.

MOTS-CLÉS: online tracking, machine learning, computer vision, scene context

Laboratoire (s) de recherche : LIRIS – Equipe IMAGINE

Directeur de thèse: Prof. Atilla BASKURT

Président de jury :

Composition du jury :

CHATEAU, Thierry	Prof.	Université Blaise Pascal	Rapporteur
BREMOND, François	DR	INRIA Sophia-Antipolis	Rapporteur
ODOBEZ, Jean-Marc	MER	IDIAP Reseash Institure	Examinateur
BENOIS-PINEAU, Jenny	Prof.	Université Bordeaux 1	Examinatrice
BASKURT, Atilla	Prof.	INSA Lyon	Directeur de thèse
DUFFNER, Stefan	MDC	INSA Lyon	Examinateur