COPPE
Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia UFRJ

DETECTION AND TRACKING OF FACIAL LANDMARKS IN HIGH DEFINITION VIDEO SEQUENCES

Gabriel Matos Araujo

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia Elétrica.

Orientador: Eduardo Antônio Barros da Silva

Rio de Janeiro
Março de 2015

DETECTION AND TRACKING OF FACIAL LANDMARKS IN HIGH DEFINITION VIDEO SEQUENCES

Gabriel Matos Araujo

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Examinada por:

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2015

Araujo, Gabriel Matos

Detection and Tracking of Facial Landmarks in High Definition Video Sequences/Gabriel Matos Araujo. – Rio de Janeiro: UFRJ/COPPE, 2015.

XXII, 144 p.: il.; $29,7$cm.

Orientador: Eduardo Antônio Barros da Silva

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2015.

Referências Bibliográficas: p. 80 – 89.

1. Video Tracking. 2. Face Tracking. 3. Machine Learning. I. Silva, Eduardo Antônio Barros da. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

À minha famlia.

Acknowledgments

A enorme lista de pessoas que preciso agradecer só me mostra o quanto sou um privilegiado. Agradeço a Deus por ter ao meu entorno tantos que me querem bem.

Agradeço a minha família por me apoiar incondicionalmente em todos os momentos e, na maioria das vezes, acreditar em mim mais que eu mesmo. Obrigado Sandra, Carlos (pais), Juliana, Felipe (irmãos), Sônia, Emanoel, Zizi (avós), tios e primos. Eles estão comigo desde o início da minha formação e certamente me ajudaram a realizar esse sonho.

Agradeço também a Meryelle e sua família (representada por D. Rosangela), por me apoiar e me incentivar em vários momentos. Meryelle merece um agradecimento especial por ser uma companheira leal e por ter me dado um presente sem igual: o nosso filho Guilherme. Os dois fizeram com que esse processo fosse muito mais fácil.

A família Molina me recebeu no Rio de Janeiro, quando vim fazer pós-graduação em 2008 e mesmo sem me conhecerem me acolheram como um membro da família. Esse período de transição foi um dos mais difíceis da minha vida, mas eles facilitaram muito esse processo. Tenho imensa gratidão pelo que fizeram.

Reservo um cantinho deste agradecimento para os meus amigos de infância do CODAP. As nossas diversas e intermináveis discussões sobre coisas sérias e, muitas vezes, não tão sérias assim, me ajudaram a suportar mais tranquilamente diversos momentos de dificuldade (mesmo que não soubessem disso). Agradeço ainda a paciência por suportar minhas brincadeiras e opiniões, por vezes pouco ortodoxas.

Meus amigos da UFS, em especial os Professores Eduardo, Jugurta, Lucas, Elyson e Jânio, que ajudaram a despertar em mim o gosto pela carreira acadêmica e sem isso eu não teria chegaria até aqui. Sou grato a eles por isso.

Em 2012 me tornei professor do CEFET/RJ. A partir deste momento percebi o quanto é difcil fazer o doutorado, ministrar aulas e estar envolvido em outras atividades como projetos de pesquisa. Os meus colegas professores e servidores, especialmente do campus de Nova Iguaçu, me ajudaram a alcançar este objetivo.

Meus alunos me deram lições valiosas. Me ensinaram a ser um professor melhor, o que pretendo continuar aprendendo sempre. Além disso, ensinaram que é possível morar em uma cidade, trabalhar em outra e estudar em uma terceira. O comprometimento exigido para completar essa façanha é tamanho, que faz parecer bobagem muitas dificuldades que tenho na minha vida. Muito obrigado por me mostrarem essas duas coisas.

O SMT é um local especial de se trabalhar graças aos seus integrantes. Se por um lado me ajudaram tecnicamente em diversos momentos, por outro foram responsáveis por diversos momentos de descontração quando eles eram necessários (e também quando não eram). Correndo o risco de esquecer alguém, são eles: Alan Tygel, Dr. Alessandro Dutra, Dr. Alexandre Ciancio, Dr. Alexandre Leizor, Allan da Silva, Amanda Loiola, Prof. Amaro Lima, Ana Fernanda, Anderson Oliveira, Dr. Andreas Ellmauthaler, Andressa Sivolella, Balabram Filho, Breno Espíndola, Camila Gussen, Carlos Carlim, Carlos Jr., Catia Valdman, Claudio Verdun, Diego Felix, Prof. Diego Haddad, Eliamara Souza, Prof. Fabiano Castoldi, Dr. Fábio Freeland, Felipe Barboza, Felipe Gonçalves, Felipe Grael, Felipe Ribeiro (Curicica), Fernando Ferreira, Dr. Filipe Diniz, Prof. Flávio Avila, Frederico Wegelin, Hugo Carvalho, Dr. Iker Sobron, Isabela Apolinário, Prof. João Dias, Jonathan Gois, Dr. José Fernando (Zé), Leonardo Nunes (Lonnes), Prof. Lisandro Lovisolo, Lucas Thomaz, Luiz Gustavo (Maestro), Prof. Markus Lima, Maurício da Costa, Prof. Michel Tcheou, Michelle Nogueira, Rafael Amado, Rafael de Jesus, Prof. Rodrigo Peres, Rodrigo Prates, Dr. Rodrigo Torres, Prof. Tadeu Ferreira, Prof. Thiago Prego, Prof. Waldir Sabino, e Prof. Wallace Martins.

Algumas pessoas me ofereceram, além de amizade, conhecimentos técnicos valiosos. Conhecimentos sem os quais este trabalho não teria o mesmo nível. Sou profundamente grato ao Prof. Siome Goldenstein, Prof. Waldir Sabino, Dr. José Fernando e Felipe Ribeiro.

Agradeço ao meu orientador, Prof. Eduardo Antônio Barros da Silva, pela paciência, incentivo e generosidade. O que ele tem me ensinado ao longo destes sete anos vai além da academia. O Prof. Eduardo continua a me surpreender mesmo após este tempo. Apesar da grande quantidade de trabalho que ele está continuamente envolvido, sempre esteve de portas abertas para me receber, mesmo quando não se tratava de assuntos profissionais. Espero continuar fazendo parte, não só do seu círculo profissional, mas também pessoal. Muito obrigado, mestre.

Agradeço também aos membros da banca, Prof. Cláudio Jung, Prof. Hae Yong Kim, Prof. José Gabriel Gomes e Prof. Sergio Netto. Eles ofereceram contribuições valiosas que ajudaram a aumentar a qualidade deste trabalho.

E, como não poderia deixar de ser, agradeço ao povo brasileiro, que por meio de instituições como a CAPES e o CNPQ me financiou durante todos estes anos. E continua a me financiar! Espero poder um dia devolver para a sociedade tudo aquilo que me foi dado.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

DETECÇÃO E RASTREAMENTO DE PONTOS FIDUCIAIS FACIAIS EM SEQUÊNCIAS DE VÍDEO DE ALTA DEFINIÇÃO

Gabriel Matos Araujo

Março/2015

Orientador: Eduardo Antônio Barros da Silva

Programa: Engenharia Elétrica

Nos últimos anos, estamos experimentando uma revolução no campo da interface homem-máquina. Cada vez mais estamos utilizando formas alternativas de interagir com os nossos dispositivos. Neste contexto, a face, ou partes dela, pode(m) ser utilizada(s) em várias aplicações. Por exemplo, o rastreamento da posição dos olhos é importante para manipular conteúdo 3D em dispositivos com tela autoestereoscópica. Neste trabalho, nós propomos uma nova abordagem para detectar e rastrear pontos da face em sequências de vídeo de alta definição. A detecção é baseada no SVM (do inglês *Support Vector Machine*) com *kernel* gaussiano ou no IPD (do inglês *Inner Product Detector*). O método de detecção baseado no IPD é capaz de encontrar os olhos em $88,3\%$ das imagens da BioID com um erro relativo menor que 5% da distância interocular. Nós também propomos duas abordagens para integrar o IPD com o filtro de partícula para rastrear características faciais como as pupilas. Um erro relativo médio de $7,7\%$ da distância interocular foi obtido na nossa base de dados de alta definição. Além disso, propomos um método global para rastrear conjuntamente um grupo de características nos olhos. Neste caso, o rastreamento pode ser realizado pelo algoritmo de Kanade-Lucas (KL) ou pelo filtro de partículas. A novidade deste método consiste na integração da detecção com o rastreamento, na avaliação da consistência temporal, para diminuir a taxa de falsos positivos, e no uso de restrições geométricas para inferir a posição dos pontos faltantes. Nos experimentos, nós usamos as bases de dados BioID e FERET, bem como cinco sequências de vídeo de alta definição. Estas sequências possuem quatro indivíduos, tipos diferentes de plano de fundo, movimentos rápidos, borramento e oclusão. Os resultados obtidos indicam que as técnicas propostas são competitivas e capazes de detectar e rastrear objetos com uma boa confiança.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

DETECTION AND TRACKING OF FACIAL LANDMARKS IN HIGH DEFINITION VIDEO SEQUENCES

Gabriel Matos Araujo

March/2015

Advisor: Eduardo Antônio Barros da Silva

Department: Electrical Engineering

In recent years we are experiencing a revolution in the field of man-machine interfaces. More and more we are using alternative ways to interact with our devices. In this context, the face, or parts of it, can be used in several applications. As an example, tracking the positions of the eyes may be important to manipulate 3D content in devices with autoestereoscopic displays. In this work, we propose a novel approach to detect and track facial landmarks in high definition video sequences. The detection is based on Support Vector Machines (SVM) with Gaussian Kernel or the Inner Product Detector (IPD). Our IPD based detection method can find the eyes in 88.3% of the BioID images with a relative error less than 5% of the interocular distance. We also propose two approaches to integrate the IPD with particle filters in order to track local features such as the pupils. An average relative error of 7.7% of the interocular distance was obtained in our high definition dataset. In addition, we propose a global method to jointly track a set o features on eyes. In this case, the tracking can be performed by the Kanade-Lucas (KL) or the particle filter. The novelty consists in the integration of detection and tracking, the evaluation of the temporal consistency to decrease the false positive rates, and the use of geometrical constraints to infer the position of missing points. In our experiments, we use the BioID and FERET databases as well as five high definition video sequences with four subjects, different types of background, fast movements, blurring and occlusion. The obtained results have shown that the proposed techniques are competitive and are capable of detecting and tracking landmarks with good reliability.

Contents

List of Figures

List of Tables

List of Abbreviations

AAM	Active Appearance Model, p. 3
ASM	Active Shape Model, p. 3
CDF	Cumulative Distributed Function, p. 4
CFA	Class-dependence Filter Analysis, p. 22
CFA	Class-dependent Feature Analysis, p. 19
CF	Correlation Filter, p. 19
CLM	Constrained Local Model, p. 4
CONDENSATION	Conditional Density Propagation, p. 16
DFT	Discrete Fourier Transform, p. 19
DoG	Difference of Gaussians, p. 29
HOG	Histogram of Oriented Gradients, p. 78
IPD	Inner Product Detector, p. 19
KLT	Kanade-Lucas-Tomasi, p. 12
KLTr	Karhunen-Loève Transform, p. 22
LDA	Linear Discriminant Analysis, p. 25
MACE	Minimum Average Correlation Energy, p. 22
MHL	Mingle-Hypothesis Localization, p. 9
MS	Mean Shift, p. 12
PCA	Principal Component Analysis, p. 22
QDA	Quadratic Discriminant Analysis, p. 25

RANSAC	Random Sample Consensus, p. 4
RBF	Radial Basis Function, p. 7
SHL	Single-Hypothesis Localization, p. 9
SIFT	Scale-invariant Feature Transform, p. 4
SMC	Sequential Monte Carlo, p. 13
SNR	Signal to Noise Ratio, p. 19
SVM	Support Vector Machine, p. 3

Chapter 1

Introduction

Currently we are experiencing a crescent demand for mobile systems. This leads to a revolution in the way we interact with our devices. More and more, we are using voice, gestures and expressions to control our gadgets. The last two concur to the problem of locating and tracking an object in video sequences. To give an example, suppose we are trying to develop a system that uses gestures to control a device. The first step of this system can be the detection of the hands. Supposing one hand was correctly detected, we also need to employ some tracking method to obtain the movement of the hand along the frames. Finally, this movement can be coded in a way the machine can interpret it. The same steps can be followed if we want to use the expression or our face to control a machine. First of all, we have to detect the face. Then, we need to detect key points, or landmarks, on it. After that, we can track these points and encode the movement. In most cases we need to use both detection and tracking, because the detection alone can introduce clutter to the output (due to false positive detections). Although the tracking can provide a smoother path, it can diverge in a long term tracking. A possible solution can be integrating them in a way we can get the best of both worlds [1]. In this work, we address the three parts of the problem: detection, tracking and their integration. The objects of interest are landmarks on the face.

In the detection part, we use two different classifiers. One is the well known Support Vector Machine (SVM). The second is the Inner Product Detector (IPD), a Correlation Filter based detector [2–4]. This detector is a weak classifier, what means that it is weakly correlated with the desired pattern. Fortunately, it is known that it is possible obtain a strong classifier by combining weak classifiers. In the case of the IPD a cascade has been used. In this context, we have three main contributions [3, 5, 6]: (i) we conducted several experiments in order to verify the effect of the detector's size in the performance; (ii) the discriminant function is more restrictive than the ones in similar linear classifiers - a desirable feature in cases such as feature/object detection in images; (iii) the stop criterion used to automatically determine the number of stages of the cascade. Our detection method can find the eyes in 88.3% of the BioID images with an error of less than 5% of the interocular distance.

We also propose some combinations between detection and tracking methods to perform landmarks tracking in video sequences. In one of them, we used the IPD to find the probability distribution function of desired features in images. There are two variations of such method, one of it is parametric and the other nonparametric. They are simple, fast and fit in a Bayesian framework, such as the Kalman or particle filter, for tracking the features in video sequences. When using the IPD as a nonparametric method for estimating the probability distribution function of the feature, we obtained an average relative error of 7.7% of the interocular distance in our high definition dataset. In other contribution, we combine the detector with two different trackers: the Kanade-Lucas and the Particle Filter. In this context, we propose an integration method to globally track a set of features on the eyes [7, 8]. This integration is, first of all, based on the analysis of the histogram of the distances between the detected and tracked points. In addition, we reduce the false positive rates by verifying the temporal consistency of the obtained points. Although at this step we have a high hit rate and good precision, there are missing points in several frames, leading to high false negative rates. To address this issue, we use the geometry of the face in order to estimate the missed points. This global method performs well for easy and intermediate/difficult sequences.

The remainder of this work is organized as follows: In Chapter 2 we provide a review of the state of the art in three parts. In the first part we describe some previous works in this area. The second part contains a description of the SVM. In the last part, we describe the state of the art in video tracking, focusing on Bayesian inference and particle filters. After these chapters, we present our contributions. In Chapter 3, we describe the IPD and in Chapter 4 we introduce our methods to combine detection and tracking. Finally, we conclude and discuss some alternatives for future works in Chapter 5.

Chapter 2

A Review of the State-of-the-Art Algorithms

The problem of locating and/or tracking facial features has been investigated in recent years. Features on human faces are useful in many applications, such as biometrics, expression recognition [9], face alignment, pose estimation [10], 3D face modeling [11, 12], and face tracking.

Eye detection and tracking, in particular, can be used in many applications as attention and gaze estimation for disabled people (in order to control assistive devices) [13, 14], driver fatigue detection [15, 16], augmented reality systems [17], biometrics and so on [18]. There are different ways to locate and track the position of the eyes [18, 19]. However, most methods in the literature are based on computer vision techniques. Some of them use the dark and bright pupil effect to track the eyes [13]. It is common to use active infrared (IR) illuminations, which force the pupil to appear brighter than the rest of the image [20–22]. These methods can achieve good accuracy, but can not be used in daylight applications and require dedicated hardware [23].

The methods to locate facial fiducial points can be divided in two groups: the group of global methods and the group of local ones [24]. The first group uses global features, such as texture or edge, to locate the fiducial points. The main characteristic of global methods is the capability of robustly locating several fiducial points simultaneously and the main drawback is their computational cost. Well known examples of global methods are Active Shape Models (ASM) [25] and Active Appearance Models (AAM) [26]. The ASM consists in searching the best match between the image and a deformable shape model. In the AAM the model is a combination of texture and shape. In [24], for example, the minimization step of the ASM is treated as a classification step and a Support Vector Machine (SVM) [27] is used to match the template and the image. In [28], to improve the robustness of the AAM, new constraints based on temporal matching and color are introduced in the objective function. Similar methods to AAM, as the one in [29], can also be found in the literature. In this last one, Cristinace and Cootes proposed a method called Constrained Local Model (CLM). The main differences between them are that the CLM learns the variation in the appearance of the regions around the features and they use a different search algorithm.

In local methods, each fiducial point is detected independently, using only local information. The main advantage of local methods is the low computational cost. There are several examples of local methods, most of them using a cascade of boosted classifiers. In [30], for example, Gabor filters are used to extract features to be classified by a type of cascade of classifiers. In [31], dot products between gradient and displacement vectors are used in an objective function whose maximum gives a good approximation of the eyes's centers. In [32], a set of isophote[1] centers with high values is used in series with a Scale-invariant Feature Transform (SIFT) [33] based matching and a kNN classifier to locate the center of the eye. At the same accuracy, the method proposed by [32] reaches a little bit higher hit rate than the one proposed by [31] but with increased computational cost. In [34], an adaptive Cumulative Distributed Function (CDF) analysis is used to locate the pupils. In [35], the eyes are located by means of projection functions. These methods have one feature in common: they do not have any type of learning scheme, so they cannot be generalized to detect other important fiducial points on face. A technique that uses eye area edge map comparison can be found in [36].

Recent works combine both local and global methods to improve the accuracy. In [37], Haar filters are used to extract the features. The global part of this method uses a mixture of Gaussians to infer the joint position the features. Other remarkable example is the one in [38]. It uses a combination of SIFT and SVM as local features. Then, they refine the detection using a global method. The global part of their work consists in an objective function, defined by a Bayesian model, optimized by a Random Sample Consensus (RANSAC) [39] based algorithm. They reach good results, but their method is too computationally demanding to be used in online applications.

All the methods presented above are capable of performing detection of multiple objects in static images. However, it is possible to use object detectors frame-by-frame in a video sequence and then employ some tracking method to find an association between the detected objects across the frames. This approach is known as tracking-by-detection and have been used in several works recently [1, 7, 40–45]. The main challenge of such approaches is that the detectors usually present false positives and false negatives at their outputs. In other words, the output of the

[1]An isophote is a curve with constant intensity. In this case, the luminance is used as an intensity measure.

detectors is usually unreliable and sparse [46]. Tracking-by-detection methods have been used in many applications [1, 47], but have become particularly popular in pedestrian tracking [45, 46].

In [47], Kalal et al. proposed a method to detect and track an unknown object defined by the user in a single frame. The detection step is performed by a sequential randomized forest, that is trained on-line to evaluate a set of 2-bit binary patterns. The output of the detection feeds a Kanade-Lucas (KL) tracker, that is based on optical flow [48]. In a more recent work [1], Kalal et al. improved the integration between the detection and tracking and proposed a robust algorithm called Predator. They developed a learning method, called P-N learning, that is capable of estimating the missing detections (false negatives) and the false alarms (false positives). This learning scheme is used on-line, so the reliability of the detections and consequently of the tracking are improved over time.

There are several multi-object tracking-by-detection algorithms employed in pedestrian tracking [40, 42, 45, 46, 49–53]. Some of them apply a temporal delay in order to use the information from future frames to improve robustness [42, 45, 50, 51, 53]. Others are concerned with online applications and adopted Markov assumptions, considering only information from past frames [40, 46, 49, 52]. The last ones used the detection in the particle filter framework.

Tracking-by-detection methods also have been used with success to track facial landmarks in video sequences. As a matter of fact, some methods described at the beginning of this section have been used in conjunction with tracking methods. In [32], for example, the eyes are tracked by employing frame-by-frame detection in a video from a webcam. In [28], the features are detected on each frame through AAM and two constraints are employed to follow them across the frames. One is the temporal matching between local features of successive frames. The other is a segmentation based on facial color. In [37], before performing the facial feature localization, they track the faces by employing the Viola-Jones's face detector [54] in each frame and then some heuristics based on the Kanade-Lucas-Tomasi tracker [55] to find the temporal connection between the detected faces. Cristinacce and Cootes [29] also applied their facial feature detectors (the CLM, as can be seen earlier in this section) frame-by-frame in order to track facial features on a video sequence. They do not try to find a temporal correlation across the frames, but claim that it is not necessary, since the template learns the appearance of the features and is constrained by a joint shape and texture model. In [56], the facial features are detected in the first frame through SIFT based detectors and the tracking is performed along the frames by employing Multiple Differential Evolution-Markov Chain (DE-MC) particle filters.

The use of a probabilistic distribution for the representation of a feature's localization leads naturally to Bayesian tracking and information fusion pipelines. For instance, a Bayesian method enables learning an underlying distribution for the position of landmarks using visual features [57], an unscented particle filter can propagate local points of interest [58], and multiple object detectors in different cameras can estimate the joint detected distribution in world space [59].

2.1 Support Vector Machine

The Support Vector Machine (SVM) is a machine learning technique, introduced by Vapnik [27], that has been used in regression and classification problems [27, 60]. In the two-class classification problem, we aim at obtaining a separating hyperplane that splits the space in two parts. The hyperplane is defined in terms of the nearest samples of each class. These samples are called support vectors and the distance between them is called margin. In other words, the SVM provides the hyperplane, H, that maximizes the margin. If the problem is linearly tractable, this hyperplane can be described as follows:

$$f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + b, \tag{2.1}$$

where $\mathbf{w}$ is orthogonal to the hyperplane H, $|b|$ is the distance between the hyperplane and the origin ($\|\mathbf{w}\| = 1$) and $[\cdot]^{\mathrm{T}}$ denotes the transpose.

Since in most cases the problem is not linearly separable, we must introduce a tolerance variable ξ_n that allows some samples to be in the margin region or even in the region of the other class. So, the hyperplane equation can be rewritten as

$$y_n(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n + b) - 1 + \xi_n \geq 0, \tag{2.2}$$

where $\mathbf{x}_n$ is a training sample and y_n is its label. If $0 \leq \xi_n \leq 1$, the sample is in the margin and if $\xi_n > 1$ the sample is in the region of the other class. This formulation is known as soft margin and is illustrated in Figure 2.1. Note that the hyperplanes H_{-1} and H_{+1}, the bounds of the margin, are parallel to the separating hyperplane H.

The optimal separating hyperplane can be obtained by solving the following optimization problem:

$$\underset{\mathbf{w},b,\boldsymbol{\xi}}{\arg\min} \left[\frac{1}{2}\|\mathbf{w}\|^2 + C \left(\sum_{n=1}^{N} \xi_n \right) \right], \tag{2.3}$$

subject to

$$y_n(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n + b) - 1 + \xi_n \geq 0, \qquad \xi_n \geq 0, \qquad \forall n, \tag{2.4}$$

where C is a free parameter that defines a penalty for the misclassification and $\boldsymbol{\xi}$ is

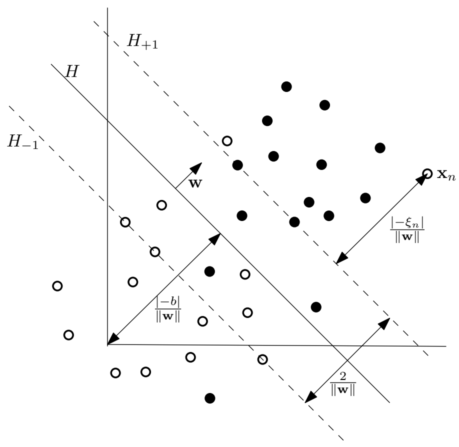

H_{+1}
H
H_{-1}
$\mathbf{w}$
$\mathbf{x}_n$
$\frac{|-\xi_n|}{\|\mathbf{w}\|}$
$\frac{|-b|}{\|\mathbf{w}\|}$
$\frac{2}{\|\mathbf{w}\|}$

Figure 2.1: Linear SVM with soft margin.

a set composed by N elements ξ_n.

The Equations (2.3) and (2.4) stand for a quadratic optimization problem with linear constraints. Adopting a Lagrangian solution, the optimization problem can be rewritten as [27]

$$L_d = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j), \tag{2.5}$$

subject to

$$0 \leq \alpha_n \leq C, \tag{2.6}$$

$$\sum_{n=1}^{N} \alpha_n y_n = 0, \tag{2.7}$$

where L_d is the dual objective function, α_i is a Lagrange multiplier and $K(\mathbf{x}_i, \mathbf{x}_j)$ denotes a kernel function over the samples. A kernel function returns the inner product between two samples in a higher (possibly infinite) dimensional space, where the classes can be linearly tractable [60]. There are several types of kernel functions, such as linear, polynomial, sigmoidal and Gaussian. The most used are the linear and the Gaussian (also known as Radial Basis Function (RBF)) kernels. Their definitions are in Table 2.1.

Table 2.1: Most used kernel functions. For the Gaussian, $\gamma = 1/2\sigma^2$ and σ is the standard deviation.

	Function $k(\mathbf{x}, \mathbf{y})$
Linear	$\mathbf{x}^{\mathrm{T}}\mathbf{y}$
Gaussian (RBF)	$\exp\left(-\gamma\|\mathbf{x}-\mathbf{y}\|^2\right)$

Including the Karush-Kuhn-Tucker conditions as constraints

$$\begin{aligned} \alpha_n \left[y_n \mathbf{x}_n \mathbf{w} + b - 1 + \xi_n \right] &= 0, && (2.8) \\ \mu_n \xi_n &= 0, && (2.9) \\ y_n \mathbf{x}_n \mathbf{w} + b - 1 + \xi_n &\geq 0, && (2.10) \end{aligned}$$

for $n = 1, \cdots, N$, we can uniquely determine the solution to the optimization problem in the Equations (2.5), (2.6) and (2.7) [60].

The resulting discriminant functions is

$$g(\mathbf{x}) = \text{sign}\left(\sum_{i \in SV} y_i \alpha_i^* k(\mathbf{x}_i, \mathbf{x}) + b^* \right), \qquad (2.11)$$

where

$$b^* = \frac{1}{n_{\tilde{SV}}} \sum_{i \in \tilde{SV}} \left(\frac{1}{y_i} - \sum_{j \in SV} \alpha_j^* y_j k(\mathbf{x}_j, \mathbf{x}_i) \right), \qquad (2.12)$$

α_n^* and b^* are optimal solutions, SV is a set composed by the support vectors that satisfy $0 < \alpha_n < C$ and $n_{\tilde{SV}}$ is the number of elements in this set. It is important to emphasize that $\alpha_n^* \geq 0$ only for the support vectors. If $\alpha_n^* < C$, then $\xi_n^* = 0$ and the vector is over the margin. If $\alpha_n^* = C$, we have two cases: if $0 < \xi_n^* \leq C$ the vector is in the margin; the vector is over the margin if $\xi_n^* = 0$.

A more detailed description of the SVM can be found in several texts as [27, 60–64].

2.2 Video Tracking Using State Estimation

Suppose we want to measure time-varying variables related to a physical process. Recognizing these variables as state variables, we can use state estimation techniques to do so. These techniques are based on a state space model and on a measurement model. The state space model provides the link between the states of this process and the measurement model describes the relation between the states and the data obtained from the sensory system.

There are several applications of state estimation, but we are interested in object tracking in a video sequence. In this case, the variables of interest, or states, are velocity and position of a moving object. We also need an online estimation framework, that comprises the state space model, measurement models and optimal estimation. The state space model provides a link between the states in consecutive frames. The measurements are related to the two-dimensional position of the object in a frame of the video sequence. So, the measurement model provides the link between these measurements and the states. The frames of a video sequence can be represented by discrete time indexes and variables like position and velocity are real-valued vectors. That is the reason we only describe the case of discrete-time processes of continuous states. For more details, including discrete state variables, refer to [65].

The framework described above can be used in two scenarios: single and multiple-hypothesis localization [66]. In Single-Hypothesis Localization (SHL) methods, only one tracker candidate is estimated and evaluated any time. Gradient based methods, such as Kanade-Lucas-Tomasi, Mean Shift and Kalman filter are examples of SHL methods. In Multiple-Hypothesis Localization (MHL) methods, multiple tracker candidates are evaluated simultaneously. Grid Sampling and Particle Filters are examples of methods in this category.

A general framework for online estimation is described in Section 2.2.1. In Section 2.2.2, we give some details of SHL methods focusing on Kanade-Lucas-Tomasi tracker. In Section 2.2.3, we describe the Particle Filter, an MHL method.

2.2.1 A Framework for Online Estimation

We can divide the estimation problem in three parts: online estimation, prediction and retrodiction. In online estimation, also known as optimal filtering, we estimate the current state using all previous measurements. In prediction, we estimate the future states. The retrodiction, also known as smoothing or offline estimation, consists in estimating the past states.

State space model

Let $X = \mathbb{R}^M$ be a state space and $\mathbf{x}(n) \in X$ a state at a discrete time $n \in \mathbb{Z}$. Suppose that we know the state of a process from the beginning up to the present $(\mathbf{x}(0), \mathbf{x}(1), \cdots, \mathbf{x}(n))$ and want to estimate the next state $\mathbf{x}(n+1)$.

Modeling our states as random variables, an optimal estimate of $\mathbf{x}(n+1)$ can be found if we evaluate the conditional density $p(\mathbf{x}(n+1)|\mathbf{x}(0), \mathbf{x}(1), \cdots, \mathbf{x}(n))$. This problem can be simplified if we suppose that the problem is a first order Markov chain, that is, if we assume that the probability of $\mathbf{x}(n+1)$ depends only on $\mathbf{x}(n)$ and not on past states. If this assumption holds, then the conditional density is:

$$p(\mathbf{x}(n+1)|\mathbf{x}(0), \mathbf{x}(1), \cdots, \mathbf{x}(n)) = p(\mathbf{x}(n+1)|\mathbf{x}(n)). \tag{2.13}$$

The probability density $p(\mathbf{x}(n+1)|\mathbf{x}(n))$ is known as transition probability density. Knowing it and the a priori $p(\mathbf{x}(0))$, we can determine the probability density

$p(\mathbf{x}(n+1))$ at any time by recursively computing

$$p(\mathbf{x}(n+1)) = \int_{\mathbf{x}(n)\in X} p(\mathbf{x}(n+1)|\mathbf{x}(n))p(\mathbf{x}(n))d\mathbf{x}(n) \qquad \text{for } n = 0, 1, \cdots \tag{2.14}$$

In addition, the joint probability of the sequence $\mathbf{x}(0), \mathbf{x}(1), \cdots, \mathbf{x}(n)$ is given by

$$p(\mathbf{x}(0), \mathbf{x}(1), \cdots, \mathbf{x}(n)) = p(\mathbf{x}(0)) \prod_{j=1}^{n} p(\mathbf{x}(j)|\mathbf{x}(j-1)). \tag{2.15}$$

Measurement model

Let $Z = \mathbb{R}^N$ be a measurement space and $\mathbf{z}(n) \in Z$ a measurement at the discrete time n. The sequence of measurements until the present is $\mathbf{Z}(n) = \{\mathbf{z}(0), \cdots, \mathbf{z}(n)\}$ and the model of the sensory system is defined by the conditional probability density $p(\mathbf{z}(n)|\mathbf{x}(0), \cdots, \mathbf{x}(n), \mathbf{z}(0), \cdots, \mathbf{z}(n-1))$. Assuming that our sensory system is memoryless, we can rewrite the probability density of the measurements as

$$p(\mathbf{z}(n)|\mathbf{x}(0), \cdots, \mathbf{x}(n), \mathbf{Z}(n-1)) = p(\mathbf{z}(n)|\mathbf{x}(n)). \tag{2.16}$$

Optimal online estimation

In online estimation we start with a prior distribution $p(\mathbf{x}(0))$ at the discrete time index $n = 0$. Then, we can receive the first measurement $\mathbf{z}(0)$ and obtain the posterior density $p(\mathbf{x}(0)|\mathbf{Z}(0))$. After that, at each time index n, we can obtain the posterior density $p(\mathbf{x}(n)|\mathbf{Z}(n))$ by means of the measurement $\mathbf{z}(n)$ and derive an estimate $\hat{\mathbf{x}}(n)$ by using this posterior function. This posterior density is updated, at each time n, through a recursion as depicted in Figure 2.2.

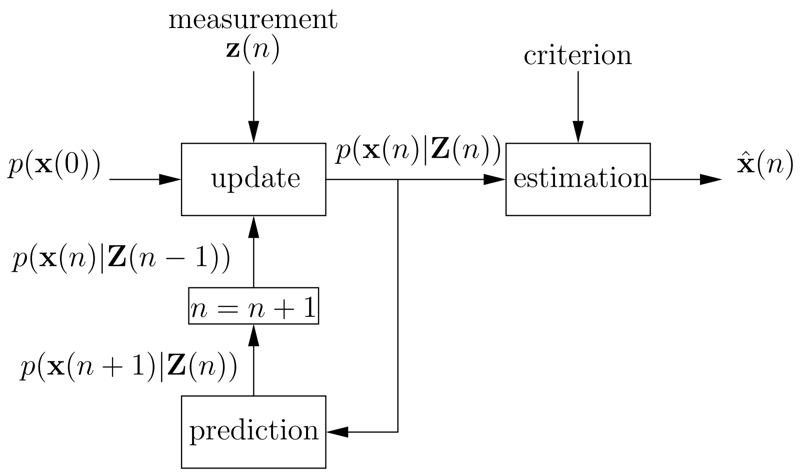


Figure 2.2: A general scheme for online estimation.

The first step of this recursion is the prediction, where we can obtain the density

$p(\mathbf{x}(n+1)|\mathbf{Z}(n))$. To do so, we apply Bayes' theorem combined with the Markov condition as follows:

$$\begin{aligned} p(\mathbf{x}(n+1)|\mathbf{Z}(n)) &= \int_{\mathbf{x}(n)\in X} p(\mathbf{x}(n+1),\mathbf{x}(n)|\mathbf{Z}(n))d\mathbf{x}(n) \\ &= \int_{\mathbf{x}(n)\in X} p(\mathbf{x}(n+1)|\mathbf{x}(n),\mathbf{Z}(n))p(\mathbf{x}(n)|\mathbf{Z}(n))d\mathbf{x}(n) \\ &= \int_{\mathbf{x}(n)\in X} p(\mathbf{x}(n+1)|\mathbf{x}(n))p(\mathbf{x}(n)|\mathbf{Z}(n))d\mathbf{x}(n). \qquad (2.17) \end{aligned}$$

The last step of the recursion is the update, where we increment the variable n so that $p(\mathbf{x}(n+1)|\mathbf{Z}(n))$ becomes $p(\mathbf{x}(n)|\mathbf{Z}(n-1))$ and another measurement $\mathbf{z}(n)$ is collected in order to obtain the next posterior density $p(\mathbf{x}(n)|\mathbf{Z}(n))$. Since our sensory system is memoryless, we can use the Bayes' theorem to obtain the following density function:

$$\begin{aligned} p(\mathbf{x}(n)|\mathbf{Z}(n)) &= p(\mathbf{x}(n)|\mathbf{Z}(n-1),\mathbf{z}(n)) \\ &= \frac{1}{c}p(\mathbf{z}(n)|\mathbf{x}(n),\mathbf{Z}(n-1))p(\mathbf{x}(n)|\mathbf{Z}(n-1)) \\ &= \frac{1}{c}p(\mathbf{z}(n)|\mathbf{x}(n))p(\mathbf{x}(n)|\mathbf{Z}(n-1)), \qquad (2.18) \end{aligned}$$

where c is a normalization constant given by

$$c = \int_{\mathbf{x}(n)\in X} p(\mathbf{z}(n)|\mathbf{x}(n))p(\mathbf{x}(n)|\mathbf{Z}(n-1))d\mathbf{x}(n). \qquad (2.19)$$

The recursion using the Equations (4.1), (2.18) and (2.19) is known as Bayes, or Predictive, filtering [66, 67]. Since it provides the posterior density in each iteration, we just need an optimality criterion to obtain an optimal estimate of the current state [65]. If the state and measurement spaces are discrete, the integrals turns into summations and this recursion becomes a powerful tool. As a matter of fact, it is difficult to implement this recursion for the continuous case, since it requires good representations for the density functions with N and M dimensions and efficient algorithms for the integrations over an M-dimensional space [65]. However, under known constraints the posterior function can be simplified. The Kalman filter is a classical example – it is the optimal solution when both state and measurement models are linear with Gaussian noises [65–67]. In other cases, the state and measurement models are linear, but the distributions of the states are not Gaussian. A remarkable approximation for this type of problem is the particle filtering, which implements a recursive Bayesian filter through Monte-Carlo simulations [65–67]. A brief description of the particle filter is provided in Section 2.2.3.

2.2.2 Single-Hypothesis Localization

In Single-Hypothesis Localization, only one state candidate is estimated and evaluated at each time index n. There are several trackers in this category, such as Kanade-Lucas-Tomasi (KLT) tracker, Mean shift (MS) tracker and Kalman filter. In this section, we discuss one of the most used trackers in this category: the KLT tracker, a gradient-based technique.

Kanade-Lucas-Tomasi Tracker

The (KLT) [55] tracker uses a Newton-Raphson minimization procedure to compute affine changes between images. In addition, this tracker also uses a set of features that can improve the tracker's accuracy. The KLT tracker is based on a technique proposed by Lucas and Kanade [48], that can compute translational changes using a Newton-Raphson minimization procedure.

Let $I_T(\cdot)$ be a squared window template of size $(2W-1)\times(2W-1)$ aligned with the coordinate system of the image I_n, $\forall n$. Defining the state $\mathbf{x}(n)$ as a pixel location in the image and $\tilde{\mathbf{x}}(n)$ the center of the template, the displacement between the template and a square target in the image can be given by

$$\mathbf{x}(n) = \tilde{\mathbf{x}}(n) + \Delta\mathbf{x}(n). \tag{2.20}$$

Using the estimated displacement $\mathbf{x}(n-1)$ at the time $n-1$ as the initial position of the target $\tilde{\mathbf{x}}(n)$, the term $\Delta\mathbf{x}(n)$ can be interpreted as a small displacement added to the previous displacement. Adopting the constant-illumination constraint, we can write the relationship between the template and the window centered at the state $\mathbf{x}(n)$ as

$$I_n(\mathbf{w}) = I_T(\mathbf{w}-\mathbf{x}(n)) + \nu_n(\mathbf{w}) = I_T(\mathbf{w}-\tilde{\mathbf{x}}(n)-\Delta\mathbf{x}(n)) + \nu_n(\mathbf{w}), \tag{2.21}$$

where $\mathbf{w}$ is a pixel location, $\nu_n(\mathbf{w})$ is an additive noise, $|\mathbf{w}-\mathbf{x}(n)|_1 < W$ and $|\cdot|_1$ denotes the L_1 norm.

So, tracking the template in the image consists in obtaining the small displacement that minimizes the error between the window around the state $\mathbf{x}(n)$ and the template. The error can be defined as

$$\epsilon(\Delta\mathbf{x}(n)) = \sum_{|\mathbf{w}-\mathbf{x}(n)|_1<W} \left[I_T(\mathbf{w}-\tilde{\mathbf{x}}(n)-\Delta\mathbf{x}(n)) - I_n(\mathbf{w})\right]^2. \tag{2.22}$$

If the displacement is small, we can rewrite the template $I_T(\cdot)$ through the linear terms of its Taylor series:

$$I_T(\mathbf{w} - \tilde{\mathbf{x}}(n) - \Delta\mathbf{x}(n)) \approx I_T(\mathbf{w} - \tilde{\mathbf{x}}(n)) + \mathbf{b}^{\mathrm{T}}\Delta\mathbf{x}(n), \tag{2.23}$$

where $\mathbf{b} = \frac{\partial I_T(\mathbf{w}-\tilde{\mathbf{x}}(n))}{\partial \mathbf{w}}$ and $[\cdot]^{\mathrm{T}}$ denotes the transpose. Substituting the Equation (2.23) in the Equation (2.22), we have

$$\epsilon(\Delta\mathbf{x}(n)) = \sum_{|\mathbf{w}-\mathbf{x}(n)|_1 < W} \left[I_T(\mathbf{w} - \tilde{\mathbf{x}}(n)) - I_n(\mathbf{w}) - \mathbf{b}^{\mathrm{T}}(\mathbf{w})\Delta\mathbf{x}(n)\right]^2. \tag{2.24}$$

Now, we can obtain the displacement that minimizes the errors by just applying

$$\frac{\partial \epsilon(\Delta\mathbf{x}(n))}{\partial \Delta\mathbf{x}(n)} = 0. \tag{2.25}$$

So, the displacement is

$$\Delta\mathbf{x}(n) = \frac{\sum_{|\mathbf{w}-\mathbf{x}(n)|_1 < W} \left[I_T(\mathbf{w} - \tilde{\mathbf{x}}(n)) - I_n(\mathbf{w})\right] \mathbf{b}(\mathbf{w})}{\sum_{|\mathbf{w}-\mathbf{x}(n)|_1 < W} \mathbf{b}^{\mathrm{T}}(\mathbf{w})\mathbf{b}(\mathbf{w})}. \tag{2.26}$$

Although this formulation can lead to a local minimum, we can approach the global one by using it in an iterative algorithm. In this case, we just have to replace $\tilde{\mathbf{x}}(n)$ by $\tilde{\mathbf{x}}(n) + \Delta\mathbf{x}(n)$ in each step until convergence is achieved. This technique works well for pure translational movement. However, as can be seen in [55], it can be extended to incorporate more complex movements, such as affine transformations.

2.2.3 Multiple-Hypothesis Localization and the Particle Filter

In Multiple-Hypothesis Localization (MHL), multiple state candidates are evaluated simultaneously at each time index n. The state space and measurement models are used to validate these states, or hypotheses. The most likely states are propagated while the unlikely ones are removed. Multiple hypothesis methods can deal better with multi-modal density functions, that is the type of distribution we obtain when dealing with occlusion, noisy environments, clutter and so on. The main drawback of MHL methods is the relationship between the computational cost and the dimensionality of the state space. This is so because the number of hypotheses that are necessary to explore multi-dimensional state spaces grows exponentially with the number of dimensions [66]. Currently, one of the most popular MHL methods is particle filter. Actually, the particle filter is a family of estimators that try an approximation of the Bayes' tracker by using Monte Carlo methods. That is why it is also known as Sequential Monte Carlo (SMC) method. The key idea of the particle filter is to use random samples, or randomly selected observations, to represent the prior probability density. So, it can deal with nonlinearity and non-Gaussian systems [65, 66]. Following the description in [65], we provide more details about the particle filter in the remaining of this section.

Importance Sampling

Monte Carlo simulation is a technique for estimating the expectation of any function for a given distribution by using a set of random samples drawn from that distribution. Mathematically, if $\mathbf{x}^k$, $k = 1, \cdots, K$, are samples drawn from the conditional density $p(\mathbf{x}|\mathbf{z})$, then the expectation of a function $g(\mathbf{x})$ can be estimated by

$$E\left[g(\mathbf{x})|\mathbf{z}\right] \approx \frac{1}{K}\sum_{k=1}^{K} g(\mathbf{x}^k). \tag{2.27}$$

This approximation asymptotically tends to the expectation as K increases.

In particle filtering context, we draw a set of samples $\mathbf{x}^k(n)$ in each time index n. These samples, called particles, are used to represent the posterior density $p(\mathbf{x}(n)|\mathbf{Z}(n))$. The problem is that we do not know this density in advance. So, we have to use another distribution $q(\mathbf{x})$, called proposal density, to get the samples. As a matter of fact, the choice of the proposal density is one of the differences between some types of particle filters. Using the proposal density, the expectation of $g(\mathbf{x})$ with respect to $p(\mathbf{x}|\mathbf{z})$ can be obtained as follows:

$$\begin{aligned} E\left[g(\mathbf{x})|\mathbf{z}\right] &= \int g(\mathbf{x})p(\mathbf{x}|\mathbf{z})d\mathbf{x} \\ &= \int g(\mathbf{x})\frac{p(\mathbf{x}|\mathbf{z})}{q(\mathbf{x})}q(\mathbf{x})d\mathbf{x}. \end{aligned} \tag{2.28}$$

Applying the Bayes' theorem in the equation above, we have:

$$E\left[g(\mathbf{x})|\mathbf{z}\right] = \int g(\mathbf{x})\frac{p(\mathbf{z}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{z})q(\mathbf{x})}q(\mathbf{x})d\mathbf{x}. \tag{2.29}$$

Defining

$$w(\mathbf{x}) = \frac{p(\mathbf{z}|\mathbf{x})p(\mathbf{x})}{q(\mathbf{x})}, \tag{2.30}$$

we can rewrite the expectation as

$$E\left[g(\mathbf{x})|\mathbf{z}\right] = \frac{1}{p(\mathbf{z})}\int g(\mathbf{x})w(\mathbf{x})q(\mathbf{x})d\mathbf{x}. \tag{2.31}$$

Where $p(\mathbf{z})$ is a normalizing factor that can be expressed as

$$\begin{aligned} p(\mathbf{z}) &= \int p(\mathbf{z}|\mathbf{x})p(\mathbf{x})d\mathbf{x} \\ &= \int \frac{p(\mathbf{z}|\mathbf{x})p(\mathbf{x})}{q(\mathbf{x})}q(\mathbf{x})d\mathbf{x} \\ &= \int w(\mathbf{x})q(\mathbf{x})d\mathbf{x}. \end{aligned} \tag{2.32}$$

Then, the expectation becomes:

$$E\left[g(\mathbf{x})|\mathbf{z}\right] = \frac{\int g(\mathbf{x})w(\mathbf{x})q(\mathbf{x})d\mathbf{x}}{\int w(\mathbf{x})q(\mathbf{x})d\mathbf{x}}. \tag{2.33}$$

The expectation above can be estimated by means of samples drawn from $q(\mathbf{x})$ using the following equation:

$$E\left[g(\mathbf{x})|\mathbf{z}\right] \approx \frac{\sum_{k=1}^{K} w(\mathbf{x}^k(n))g(\mathbf{x}^k(n))}{\sum_{k=1}^{K} w(\mathbf{x}^k(n))}. \tag{2.34}$$

The term $w(\mathbf{x}^k(n))$ in the equation above is also known as unnormalized importance weights. By defining the normalized importance weights as

$$w_{\text{norm}}(\mathbf{x}^k(n)) = \frac{w(\mathbf{x}^k(n))}{\sum w(\mathbf{x}^k(i))}, \tag{2.35}$$

we can simplify the Equation (2.34) as

$$E\left[g(\mathbf{x})|\mathbf{z}\right] \approx \sum_{k=1}^{K} w_{\text{norm}}(\mathbf{x}^k(n))g(\mathbf{x}^k(n)). \tag{2.36}$$

To summarize, in the importance sampling, each sample $\mathbf{x}^k(n)$ has a weight $w_{\text{norm}}(\mathbf{x}^k(n))$ and together they represent the conditional density $p(\mathbf{x}|\mathbf{z})$.

As can be seen in [65], the Equations (2.34) and (2.36) are biased estimates. Fortunately, under mild conditions and for $q(\mathbf{x})$ overlapping the support of $p(\mathbf{x})$, the estimate tends asymptotically to the expectation as K increases [65].

Resampling by Selection

Since each sample $\mathbf{x}^k(n)$ has a weight $w_{\text{norm}}(\mathbf{x}^k(n))$, we can use this information to select a new set of the samples to obtain a more accurate representation of $p(\mathbf{x}|\mathbf{z})$. This is equivalent to discarding the samples with low weight and keeping multiple copies of those samples with larger weights (maintaining the number of particles K constant). This procedure is known as selection and different types of particle filters use different types of selection strategies.

One of the most used methods employs a multinomial distribution with the probabilities given by the importance weights and draws samples with replacement from it. To do so, the first step is the calculation of the cumulative weights as follows:

$$w_{\text{cum}}(\mathbf{x}^k(n)) = \sum_{j=1}^{k} w_{\text{norm}}(\mathbf{x}^k(n)). \tag{2.37}$$

Then, we generate r^k, $k = 1, \cdots, K$ random numbers between 0 and 1 from an uniform distribution. The new set $\mathbf{x}^k_{\text{selec}}(n)$ is a copy of the sample $\mathbf{x}^j(n)$, where j is the smallest sample for which $w^j_{\text{cum}} \geq r^k$.

CONDENSATION Algorithm

The Conditional Density Propagation (CONDENSATION) algorithm is a combination of importance sampling and resampling by selection [68]. In terms of the general scheme in Figure 2.2, we start with the prediction of the previous iteration $(n-1)$: the predicted density $p(\mathbf{x}(n)|\mathbf{z}(n-1))$ is used as the proposal $q(\mathbf{x})$. At the current iteration (n), the set $\mathbf{x}^k(n)$ is used as an unweighted representation of the previously predicted density. The next step is to find the posterior density $p(\mathbf{x}(n)|\mathbf{z}(n))$ by employing the importance sampling. To do so, we use Equation (2.29) with the following substitutions:

$$\begin{aligned}
p(\mathbf{x}) &\rightarrow p(\mathbf{x}(n)|\mathbf{Z}(n-1))\\
p(\mathbf{x}|\mathbf{z}) &\rightarrow p(\mathbf{x}(n)|\mathbf{z}(n), \mathbf{Z}(n-1)) = p(\mathbf{x}(n)|\mathbf{Z}(n))\\
q(\mathbf{x}) &\rightarrow p(\mathbf{x}(n)|\mathbf{Z}(n-1))\\
p(\mathbf{z}|\mathbf{x}) &\rightarrow p(\mathbf{x}(n)|\mathbf{z}(n), \mathbf{Z}(n-1)) = p(\mathbf{x}(n)|\mathbf{Z}(n)).
\end{aligned}$$

Then, the weights $w(\mathbf{x}^k(n))$ are obtained by observing that $w(\mathbf{x}^k(n)) = p(\mathbf{z}(n)|\mathbf{x}^k(n))$ and the normalized weights $w_{\text{norm}}(\mathbf{x}^k(n))$ are used as a representation of $p(\mathbf{x}(n)|\mathbf{z}(n))$. After that, we apply the resampling by selection, where we use the selected samples $\mathbf{x}^k_{\text{selec}}(n)$ as an unweighted representation of $p(\mathbf{x}(n)|\mathbf{Z}(n))$. The last step of the recursion is the prediction, where we draw a new sample $\mathbf{x}^k(n)$ for each $\mathbf{x}^k_{\text{selec}}(n)$ from the density $p(\mathbf{x}(n+1)|\mathbf{x}^k_{\text{selec}}(n))$. This new set is a representation of the conditional density $p(\mathbf{x}(n+1)|\mathbf{Z}(n))$. The CONDENSATION algorithm is summarized in the Algorithm 1.

In a nutshell, the state space is partitioned in several parts and filled with the particles according to a probability measure (the normalized weights). The higher the probability, the more important the particle. Then, we can randomly select some particles to approximate the evolving density function by using a point-mass

```
   input  : Random draw $K$ particles from the cloud of points;
   for $k = 1 \to K$ do
       $\mathbf{x}^k(0) \sim q(\mathbf{x}(0))$;
       $w^k(0) = \frac{1}{K}$;

 1 begin
 2     begin Update:
 3         Set the importance weights;
 4         for $k = 1 \to K$ do
 5             $w(\mathbf{x}^k(n)) = p(\mathbf{z}(n)|\mathbf{x}^k(n))$;
 6         Calculate the normalized importance weights;
 7         for $k = 1 \to K$ do
 8             $w_{\text{norm}}(\mathbf{x}^k(n)) = \frac{w(\mathbf{x}^k(n))}{\sum w(\mathbf{x}^k(n))}$;
 9     end
10     begin Estimation:
11         $\hat{\mathbf{x}}(n) = \sum_{k=1}^{K} w_{\text{norm}}^k(n)\mathbf{x}^k(n)$;
12     end
13     begin Resampling:
14         $w_{\text{cum}}(\mathbf{x}^k(n)) = \sum_{j=1}^{k} w_{\text{norm}}(\mathbf{x}^k(n)), \forall k \in [1, K]$;
15         for $k = 1, \cdots, K$ do
16             $r^k \sim U[0, 1]$;
17             $j = \min_i w_{\text{cum}}^i \geq r^k$;
18             Select $\mathbf{x}_{\text{selec}}^k(n) = \mathbf{x}^j(n)$;
19     end
20     begin Prediction:
21         Set $n = n + 1$;
22         for $k = 1, \cdots, K$ do
23             $\mathbf{x}^k(n) \sim p(\mathbf{x}(n)|\mathbf{x}(n-1) = \mathbf{x}_{\text{selec}}^k(n))$;
24     end
25 end
   output: $\hat{\mathbf{x}}(n)$
```

Algorithm 1: CONDENSATION algorithm [65, 68].

histogram representation. However, the posterior density is unknown. So, we choose another distribution, the proposal density, from which we get the samples. As can be seen in [69], the particle filter was addressed in different areas with many terminologies. They all can be viewed as different variants in the generic Monte Carlo filter family. The CONDENSATION algorithm is just one of them. More detailed descriptions of the particle filter can be found in [65–69].

Chapter 3

Facial Landmarks Detection using the Inner Product Detector

Matched filter is a technique frequently used to detect a known signal, the so-called template, in an unknown signal composed by the desired signal added to an additive stochastic noise. The filtering, or detection, is achieved by cross-correlating the filter (or detector) with the unknown signal. The output of the filter is large when there is the presence of the desired pattern at the input and small otherwise. The detector, obtained by time-reversing the conjugate of the template, is optimal in the maximum signal-to-noise ratio (SNR) sense [70].

In a Correlation Filter (CF), we apply the matched filter in the frequency domain. To do so, the Discrete Fourier Transform (DFT) is used on the detector and the signal before computing the cross-correlation between them. The outcome will present a peak in case the sample is correlated with the detector [70]. One remarkable advantage of this technique is the tolerance to small variations of the pattern to be detected. Because of this, the CF and its variations, like Class-dependent Feature Analysis (CFA), have been widely used to detect objects in images [70–73].

In this work, in order to detect the landmarks, we also use a CF based detector called Inner Product Detector (IPD) [2–4]. In the IPD, the obtained detector is optimal with respect to the minimization of the squared classification error. As in correlation filters, the outcome is the inner product between the detector and a sample. As in the previous approach, the output is large if the sample is close to the desired pattern and small otherwise. However, there are two remarkable differences between the IPD and the CF: (i) CF performs a cross-correlation in the frequency domain, while IPD does it in the pixel domain. (ii) CF can only incorporate statistics by weighting the DFT samples, but IPD naturally incorporates a priori statistics in the autocovariance matrices.

Usually, a one-class classifier either trains with only positive examples (the ones belonging to the class of interest), or treats positive and non-positive (the ones that do not belong to the class of interest) examples differently during training. The name *one-class classifier* was coined in a Neural Network context [74], but has since then been applied to several other contexts, such as SVMs [75]. The one-class classifier is sometimes also called Positive-Unlabeled (PU) classifier [76].

3.1 One-Class IPD

Let $\mathcal{X}$ be a d-dimensional complex random variable with a realization $\mathbf{x}$. This realization can be associated with one of two classes, C_1 or C_0. We aim to find a d-dimensional detector $\mathbf{h}_1$, optimal in the least square sense, capable of detecting an object that belongs to C_1. Ideally, we want the following classification rule:

$$\mathbf{h}_1^{\mathrm{T}}\mathbf{x} = \begin{cases} 1, \text{ if } \mathbf{x} \in C_1 \\ 0, \text{ otherwise.} \end{cases} \tag{3.1}$$

With $[\cdot]^{\mathrm{T}}$ denoting a transpose operation. In other words, the classification is performed by the discriminant function

$$\mathbf{h}_1^{\mathrm{T}}\mathcal{X} = c, \tag{3.2}$$

where ideally $c = 1$ if $\mathbf{x} \in C_1$ and $c = 0$ if $\mathbf{x} \in C_0$.

Defining the classification error e as

$$e = \mathbf{h}_1^{\mathrm{T}}\mathcal{X} - c, \tag{3.3}$$

the squared error can be written as

$$\|e\|^2 = (\mathbf{h}_1^{\mathrm{T}}\mathcal{X} - c)(\mathbf{h}_1^{\mathrm{T}}\mathcal{X} - c)^T. \tag{3.4}$$

Since $\mathbf{h}_1$ and $\mathcal{X}$ are real-valued vectors, $\mathbf{h}_1^{\mathrm{T}}\mathcal{X}$ and c are scalars. So, Equation (3.4) can be expanded as follows:

$$\begin{aligned} \|e\|^2 &= (\mathbf{h}_1^{\mathrm{T}}\mathcal{X} - c)(\mathbf{h}_1^{\mathrm{T}}\mathcal{X} - c)^{\mathrm{T}} \\ &= (\mathbf{h}_1^{\mathrm{T}}\mathcal{X})(\mathbf{h}_1^{\mathrm{T}}\mathcal{X})^{\mathrm{T}} - (\mathbf{h}_1^{\mathrm{T}}\mathcal{X})c^{\mathrm{T}} - c(\mathbf{h}_1^{\mathrm{T}}\mathcal{X})^{\mathrm{T}} + cc^{\mathrm{T}} \\ &= \mathbf{h}_1^{\mathrm{T}}\mathcal{X}\mathcal{X}^{\mathrm{T}}\mathbf{h}_1 - \mathbf{h}_1^{\mathrm{T}}\mathcal{X}c - \mathbf{h}_1^{\mathrm{T}}\mathcal{X}c + c^2 \\ &= \mathbf{h}_1^{\mathrm{T}}\mathcal{X}\mathcal{X}^{\mathrm{T}}\mathbf{h}_1 - 2\mathbf{h}_1^{\mathrm{T}}\mathcal{X}c + c^2. \end{aligned} \tag{3.5}$$

Taking the expectation of the squared error, $E\left[\|e\|^2\right]$, we have:

$$E\left[\|e\|^2\right] = \mathbf{h}_1^{\mathrm{T}}E\left[\mathcal{X}\mathcal{X}^{\mathrm{T}}\right]\mathbf{h}_1 - 2\mathbf{h}_1^{\mathrm{T}}E\left[\mathcal{X}c\right] + c^2. \tag{3.6}$$

The desired detector is the one that minimizes the equation above. We can obtain it by taking the derivative of the Equation (3.6) with respect to $\mathbf{h}_1$ and equating it to zero. To do so, consider the following matrix properties:

$$\tfrac{\partial \mathbf{a}^{\mathrm{T}}\mathbf{B}\mathbf{a}}{\partial \mathbf{a}} = (\mathbf{B} + \mathbf{B}^{\mathrm{T}})\mathbf{a}, \tag{3.7}$$

$$\tfrac{\partial \mathbf{a}^{\mathrm{T}}\mathbf{b}}{\partial \mathbf{a}} = \mathbf{b}. \tag{3.8}$$

We find the minimum by taking the derivative and equating to zero:

$$\begin{aligned}\frac{\partial \|e\|^2}{\partial \mathbf{h}_1} &= \frac{\partial}{\partial \mathbf{h}_1}\left(\mathbf{h}_1^{\mathrm{T}} E\left[\mathcal{X}\mathcal{X}^{\mathrm{T}}\right]\mathbf{h}_1 - 2\mathbf{h}_1^{\mathrm{T}} E\left[\mathcal{X}c\right] + E\left[c^2\right]\right)\\ &= \left(E\left[\mathcal{X}\mathcal{X}^{\mathrm{T}}\right] + E\left[\mathcal{X}\mathcal{X}^{\mathrm{T}}\right]^{\mathrm{T}}\right)\mathbf{h}_1 - 2E\left[\mathcal{X}c\right]\\ &= 2E\left[\mathcal{X}\mathcal{X}^{\mathrm{T}}\right]\mathbf{h}_1 - 2E\left[\mathcal{X}c\right]\\ &= 0. \end{aligned} \tag{3.9}$$

Supposing that the training set has enough independent samples or, in other words, that $E[\mathcal{X}\mathcal{X}^{\mathrm{T}}]$ is a full-rank matrix (invertible), we have:

$$\mathbf{h}_1 = \left(E[\mathcal{X}\mathcal{X}^{\mathrm{T}}]\right)^{-1} E[\mathcal{X}c]. \tag{3.10}$$

Now, we just have to rewrite the terms $E[\mathcal{X}\mathcal{X}^{\mathrm{T}}]$ and $E[\mathcal{X}c]$ as functions of the training set samples. For the first term, we have:

$$E[\mathcal{X}\mathcal{X}^{\mathrm{T}}] = E[\mathcal{X}\mathcal{X}^{\mathrm{T}}|C_0]p(C_0) + E[\mathcal{X}\mathcal{X}^{\mathrm{T}}|C_1]p(C_1), \tag{3.11}$$

where $p(C_i)$, with $i = \{0, 1\}$, is the probability of a sample being from C_i. Since the class C_i has N_i samples $\mathbf{x}_{ik}$ ($k = \{1, \cdots N_i\}$), Equation (3.11) can be rewritten as follows:

$$E[\mathcal{X}\mathcal{X}^{\mathrm{T}}] = p(C_0)\frac{1}{N_0}\sum_{k=1}^{N_0}\mathbf{x}_{0k}\mathbf{x}_{0k}^{\mathrm{T}} + p(C_1)\frac{1}{N_1}\sum_{k=1}^{N_1}\mathbf{x}_{1k}\mathbf{x}_{1k}^{\mathrm{T}}. \tag{3.12}$$

We can expand the term $E[\mathcal{X}c]$ using a similar argument:

$$E[\mathcal{X}c] = E[\mathcal{X}c|C_0]p(C_0) + E[\mathcal{X}c|C_1]p(C_1). \tag{3.13}$$

As, ideally, $c = 1$ for $\mathcal{X}$ being from C_1 and zero otherwise, $E[\mathcal{X}c|C_0]$ equals to zero. So, in terms of the training samples, Equation (3.13) becomes

$$E[\mathcal{X}c] = p(C_1)\frac{1}{N_1}\sum_{k=1}^{N_1}\mathbf{x}_{1k}. \tag{3.14}$$

Putting Equations (3.12) and (3.14) together, we can express the detector $\mathbf{h}_1$ in terms of the training samples:

$$\mathbf{h}_1 = \left(p(C_0)\frac{1}{N_0}\sum_{k=1}^{N_0} \mathbf{x}_{0k}\mathbf{x}_{0k}^{\mathrm{T}} + p(C_1)\frac{1}{N_1}\sum_{k=1}^{N_1} \mathbf{x}_{1k}\mathbf{x}_{1k}^{\mathrm{T}} \right)^{-1} p(C_1)\frac{1}{N_1}\sum_{k=1}^{N_1} \mathbf{x}_{1k}. \tag{3.15}$$

Alternatively, recognizing the autocorrelation matrix $\mathbf{R}_i$ and the mean $\boldsymbol{\mu}_i$ from class C_i respectively as

$$\mathbf{R}_i = \frac{1}{N_i}\sum_{j=1}^{N_i} \mathbf{x}_{ij}\mathbf{x}_{ij}^{\mathrm{T}} \tag{3.16}$$

and

$$\boldsymbol{\mu}_i = \frac{1}{N_i}\sum_{j=1}^{N_i} \mathbf{x}_{ij}, \tag{3.17}$$

the detector $\mathbf{h}_1$ can be rewritten in terms of the moments of the random variable $\mathcal{X}$:

$$\mathbf{h}_1 = \mathbf{R}^{-1} p(C_1)\boldsymbol{\mu}_1. \tag{3.18}$$

Note that the correlation matrix of the data $\mathbf{R} = \left(\sum_{j=1}^{n} p(C_j)\mathbf{R}_j\right)$ must be a full rank matrix for the existence of $\mathbf{R}^{-1}$. It means that the dimension of the vectors must be smaller than the number of statistically independent samples. Fortunately, even if this condition does not hold, it is possible to find an approximated solution that could be the Minimum Average Correlation Energy (MACE) filter [72] or Class-dependence Filter Analysis (CFA) [71]. The vector $\mathbf{h}_1$ can be used to classify a unknown sample $\mathbf{x}_k$ according to the classification rule in Equation (3.1). It holds even if $\mathcal{X}$ is a real-valued random variable.

3.2 IPD in the Transformed Domain

An important interpretation of the IPD arises when we apply the Karhunen-Loève Transform (KLTr)[1] in the data. This transform is also known in the literature as Hotelling Transform or Principal Component Analysis (PCA) [62, 77].

Since the matrix $\mathbf{R}$ in Equation (3.16) has an inverse (because the number of training samples for the IPD is much larger than the detector's dimension), it can be expressed by its eigenvectors and eigenvalues as

$$\mathbf{R} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Lambda}\boldsymbol{\Phi}, \tag{3.19}$$

[1]The Karhunen-Loève Transform is most known in the literature as KLT. However, there is an optical flow based tracker called Kanade-Lucas-Tomasi that is also known as KLT. Since we use both in this thesis, we refer to the Karhunen-Loève Transform as KLTr and Kanade-Lucas-Tomasi as KLT to avoid ambiguity

where the columns of $\mathbf{\Phi}$ are the eigenvectors of $\mathbf{R}$ and are also referred to as the principal components. $\mathbf{\Lambda}$ is a diagonal matrix containing the variances along the principal directions and $[\cdot]^{\mathrm{T}}$ denotes the transpose operation. Therefore, Equation (3.18) becomes

$$\mathbf{h}_1 = p(C_1)\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Lambda}^{-1}\mathbf{\Phi}\boldsymbol{\mu}_1. \tag{3.20}$$

This implies that the IPD applied to a sample $\mathbf{x}$ becomes

$$\begin{aligned} \mathbf{h}_1^{\mathrm{T}}\mathbf{x} &= p(C_1)\boldsymbol{\mu}_1^{\mathrm{T}}\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Lambda}^{-1}\mathbf{\Phi}\mathbf{x} \\ &= p(C_1)\left(\mathbf{\Lambda}^{-1/2}\mathbf{\Phi}\boldsymbol{\mu}_1\right)^{\mathrm{T}}\left(\mathbf{\Lambda}^{-1/2}\mathbf{\Phi}\mathbf{x}\right). \end{aligned} \tag{3.21}$$

The operator $(\mathbf{\Lambda}^{-1/2}\boldsymbol{\phi})$ is referred to as the “Whitening Transform” [62, 63], since, after this transform, the data has equal variances directions (as is the cases in every white noise). Therefore, from Equation (3.21), one can interpret the IPD as a template matched to the mean of the class to be detected in the whitened domain. Using the matched filter terminology, it is a matched filter in the whitened domain.

3.3 IPD with Modified Discriminant Function

The IPD has a strong constraint in its formulation. The constraint imposed by the discriminant function in Equation (3.2) is that the classes are orthogonal. This orthogonality is unlikely in real world data. In practice, we can have negative samples with an IPD value greater than positive samples and even samples with IPD values outside the interval $[0, 1]$. One way of keeping the IPD values inside a bounded interval is to normalize them, and in doing so, the discriminant function becomes the cosine of the angle between the detector and the sample. With this strategy, we force the IPD value to lie in the interval $[-1, 1]$ and get rid of the negative samples that are not in the detector's direction but have a large IPD value because of their large magnitude. In doing so, we can keep just the output point $\mathbf{x}_m$ with maximum IPD value in order to obtain a single location as result. This yields the following discriminant function:

$$\mathbf{x}_m = \arg\max_{\mathbf{x}\in\mathbf{C}}\left(\frac{\mathbf{h}_1^T\mathbf{x}}{\|\mathbf{h}_1\|\|\mathbf{x}\|}\right), \tag{3.22}$$

where $\mathbf{C}$ is a set with K test samples $\mathbf{x}$.

In some cases, it is desirable to keep a set of the most likely candidates instead of a single point. In this case, it is possible to set up a threshold c_{angle} to select an interval of the IPD value that can classify correctly some percentage of the training samples.

$$\mathcal{X}_c = \left\{ \mathbf{x}_c : \frac{\mathbf{h}_1^T \mathbf{x}_c}{\|\mathbf{h}_1\| \|\mathbf{x}_c\|} > c_{\text{angle}} \right\}, \tag{3.23}$$

where $\mathcal{X}_c$ is a set of all selected points $\mathbf{x}_c$ that form a cloud. As a consequence, the IPD returns a cloud of points with high probabilities of belonging to the class of interest. This strategy is useful when the IPD is used as previous step for another method [7] or when it is desirable to keep only the most likely candidates. This feature is explored in Sections 4.1.1 and 4.1.2.

3.4 Cascade of IPD classifiers

At the end of Section 3.3 we have shown how to obtain a reduced set of candidates with high probability of being the desired feature. In this section we show an alternative way of doing so. The IPD is a weak classifier, what means that it is weakly correlated with the desired pattern. Fortunately, it is known that it is possible obtain a strong classifier by combining weak classifiers. In the case of the IPD, a cascade has been used [7]. A cascade is a degenerated binary tree in which each stage is adjusted to provide a high hit rate even at the cost of a high false positive rate. Only the samples classified as belonging to the class of interest in a given stage is provided to the next one. The number of false positive errors can be reduced by increasing the number of stages, but at cost of reducing the hit rate. Figure 3.1 shows the IPD cascade structure. Each stage was designed to keep 95% of the positive training data. In practice, we just picked up the threshold that keeps 95% of the positive samples in each stage. We automatically stop adding stages when the accumulated hit rate during training is smaller then 90%. The goal is to keep only a small set of good candidates. The cascade of IPD classifiers is particularly useful when the samples to be tested tend to be different from the ones used in training stage. We explore this in Chapter 4, when using IPD for facial feature tracking in video sequences.

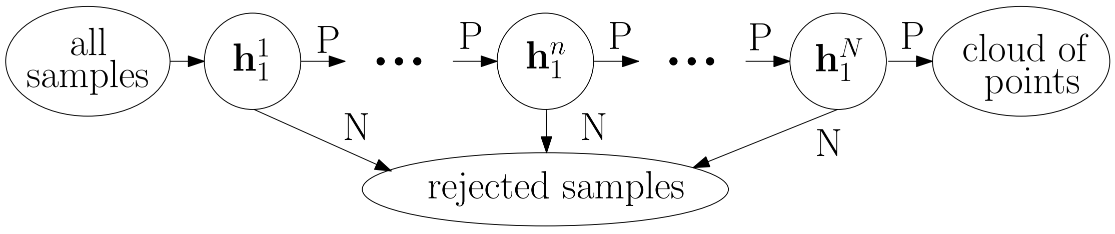


Figure 3.1: A block diagram of a cascade of IPD classifiers.

3.5 Connection of IPD to the Matched Filter, Linear Regression, and Linear Discriminant Analysis

The IPD is a dot-product operation like the Linear Discriminant Analysis (LDA) and the Linear Regression. However, some differences can be pointed out. To do so, let us first consider the optimal Bayes solution for the one-class classification case. Suppose that the classes are normally distributed, that is, the probability that a sample $\mathbf{x}$ belongs to the class C_i is given by $p(\mathbf{x}|C_i) \sim N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$. In that case, the sample belongs to the class C_1 if

$$(\mathbf{x}-\boldsymbol{\mu}_1)^{\mathrm{T}}\boldsymbol{\Sigma}_1^{-1}(\mathbf{x}-\boldsymbol{\mu}_1) + \ln|\boldsymbol{\Sigma}_1| - (\mathbf{x}-\boldsymbol{\mu}_0)^{\mathrm{T}}\boldsymbol{\Sigma}_0^{-1}(\mathbf{x}-\boldsymbol{\mu}_0) - \ln|\boldsymbol{\Sigma}_0| < c_{\mathrm{qda}}, \quad (3.24)$$

where $\boldsymbol{\Sigma}_i$ is the covariance matrix of the data from C_i, $|\cdot|$ is denoting the determinant, and c_{qda} is a threshold. The discriminant function in Equation (3.24) is also known as Quadratic Discriminant Analysis (QDA).

In the LDA, we consider that the covariance of the classes are equal with full rank. Mathematically, it means $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_0$. Considering this and doing some manipulations, Equation (3.24) becomes:

$$\left(\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_0-\boldsymbol{\mu}_1)\right)^{\mathrm{T}}\mathbf{x} > c_{\mathrm{lda}}, \quad (3.25)$$

for some constant c_{lda}. Defining the detector $\mathbf{w}_1 = \left(\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_0-\boldsymbol{\mu}_1)\right)^{\mathrm{T}}$, we have a discriminant function similar to the IPD's (see Equation (3.18)):

$$\mathbf{w}_1^T\mathbf{x} > c_{\mathrm{lda}}, \quad (3.26)$$

Assuming a zero mean data, that is, $\boldsymbol{\mu} = 0$, then $\boldsymbol{\Sigma} = \mathbf{R}$. So, we can find an interpretation of the LDA in the transformed domain similar to the one in Section 3.2:

$$\begin{aligned}
\mathbf{w}_1^{\mathrm{T}}\mathcal{X} &= \left(\left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Lambda}\boldsymbol{\Phi}\right)^{-1}(\boldsymbol{\mu}_0-\boldsymbol{\mu}_1)\right)^{\mathrm{T}}\mathcal{X} \\
&= (\boldsymbol{\mu}_0-\boldsymbol{\mu}_1)^{\mathrm{T}}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Lambda}^{-1}\boldsymbol{\Phi}\mathcal{X} \\
&= \left(\boldsymbol{\Lambda}^{-\frac{1}{2}}\boldsymbol{\Phi}(\boldsymbol{\mu}_0-\boldsymbol{\mu}_1)\right)^{\mathrm{T}}\left(\boldsymbol{\Lambda}^{-\frac{1}{2}}\boldsymbol{\Phi}\mathcal{X}\right). \quad (3.27)
\end{aligned}$$

From Equation (3.27), the LDA can be viewed as a template matched to the difference between the means of both classes (positive and negative) in the whitened domain. Thus, the LDA takes into account the difference between the means of both classes while the IPD takes into account the average of the class of interest only.

IPD is also similar to Linear Regression, particularly when Least Squares are used for estimating the parameters vector $\boldsymbol{\beta}$ (also known as regression coefficients). As can be seen in [60], an estimate $\hat{\boldsymbol{\beta}}$ can be obtained by

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}, \tag{3.28}$$

where $\mathbf{X}$ is a $K \times (d+1)$ matrix with samples $\mathbf{x}_k$ in the rows and a 1 in the first position of each and $\mathbf{y}$ is a N-dimensional vector of outputs from the training set. Defining $\mathbf{C} = \mathbf{X}^{\mathrm{T}}\mathbf{X}$ and $\boldsymbol{\mu} = \mathbf{X}^{\mathrm{T}}\mathbf{y}$, we have:

$$\hat{\boldsymbol{\beta}} = (\mathbf{C})^{-1}\boldsymbol{\mu}. \tag{3.29}$$

Following steps similar to those adopted in the IPD and LDA cases, the Linear Regression in the transformed domain is:

$$\begin{aligned}
\hat{\boldsymbol{\beta}}^{\mathrm{T}}\mathcal{X} &= \left(\left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Lambda}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\mu}\right)^{\mathrm{T}}\mathcal{X} \\
&= \boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Lambda}^{-1}\boldsymbol{\Phi}\mathcal{X} \\
&= \left(\boldsymbol{\Lambda}^{-\frac{1}{2}}\boldsymbol{\Phi}\boldsymbol{\mu}\right)^{\mathrm{T}}\left(\boldsymbol{\Lambda}^{-\frac{1}{2}}\boldsymbol{\Phi}\mathcal{X}\right).
\end{aligned} \tag{3.30}$$

So, the Linear Regression can be viewed as a template matched to the average of the training data in the whitened domain. The Linear Regression takes into account the average of the whole data, while the IPD takes into account the average of the class of interest only.

Up to this moment, we have shown that LDA and Linear Regression are slightly different from IPD, but a greater difference takes place when we compare their discriminant functions. As a matter of fact, there is no discriminant function in Linear Regression. The main goal of it is just finding an optimal hyperplane in the least squares sense that best describes the training data. The IPD detects the samples that, when whitened, have the smallest angular distance to the class of interest (specially when using the modified discriminant functions in Equations (3.22) and (3.23)). Since both, LDA and Linear Regression define a plane that bisects the space, the IPD is more selective, and has a better performance in the one-class scenario, when the classes are unbalanced. It occurs when the class of interest has a much smaller volume in space than the non-class case – exactly the case of feature/object detection in images.

3.6 Facial Landmarks Detection using IPD

One of the contributions of this work is the use of the IPD to detect facial landmarks in images. To do so, we employed the system depicted in Figure 3.2. In this section we describe each part of this block diagram. We conducted some experiments in order to show that the IPD can be used for this purpose. In this section we also describe these experiments, present the results and discuss them.

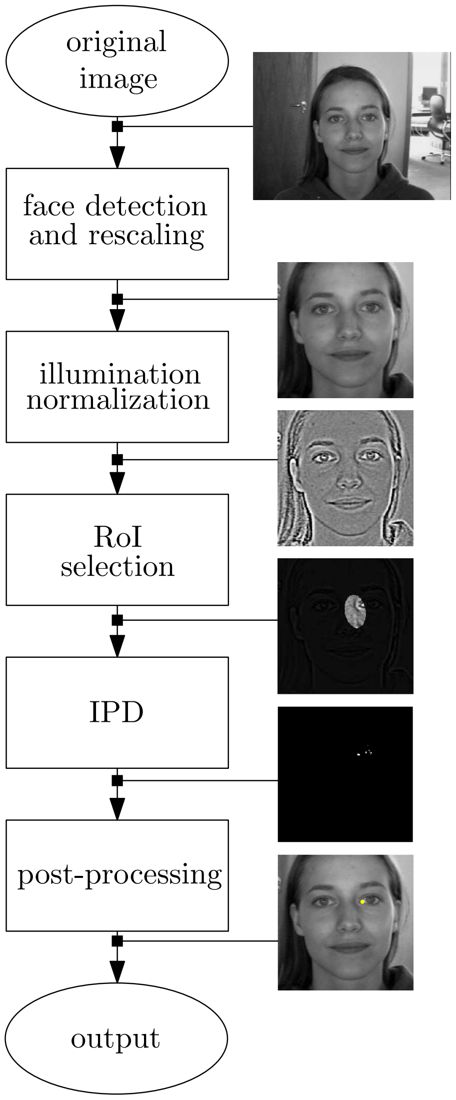


Figure 3.2: Block diagram of the facial landmark detection system employed in this work.

3.6.1 Experimental Procedures

We define a sample as a block centered on a point inside a Region of Interest (RoI) (more details about the RoI are given in the sequel). For training, the positive examples are the blocks whose central point is the coordinate of the ground truth (manual annotation) and its 8-connected neighbors – that provides nine positive samples per image. The negative class consists of the points inside the RoI that do not belong to the positive class. Note that there are much more negative samples per image than positive ones. In order to keep the balance between classes, we randomly selected nine negative samples from each training image. So, the training data consists of nine positive and nine negative samples for each training image. We subtract the average of the training set from all samples.

We use *10-fold* cross validation in our experiments, reporting the average of the folds as the result. The exceptions are the experiments in which we performed cross dataset validation.

Datasets

We use two datasets, BioID [78] and FERET [79]. BioID dataset consists of 1,521 gray level images of 23 subjects in frontal pose with a resolution of 384 × 286 pixels. This dataset is provided with two sets of annotations. The first set contains the coordinates of the center of the eyes. The second describes coordinates of 20 landmarks in each face, shown in Figure 3.3. The BioID contains images of individuals wearing glasses and with closed eyes, illumination and background with large variations, and multiple face scales, rotations and expressions.

The FERET database is composed by 11,338 facial images of 994 subjects at various angles. The images are colored, with a resolution of 512 × 786 pixels. In our experiments we use a subset of 2003 images, in which the subjects are in a near frontal pose, not wearing glasses and do not have facial hair. In this dataset, the background is plain with frontal illumination and a small range of expressions. The images used for the experiment were manually annotated with 13 facial points, as depicted in Figure 3.3. The annotation is also a contribution of this work and is available at [80].

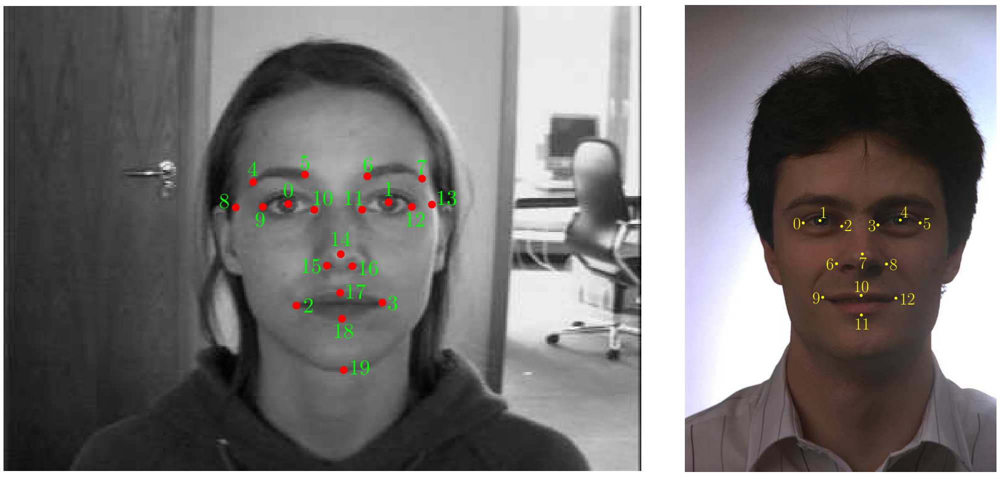


Figure 3.3: *Left*: BioID markup. *Right*: FERET markup. The annotation over the pupils is used as reference.

Preprocessing

All images are preprocessed before the detection procedure. Initially, the face rectangle is located using the OpenCV implementation of the Viola-Jones [54] detector. Although it only occurs in a small number of cases, in some images the algorithm fails to detect a face. These cases were not considered during evaluation. On the average, the face detector missed 5.3 of the 152.1 images per fold. The detected faces are scaled to a predefined size.

The images from BioID dataset were taken in a non-controlled environment, and have large variations in illumination and/or nonlinear illumination – characteristics present in many real-world applications. We incorporate robustness to these characteristics employing a four-step illumination normalization procedure [81, 82]: gamma correction, Difference of Gaussians (DoG) filtering, contrast normalization and hyperbolic tangent compression (to adjust the dynamic range). Figure 3.4 shows the block diagram of the employed illumination normalization method and Figure 3.5 shows the result of this method for some images from BioID dataset.

To reduce the computational complexity of our method, we constrained the IPD search region. We use the vertical symmetry of the human face and the fact that the eyes occupy approximately the same relative region of a face-detected window. Our search region is restricted to an area called *Region of Interest* (RoI) that has a high probability of containing the desired point. We learn the RoI from the ground truth of the training set, assuming that the location of the feature is a random variable $\mathcal{X}$ with Gaussian distribution, mean $\boldsymbol{\mu}_{\mathcal{X}}$ and covariance $\boldsymbol{\Sigma}_{\mathcal{X}}$. For each position $\mathbf{x}$ of the N images from the training set, we the compute the Mahalanobis distance

$$d = \sqrt{(\mathbf{x} - \boldsymbol{\mu}_{\mathcal{X}})^t \, \boldsymbol{\Sigma}_{\mathcal{X}}^{-1} \, (\mathbf{x} - \boldsymbol{\mu}_{\mathcal{X}})}. \tag{3.31}$$

The RoI is the elliptical region bounded by the training sample that maximizes the Mahalanobis distance $d_{\max}$ with a tolerance of 5%. The Equation (3.32) is used to verify whether a candidate point $\mathbf{x}_c$ belongs to the RoI or not. A candidate is considered inside it when

$$(\mathbf{x}_c - \boldsymbol{\mu}_{\boldsymbol{\mathcal{X}}})^t \, \boldsymbol{\Sigma}_{\mathcal{X}}^{-1} \, (\mathbf{x}_c - \boldsymbol{\mu}_{\mathcal{X}}) \leq (1.05 \; d_{\max})^2 \tag{3.32}$$

is satisfied. Figure 3.6 (*left*) shows the normalized ground truth of the training set superimposed to an image from BioID dataset with normalized size. The elliptical regions have a high probability of containing the centers of the eyes. Figure 3.6 (*right*) illustrates the RoIs. The region of the right eye contains 814 candidate points and the region of the left eye contains $1,923$. Note that, from this figure, the Gaussian assumption seems to be realistic. The points from the ground truth tend

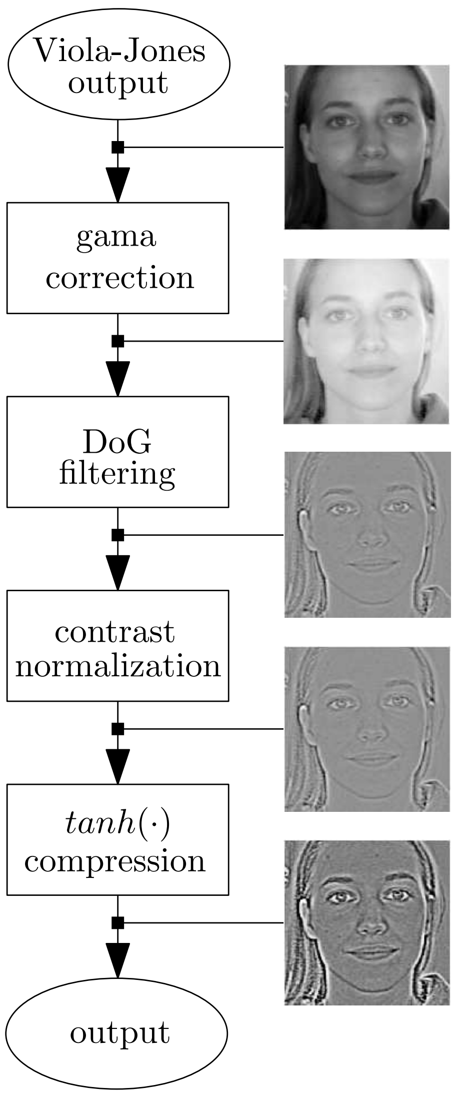


Figure 3.4: Block diagram of the illumination normalization method used in this work.

to spread in an approximately elliptical region of the faces with normalized size.

After the pre-processing step, we perform the detection using a sliding window with a low resolution template. In Section 3.6.2 we discuss the ideal sizes of the face and template used in the proposed methods.

Evaluation

Three accuracy measures were employed in order to assess the performance of the proposed methods: a local and two global metrics, all inspired on Jesorsky's work [83]. The local metric is

$$e = \frac{\|\mathbf{l}_a - \mathbf{l}_g\|}{d_{\text{eyes}}}, \tag{3.33}$$

where $\mathbf{l}_a$ is the coordinate of the automatic label given by the proposed method, $\mathbf{l}_g$ is the coordinate of the ground truth (the manually annotated point) and d_{eyes} is

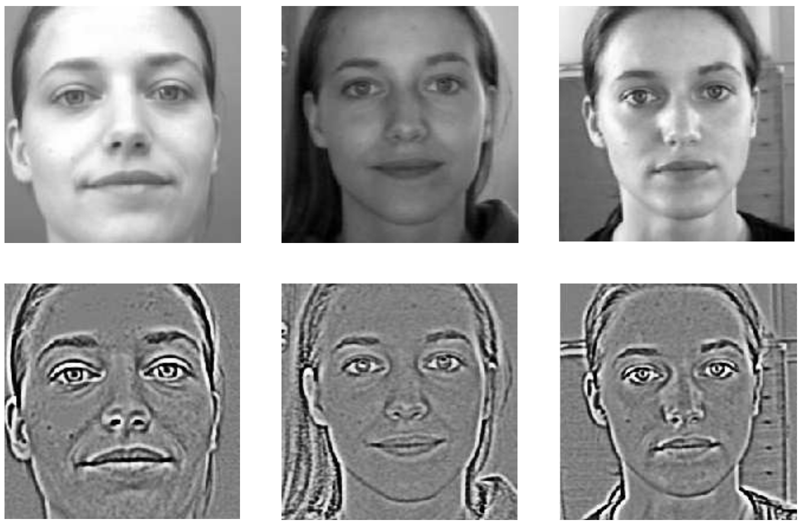


Figure 3.5: The result of the illumination normalization for some images from BioID.

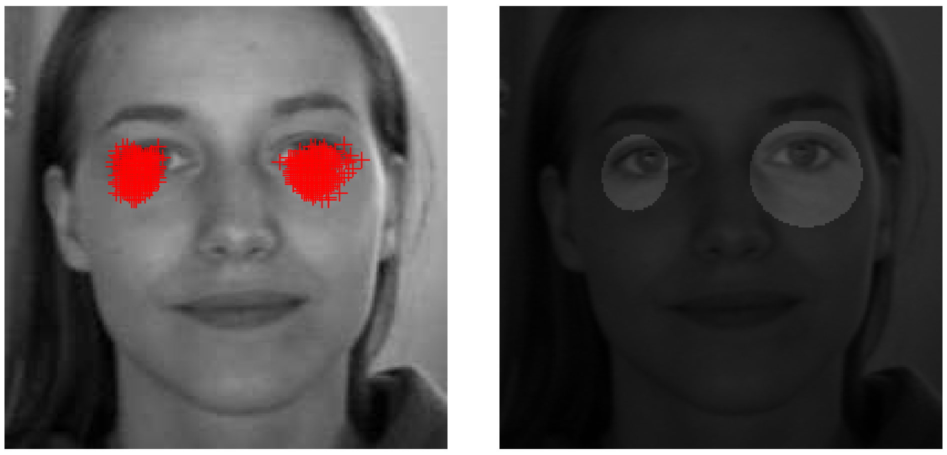


Figure 3.6: *Left*: training samples ground truth superimposed to a BioID image. *Right*: RoI estimated from the training samples.

the distance between the pupils from the ground truth. The local metric is used when evaluating the performance of a single point. It can be interpreted as the displacement between the output of the method and the ground truth normalized by the correct interocular distance.

As described in the literature [23, 83], we assess the precision of our method considering the intervals in which the value of e is contained. For example, if $e \leq 0.25$, the method is precise enough to locate the eye. This is so because the distance between the eye and the corner is approximately 25% of the interocular distance. Similarly, the method is capable of locating the iris if $e \leq 0.10$ and the pupil if $e \leq 0.05$.

The first global measure is

$$m_{eN} = \frac{1}{N d_{\text{eyes}}} \sum_{i=1}^{N} d_i, \tag{3.34}$$

where N is the number of features (17 in our case) and d_i is the Euclidean point-to-point distance between the output of the system and the ground truth for each individual feature location (error). Note that this is nothing but an averaged error measure over all landmarks.

In some cases, we only want to evaluate the performance in eye detection. So, it is common in the literature [23, 31, 34, 36, 57, 83–93] to use the metric e_{worst} (our second global measure) as follows:

$$e_{\text{worst}} \leq \frac{\max(e_l, e_r)}{d_{\text{eyes}}}, \tag{3.35}$$

where d_{eyes} is the interocular distance and e_l and e_r are the distances between the estimations for the left and right eyes and their ground truth, respectively. With this metric, we can compute the maximum normalized error between the two eyes. In a similar way, we can define two additional variations of the metric in Equation (3.35): one using the minimum normalized error between the two eyes, and the other using the averaged normalized error. These normalized measures give us, respectively, the curves labeled as *worse eye*, *best eye* and *average eye*, as can be seen in Section 3.6.6. These curves can be used to verify the bounds on the performance of the proposed method.

3.6.2 The Effect of the Face and Template Sizes on the Performance of the IPD

We evaluate training and testing procedures using different template and face sizes. As such, the faces obtained by the Viola-Jones Algorithm were scaled to a common size. Since RoI sizes varies with the face size, the detection computational cost is proportional to the face size M and the template size N. The relationship between the face and template sizes is depicted in Figure 3.7. The smaller the face and the template, the lower is the complexity, producing a trade-off between performance and computational cost. As an example, the computational complexity when using faces with 125×125 pixels and templates with 35×35 pixels is approximately 1.62 larger than using faces with 100×100 pixels and templates with 27×27 pixels $((135 \times 35)/(100 \times 27) \approx 1.62)$.

In this work, the faces were rescaled from 25×25 pixels up to 125×125 pixels. In most cases the faces were downscaled, but there are some cases in which faces were upscaled. With respect to the template, we started with block sizes from 15% up to 38% relative to the face size. Table 3.1 shows all face and block sizes used in this work. The entries of the tables are the absolute sizes of the block.

In Figures 3.8 to 3.10 we show the hit rate curves for the experiment with the

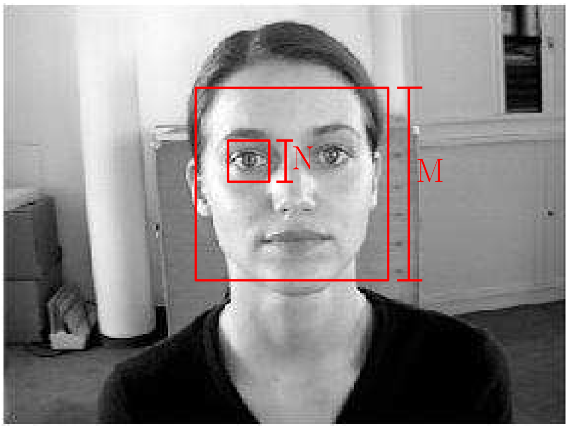


Figure 3.7: The computational cost is proportional to the face (M) and template (N) sizes.

Table 3.1: Evaluated relative block sizes and face sizes

		Size of the face (in pixels)				
		25×25	50×50	75×75	100×100	125×125
Block size (w.r.t. to the face size)	$\sim 15\%$	3×3	7×7	11×11	15×15	19×19
	$\sim 20\%$	5×5	11×11	15×15	21×21	25×25
	$\sim 28\%$	7×7	15×15	21×21	27×27	35×35
	$\sim 35\%$	9×9	17×17	27×27	35×35	45×45
	$\sim 38\%$	-	19×19	29×19	37×37	47×47

face and template sizes (using the local metric as explained in Section 3.6.1). We only show the results for three landmarks, but the results for all of them can be found in Appendix A. Before we discuss the sizes issue, the reader should keep in mind that the hit rate for an error smaller than 5% of the inter-ocular distance suffices for most of landmarks. In addition, the plateau present at the beginning of most of graphs from now on is due to the downscaling employed on the rectangle of the detected faces (most of them are larger than 125×125 pixels). As can be seen in the plots, the hit rate tends to increase with the block up to a given size. For the right pupil (Figure 3.8), for example, the hit rate increases with the block size up to blocks with approximately 28% of the face size. After this size, there is no significant increase in the hit rate. A similar behavior can be found in other landmarks, such as outer corner of right eye, inner corner of right eye, inner end of right eyebrow, tip of nose, right nostril and central point on outer edge of upper lip. To see the plots for these points, please refer to Appendix A. The same does not occur for more challenging landmarks such as the right corner mouth (Figure 3.10). For this point, the hit rate increases with the block size up to blocks with approximately 20% of the face size and then starts to fall as the block size increase. As can be seen in Appendix A, a similar behavior occurs for other landmarks as outer end of right eyebrow, right mouth corner and lower lip. All these facial features have larger variations. Increasing the feature size (and the details, as consequence) probably highlights the differences between samples more than their similarities. So, our detector cannot benefit from increasing the feature size when detecting patterns with large variations.

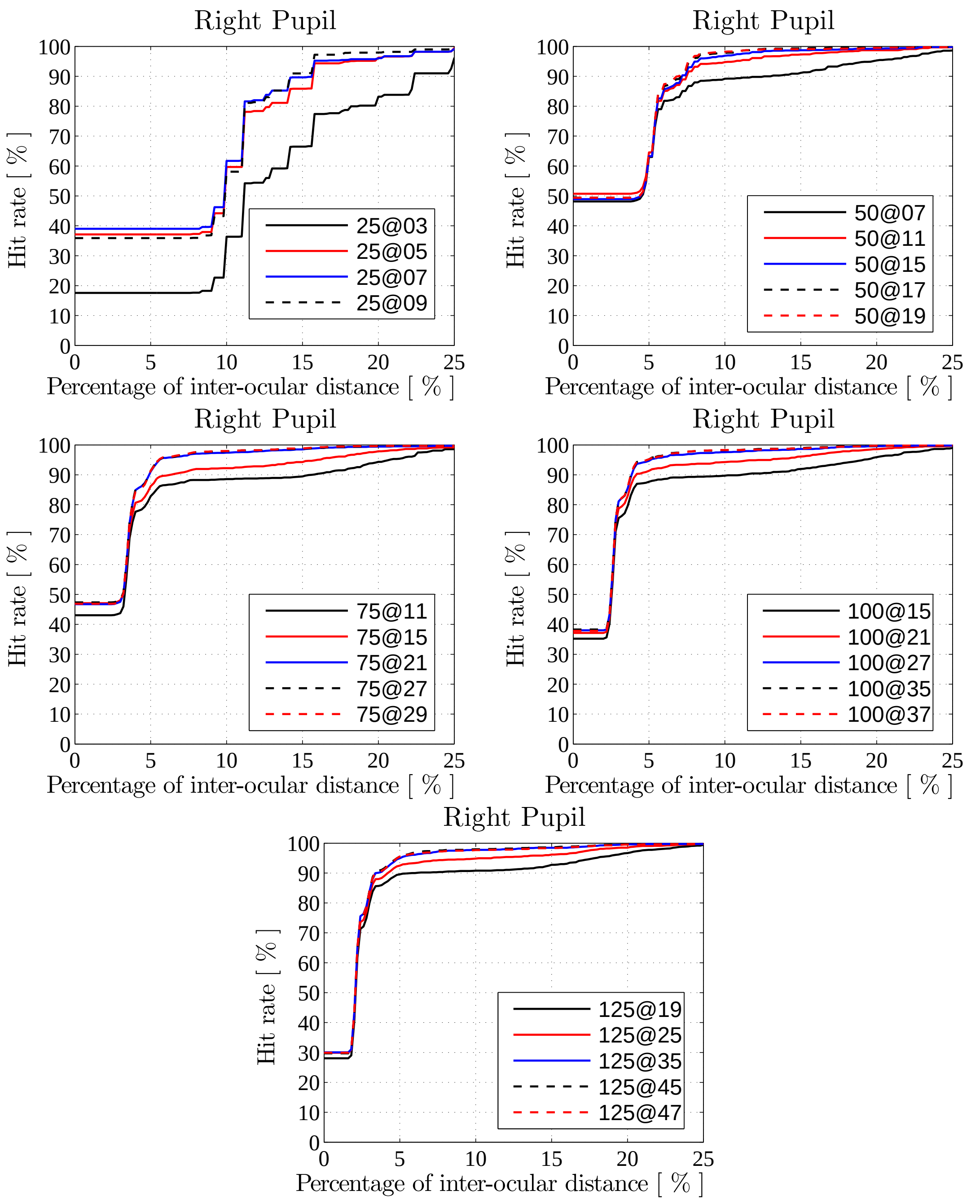


Figure 3.8: Average hit rate for the right pupil using faces from 25 × 25 up to 125 × 125 pixels and several block sizes.

Another important conclusion can be drawn from Figures 3.8 to 3.10, but is highlighted in Figure 3.11. It can be seen that there is no significant gain in using faces larger than 100 × 100 pixels in all cases. The same conclusion can be obtained by analyzing the complete results in Appendix A. So, this is the face size used in the remaining of this work.

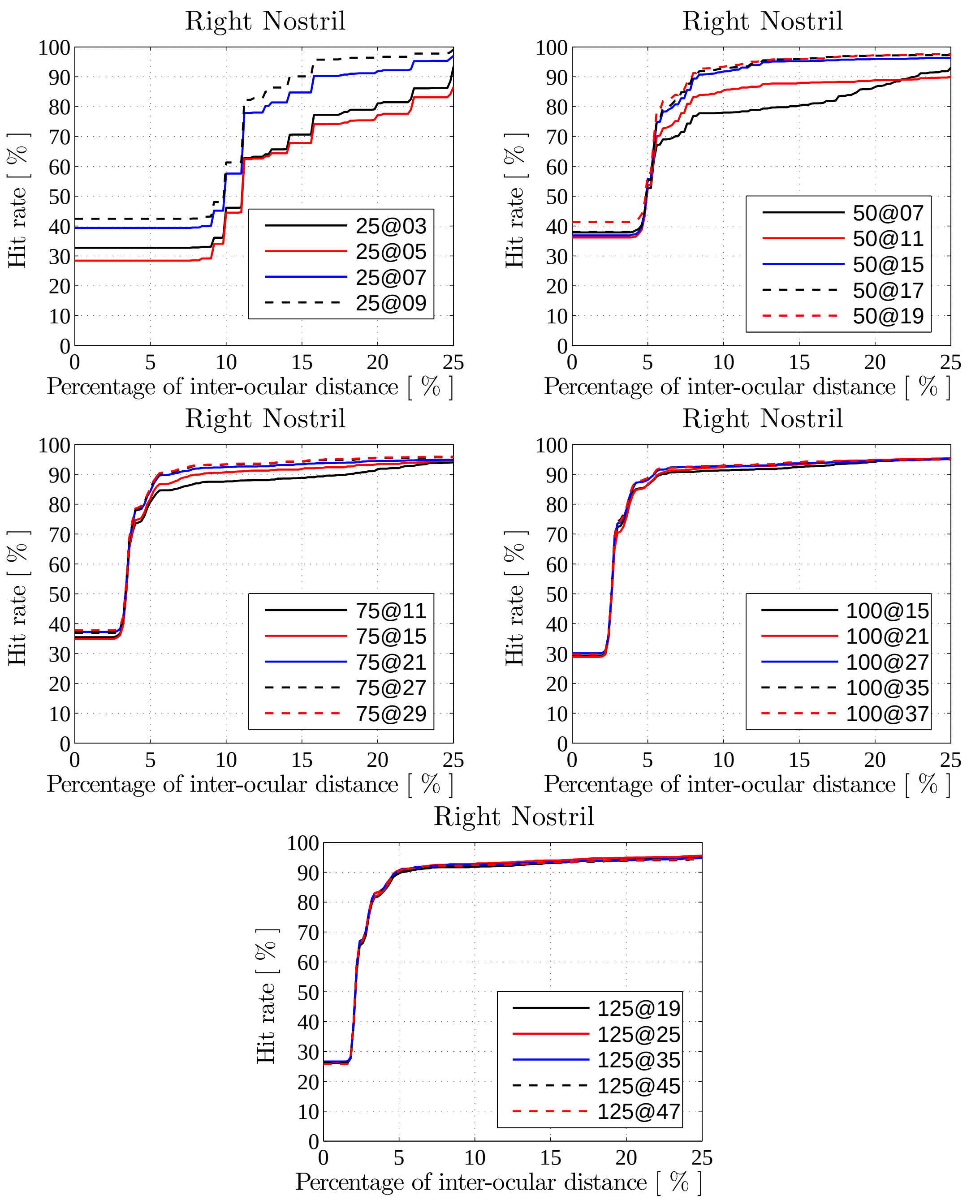


Figure 3.9: Average hit rate for the right nostril using faces from 25 × 25 up to 125 × 125 pixels and several block sizes.

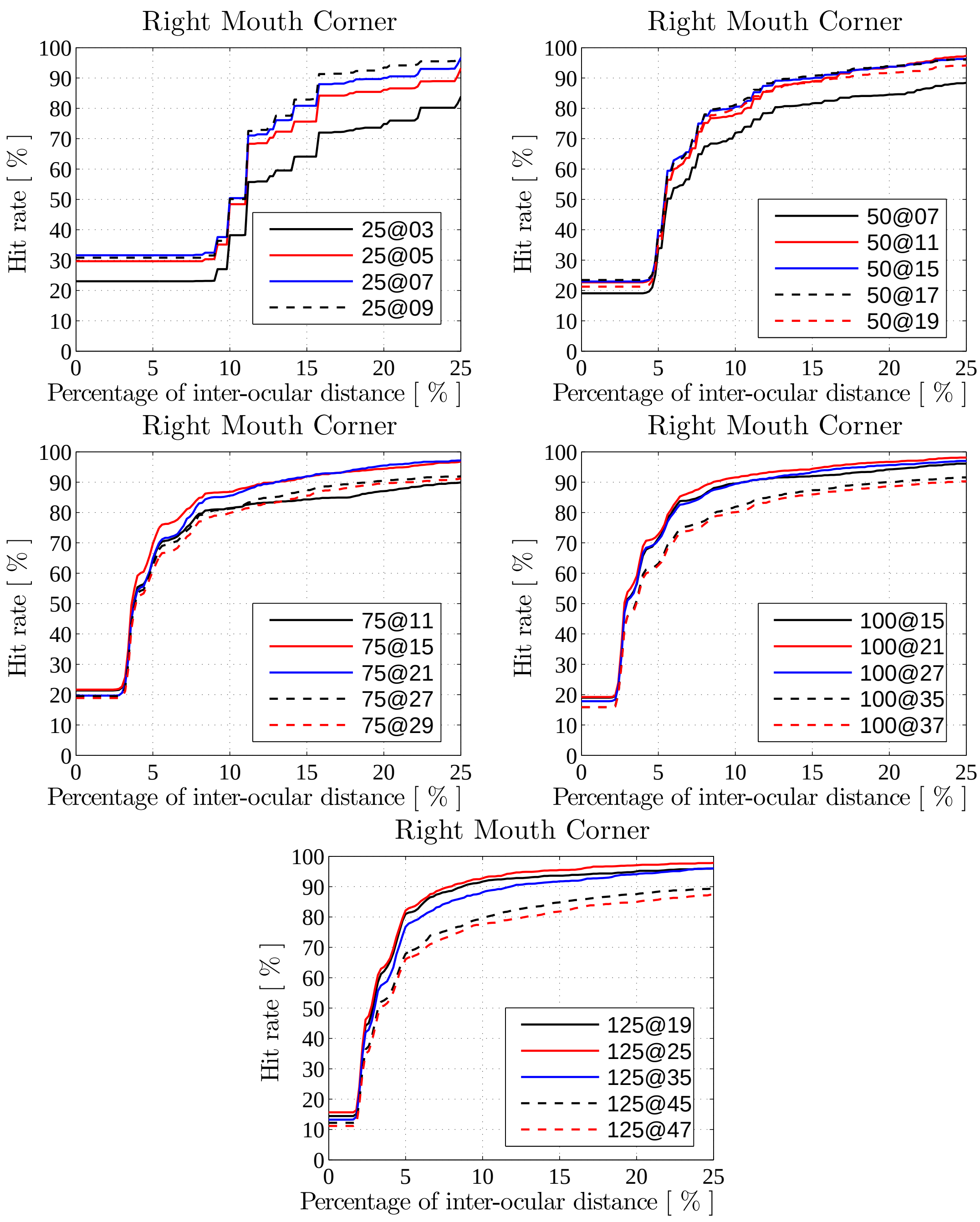


Figure 3.10: Average hit rate for the right mouth corner using faces from 25×25 up to 125×125 pixels and several block sizes.

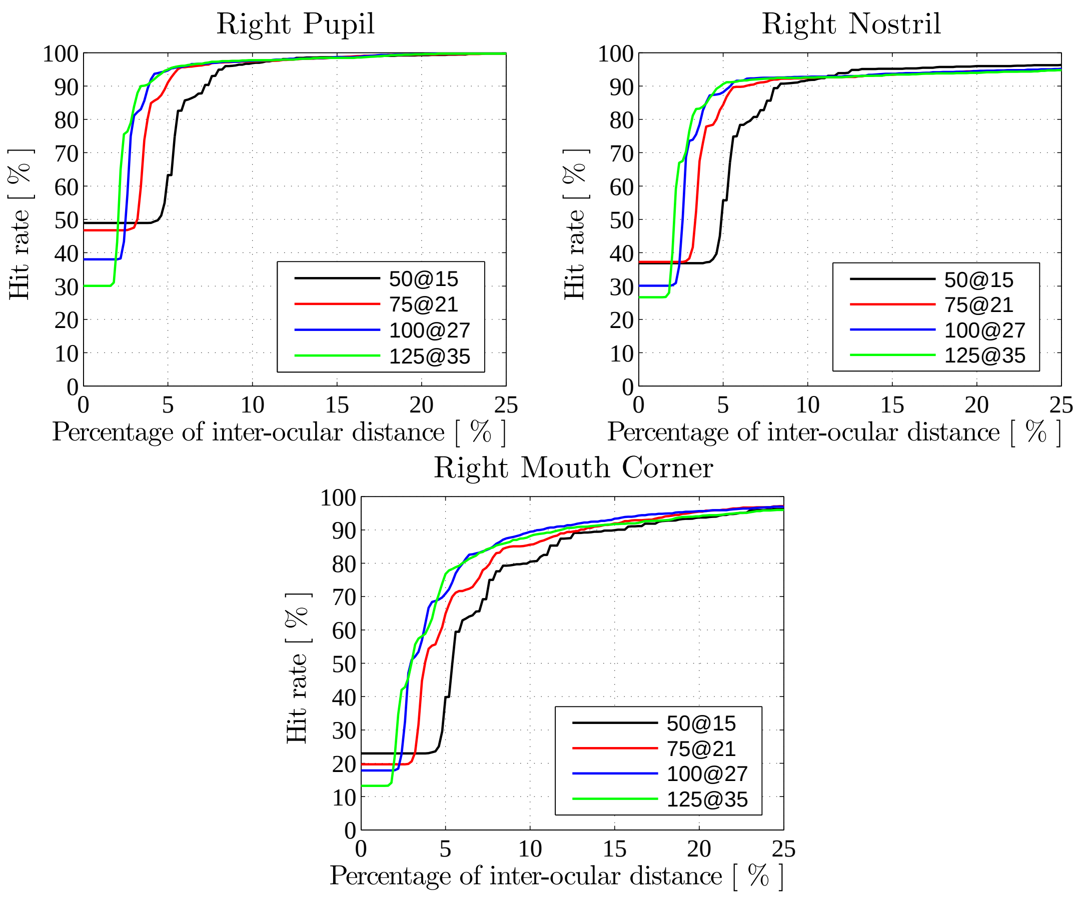


Figure 3.11: Average hit rate for the right pupil, right nostril and right mouth corner using faces from 50×50 up to 125×125 pixels and blocks with approximately 28% of the faces size

In Table 3.2, we show the size of the blocks with best results for each landmark (the numbering follows the convention in Figure 3.3, left). Note that, due to the symmetry of the face, we only show half of the points.

Table 3.2: Sizes of the blocks (in pixels) with best results for each landmark (using faces with 100×100 pixels).

Landmark	Feature	Block size
0	pupil	27×27
2	mouth corner	21×21
4	outer end of eyebrow	27×27
5	inner end of eyebrow	35×35
9	outer corner of eye	35×35
10	inner corner of eye	35×35
14	tip of nose	35×35
15	nostril	27×27
17	upper lip	35×35
18	lower lip	21×21

3.6.3 Cross-Dataset Validation Between BioID and Color FERET Databases

In order to evaluate the capability of generalization of our method, we perform cross dataset validation between BioID and FERET, with results in Figure 3.12. All plots were obtained using the local metric as explained in Section 3.6.1. Due to the symmetry of the face, we only show the results for half of the points (the complete results can be seen in Appendix A). The curves labeled as FERET represent the experiments in which we trained the IPD using BioID and ran the test in the FERET. The converse has been made for the curves labeled as BioID. For most features, the hit rate is high enough to assure the generalization capability. In other words, our facial features detection system can keep the hit rate, even when samples different from those used in training are presented. In addition, as can be seen in the plots, the results tend to be better when training using BioID and testing on FERET. Such results are expected, since the BioID is more challenging than the FERET database.

3.6.4 Comparison Between IPD and SVM-RBF

In this Section, we compare the IPD with a state-of-the-art classifier: the SVM with RBF kernel. The experiments with the SVM were conducted in Matlab environment using the SVMlight toolbox [61]. We used the same pre-processing used in the experiments with the IPD. Since the SVM also provides a cloud of points at its output, we need to employ a post-processing step. For simplicity, we just use the vector median of the cloud as the candidate point. All detected faces were resampled to 200×200 pixels. The blocks with size 13×13 pixels centered at the ground truth and their 8-connected neighbors were considered as positive samples. The blocks centered at the remaining points inside the RoI were considered as negative samples.

The best parameters γ and C (RBF kernel) where selected by employing a grid search with an exponential growing rule for the parameters:

$$\begin{aligned} \gamma &= 2^i, \text{ with } i = \{-18, -17, \cdots, 1\}. \\ C &= 2^j, \text{ with } j = \{-15, -14, \cdots, 15\}. \end{aligned}$$

Each combination of the parameters where evaluated using k-fold cross validation with seven folds. The parameters that led to the best average results in the training stage were $\gamma = 2^{-10}$ and $C = 2^1$. Given the best parameters, we performed the test procedure using another k-fold cross validation with seven folds in the whole dataset.

In Figure 3.13, we have the hit rate curves for the IPD and the SVM. The curves

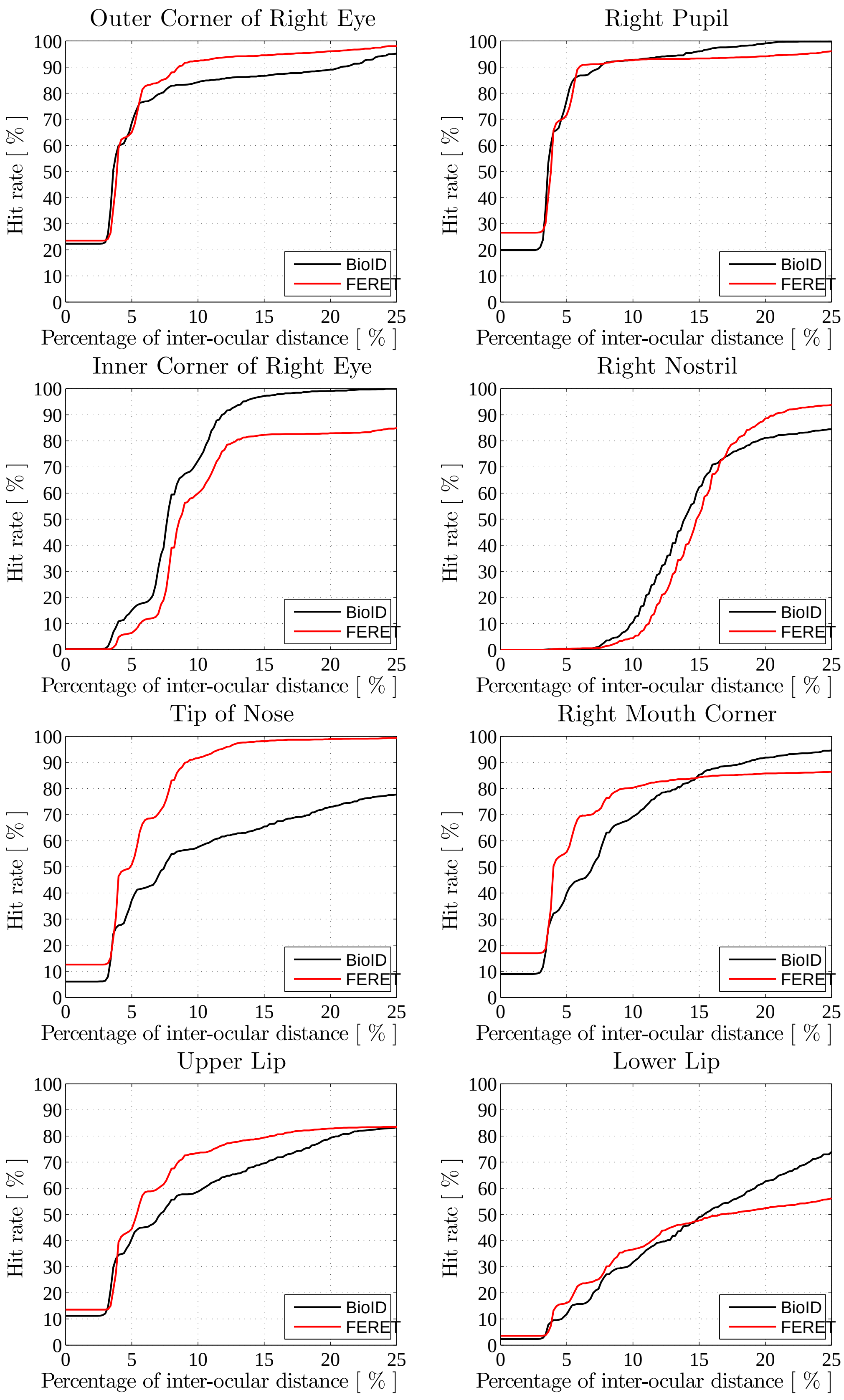


Figure 3.12: Cross dataset validation results. Red curves: training with BioID and testing on FERET. Black curves: training with FERET and testing on BioID.

in this figure are global results, since they were obtained using the e_{17} metric (see Equation (3.34) in Section 3.6.1). As we can notice, the IPD has a performance that is competitive with the SVM based method for an accuracy bigger than 5% of the inter-ocular distance. On the other hand, the SVM is better than the IPD for a smaller accuracy. This is so because the hit rate obtained for landmarks like the ones on the lips, nose and eyebrows have a smaller maximum value when using IPD.

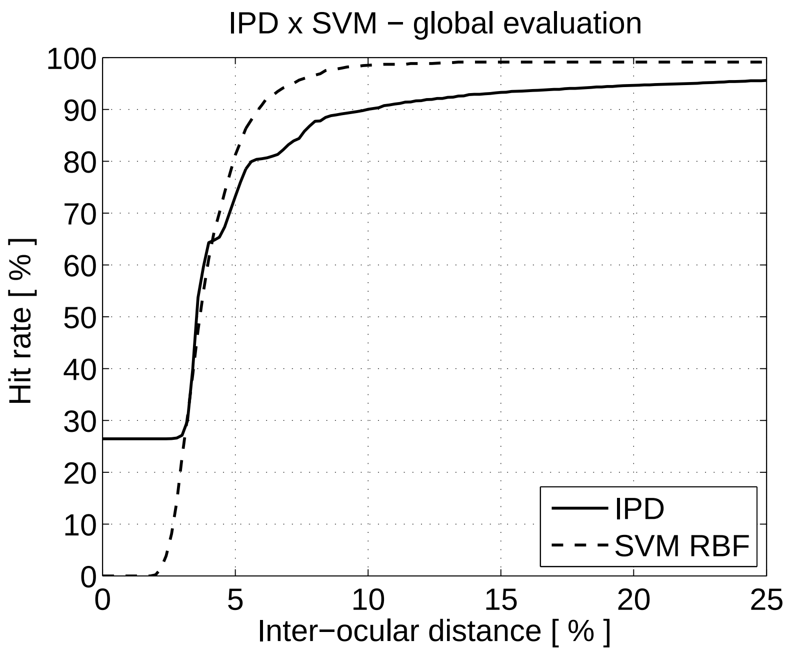


Figure 3.13: Comparison between IPD with SVM with RBF kernel. In this plot, both curves were obtained using the e_{17} error measure.

From Figures 3.14 to 3.17 we show the local results of all 10 inner landmarks for both IPD and SVM (using the local metric as explained in Section 3.6.1). Note that, due to the symmetry of the face, we only show half of the points. To see the complete results, refer to Appendix A.

In top left, top right and bottom parts of Figure 3.14, we have a comparison between IPD and the SVM for the landmarks of the eye.

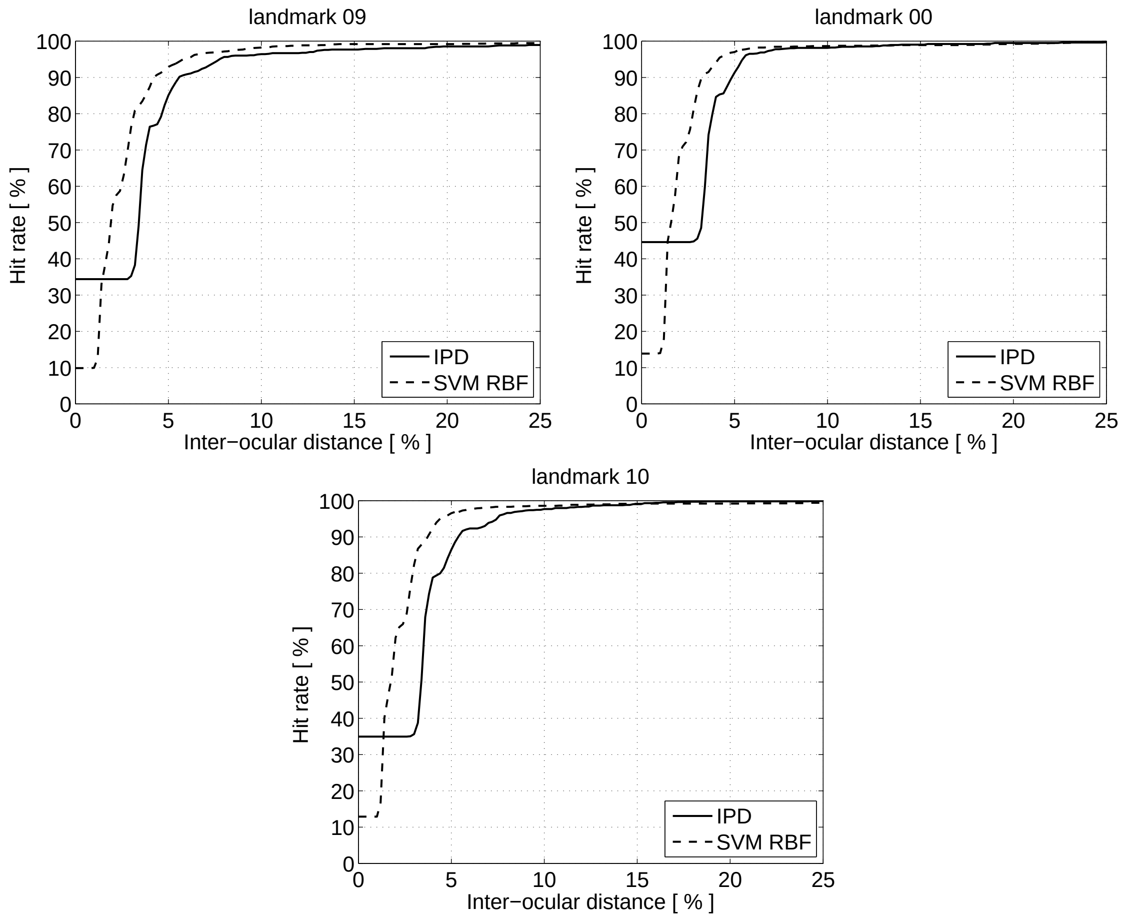
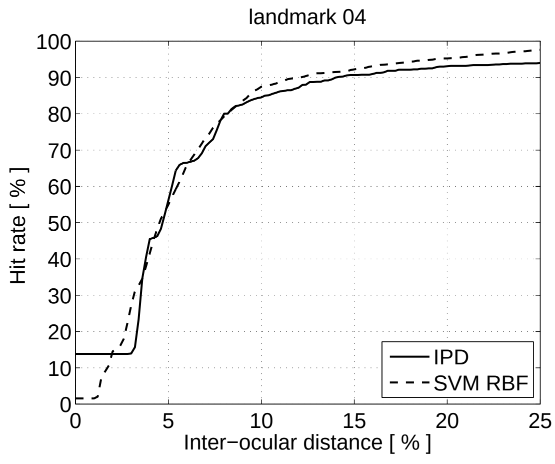


Figure 3.14: *Top left*: the results for the outer corner of right eye. *Top right*: the results for the right pupil. *Bottom*: the results for the inner corner of right eye.

In left and right sides of Figure 3.15, we have a comparison between IPD and the SVM for the landmarks on the eyebrows.



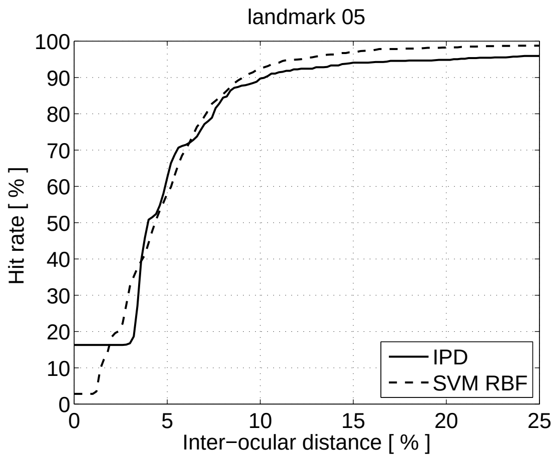
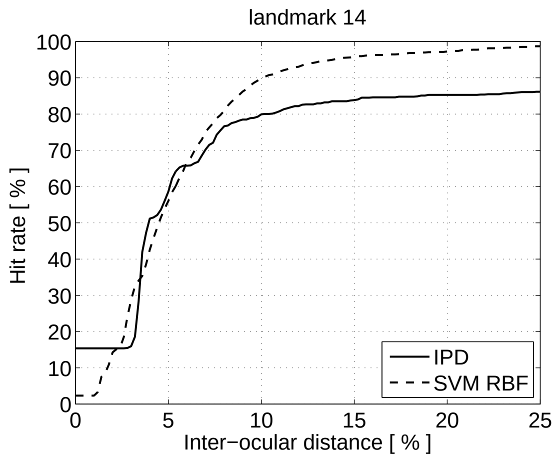


Figure 3.15: *Left*: the results for the outer end of right eyebrow. *Right*: the results for the inner end of right eyebrow.

In left and right sides of Figure 3.16, we have a comparison between IPD and the SVM for the landmarks on the nose.



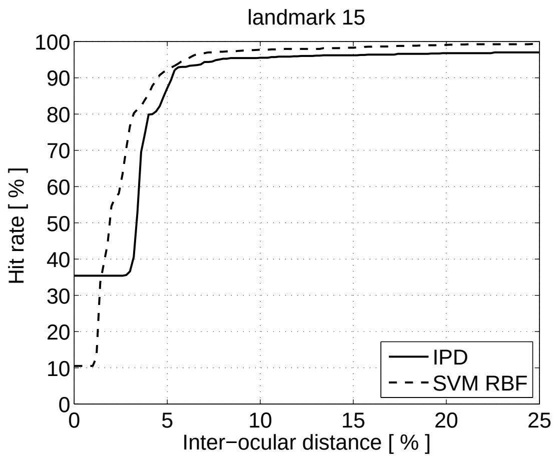


Figure 3.16: *Left*: the results for for the tip of nose. *Right*: the results for the right nostril.

In top left, top right and bottom parts of Figure 3.17, we have a comparison between IPD and the SVM for the landmarks on the mouth.

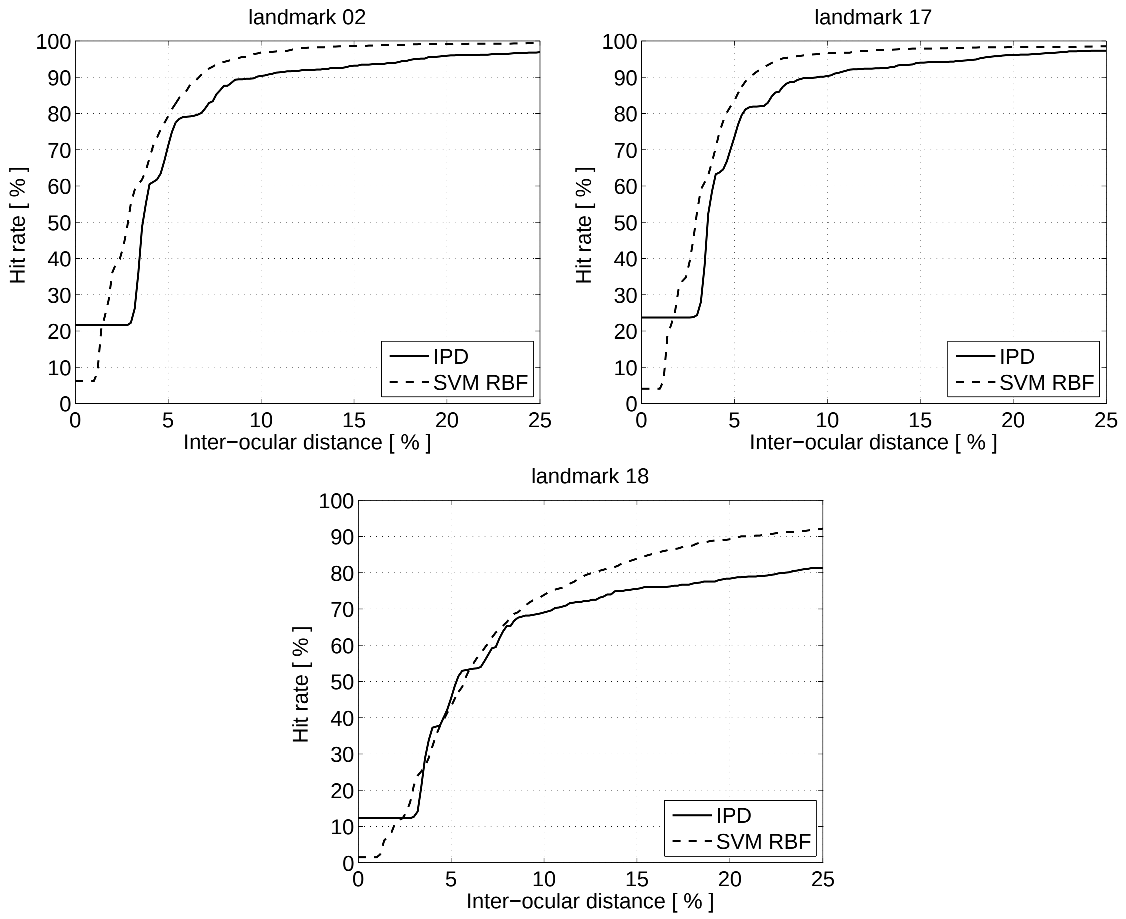


Figure 3.17: *Top left*: the results for the right mouth corner. *Top right*: the results for the central point on outer edge of upper lip. *Bottom*:the results for central point on outer edge of lower lip.

In most cases the SVM has a performance equal to or better than the IPD. However, the difference is small even in the worst case. This result is expected, since the SVM is much more complex in computational terms. For the landmark 9, for example, the evaluation using SVMlight in Matlab environment took approximately 28s per image on the average (in a computer without load). In the same conditions, the IPD took approximately 0.09s per image. Considering that the IPD can be used in real time applications [5], it can be advantageous to use it in facial landmark detection systems. Mainly if an accuracy greater than 5% of the inter-ocular distance is required. It means that if we need more accurate localization, we have a double benefit from using the IPD: it is competitive for this accuracy and is computationally cheaper.

As we have shown in Section 3.6.3, the IPD has a good generalization capability. It will be even more evident in Chapter 4, where the classifiers trained using the BioID database are employed with success on our video database. The same cannot be said about the SVM. Although the hit rate of the SVM on the BioID is greater than the IPD, the hit rate is low when using the classifiers trained through BioID on our video database.

3.6.5 Facial Landmark Detection using IPD - Comparison with State-of-the-Art Techniques

In this section, we show a comparison with state-of-the-art techniques in facial landmarks detection. To do so, we evaluated our method on the BioID dataset using 10-fold cross-validation and the global m_{e17} error measure (proposed by [29] and described in Section 3.6.1). This comparison is in Figure 3.18, in which all depicted curves were obtained from [94] using the Engauge digitizer, an open source digitizing software [95]. The plateau at 20% for the IPD curve is due the downscaling to 100×100 pixels employed on the detected faces (most of them are bigger than this size).

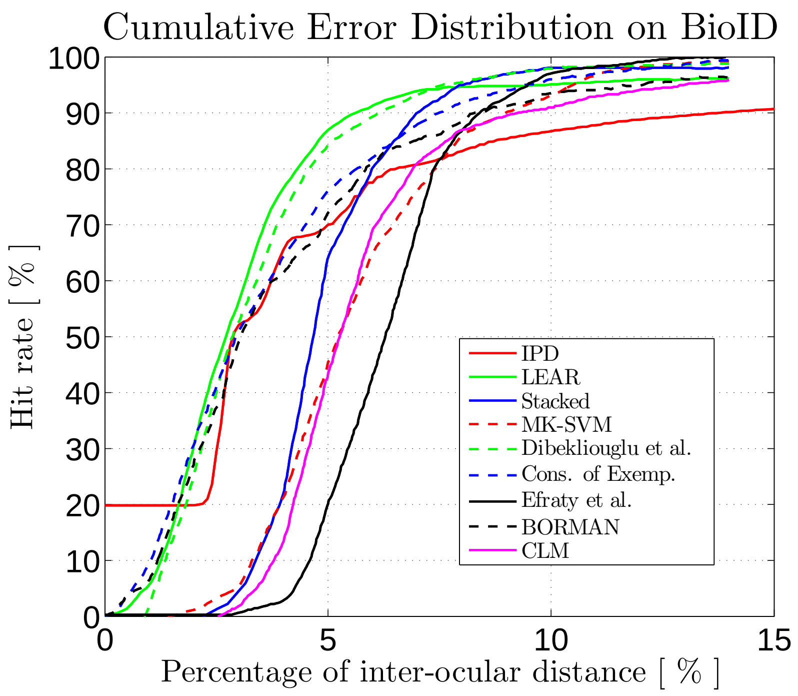


Figure 3.18: Comparison with the state of the art. The references are: Lear [94]; BORMAN [96]; Stacked [97]; NK-SVM [98]; CLM [99]; Cons. of Exemp. [100]; Efraty etal [101]; Dibeklioglu et al. [102].

Most of the methods shown in Figure 3.18 are dataset independent. The one in [101] was trained using 104 images from BioID. We also trained our method using BioID, but employing a k-fold cross validation instead of selecting a single training group. Even using BioID for training, our method has a good generalization capability, as can be seen in Section 3.6.3. Note that the IPD has a competitive hit rate in the region of the graph related with a greater accuracy (percentage of inter-ocular distance smaller than 5%). The IPD can be even more competitive if we consider that: (1) Our method has no global face model – we only use local detectors; (2) IPD has a good generalization capability, since it can be used to detect features in other databases (actually the other methods that are not dataset dependent also have this feature); and (3) IPD's detection procedure consists basically of dot products – it is computationally fast, and thus can be used in real time applications (the same cannot be said for some methods in Figure 3.18).

3.6.6 Eye Detection using IPD - Comparison with State-of-the-Art Techniques

As we have shown in Section 3.6.5, the IPD can be used with success to locate a set of landmarks on faces simultaneously. However, considering only the hit rate, there is a decrease in competitiveness if a less accurate localization is required. This is evident if we observe Figure 3.18. Our hit rate is not competitive in the part of the graph corresponding to a percentage of the inter-ocular distance larger than 5%. As can be seen in Appendix A, a possible cause can be the low hit rate of more challenging landmarks such as the ones on the eyebrows, tip of nose and lower lip. On the other hand, many applications involve only detection of features on eyes, such as attention and gaze estimation of disabled people (in order to control assistive devices), driver fatigue detection, entertainment, biometrics and so on. So, in this section, we present the results of the proposed method in eye detection. A comparison with state-of-the-art techniques is presented as well.

A global performance evaluation of our detection method can be seen in Figure 3.19. In order to obtain the performance bounds, we have plotted the three normalized errors (as described in Section 3.6.1) in this plot. The plateau present at the beginning of the curves is due to the downscaling employed on the rectangle of the detected faces (most of them are bigger than 100×100 pixels). A qualitative result is in Figure 3.20, which depicts the detection result and ground truth superimposed on some images from BioID. We have the average hit rate, as well as the standard deviation, for relevant values of e_{worst} (see Equation (3.35)) in Table 3.3. Since we used k-fold cross validation with 10 folds in all experiments, all results are the average across all folds. From these average hit rate figures and their low variance, we can conclude that the IPD is worth considering as a fiducial point detector.

In our experiments, it took 83.4 milliseconds on average to find both eyes in the images from the BioID dataset. From the total time, it took 12.4 ms to do the

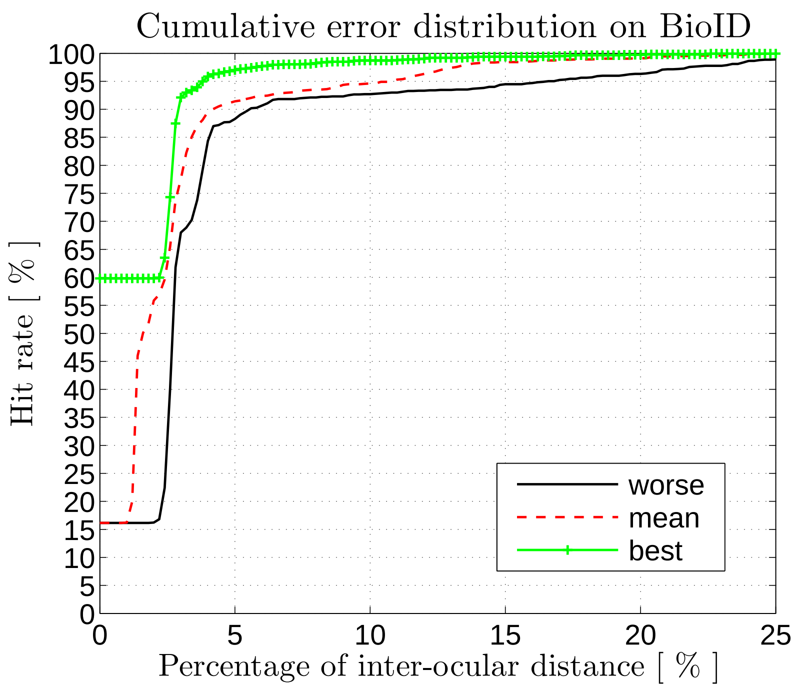


Figure 3.19: Hit rate vs. normalized error for *worse eye*, *best eye*, and *average eye* (see Section 3.6.1).

Table 3.3: Evaluation of our method for values of interest of e_{worst}.

	Precision				
	$e \leq 0.05$	$e \leq 0.1$	$e \leq 0.15$	$e \leq 0.2$	$e \leq 0.25$
IPD (BioID)	88.3 ± 2.4	92.7 ± 1.7	94.5 ± 1.9	96.3 ± 1.2	98.9 ± 0.7

pre-processing step, 20.5 ms to find the right eye, 50.2 ms to find the left eye, and 0.3 ms to do the post-processing. The difference in localization times is because the left eye RoI is larger than the right eye RoI – see in Figure 3.6. As we only compute inner products in low resolution images, IPD is fast. Additionally, if the detection and post-processing are implemented in parallel architectures, such as GPUs or multiprocessor machines, the final performance can be even better. We implemented our algorithms in Matlab, and the experiment times reported above were obtained using a machine with a 3.07 GHz Intel(R) Core(TM) i7 Q950 with 8 GB DDR3/1333MHz RAM memory.

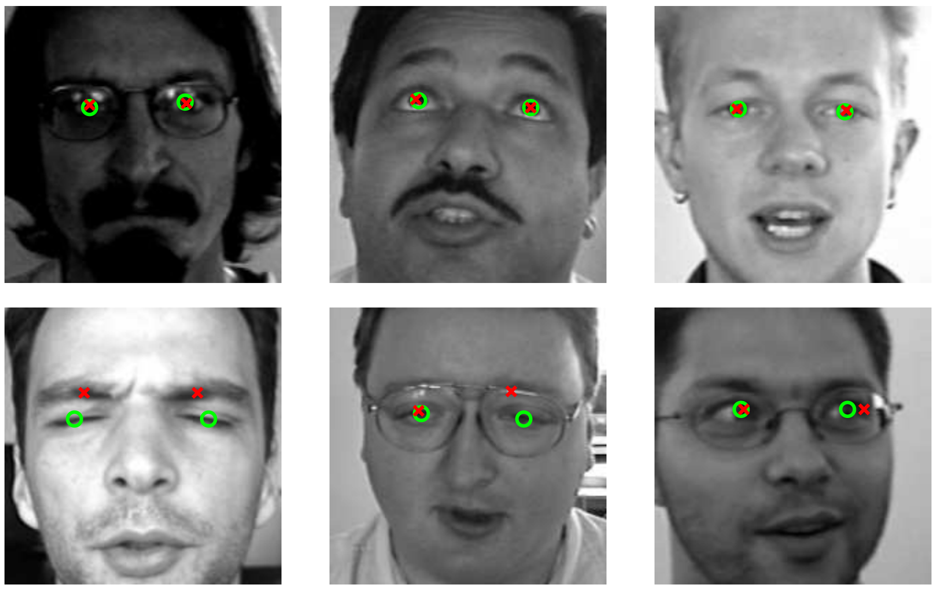


Figure 3.20: Some results from the BioID dataset. Green circles are the ground truth; red crosses are the automatic labels. At the top row, successful examples. Bottom row, unsuccessful ones.

Table 3.4 shows a comparison with the state of the art. All the methods in the table used the same dataset (BioID is the standard in the literature for this type of comparison) and the same error metric (e_{worst} is a standard metric as well). Although the training and testing procedures may differ among them, the comparisons can be considered as fair if we regard each method as a black box. Each column has the results for relevant values of e_{worst} and the values inside $\{\cdot\}$ are the respective ranks of our method. Our method has rank 1 for the best accuracy ($e \leq 0.05$), which means that it is capable of locating the pupil with good precision. Our rank goes down for worse accuracies, but our results remain competitive, particularly if we consider the advantage of the low computational complexity of our method.

Table 3.4: Performance comparison between different methods on BioID dataset. ‡ Values provided by [31]. † Values provided by [23]. § Values provided by [93]. The best and our results are in bold. (1) Using MIC+SIFT. "–" Indicate the cases in which the authors did not provide a worse eye curve and the values could not be estimated. (*) Images with closed eyes and glasses were omitted. $\{\cdot\}$ indicate the rank of our method in a specific accuracy.

Method	$e \leq 0.05$	$e \leq 0.1$	$e \leq 0.15$	$e \leq 0.2$	$e \leq 0.25$
Jesorsky et al., 2001‡ [83]	38.0%	78.8%	84.7%	87.2%	91.8%
Behnke, 2002‡ [84]	37.0%	86.0%	95.0%	97.5%	98.0%
Cristinacce et al., 2004‡ [85]	57.0%	**96.0**%	**96.5**%	97.5%	97.1%
Hamouz et al., 2005‡ [86]	58.6%	75.0%	80.8%	87.6%	91.0%
Asteriadis et al., 2006‡ [36]	44.0%	81.7%	92.6%	96.0%	97.4%
Bai et al., 2006† [87]	37.0%	64.00%	–	–	96.00%
Campadelli et al., 2006‡ [88]	62.0%	85.2%	87.6%	91.6%	96.1%
Chen et al., 2006‡ [89]	–	89.7%	–	–	95.7
Everingham and Zisserman, 2006§ [57]	45.87%	81.35%	–	–	91.21%
Niu et al., 2006‡ [90]	75.0%	93.0%	95.8%	96.4%	97.0%
Turkan et al., 2007‡ [91]	18.6%	73.7%	94.2%	**98.7**%	**99.6**%
Kroon et al., 2008‡ [92]	65.0%	87.0%	–	–	98.8%
Asadifard and Shanbezadeh, 2010‡ * [34]	47.0%	86.0%	89.0%	93.0%	96.0%
Timm and Barth, 2011‡ [31]	82.5%	93.4%	95.2%	96.4%	98.0%
Valenti and Gevers, 2012† (1) [23]	86.09%	91.67%	–	–	97.87%
Ren et al., 2014§ [93]	77.08%	92.25%	–	–	98.99%
IPD	**88.3**% {1}	92.7% {4}	94.5% {5}	96.3% {6}	98.9% {3}

Chapter 4

Facial Landmarks Tracking using the Inner Product Detector

The method proposed in Chapter 3 is capable of performing detection of multiple objects in static images. However, it is possible to use object detectors frame-by-frame of a video sequence and then employ some tracking method to find an association between the detected objects across the frames. This approach is known as tracking-by-detection and has been used in several works recently [1, 7, 40–45]. The main challenge of this approach is that the detectors usually present false positives and false negatives at their output. In other words, the output of the detectors are usually unreliable and sparse [46]. In this chapter, we present two ways of using the IPD in a tracking-by-detection scheme in order to reduce the false negative and false positive rates. In the first one, proposed in Section 4.1, we integrate the IPD in a particle filter framework to perform pupil tracking. In the second, we use a global model of landmarks on eyes to perform an integration between detection and tracking. This integration, that is based on temporal consistency and geometrical constraints, is proposed in Section 4.2.

In order to reduce the number of points at the detection output, we employed a cascade of IPD detectors [7], trained using the BioID dataset. Each stage of the cascade was designed to keep 95% of the positive training data. The goal is to keep only a small set of good candidates. A full description of how to design a cascade of IPD classifiers can be found in Section 3.4.

Five high definition (1080p) videoconference sequences with 300 frames each were employed to assess the tracking framework. These sequences have a moderate degree of compression artifacts and contain four distinct subjects and different types of background, movement and face occlusion. The sequence we refer to as "easy" has little movement of the subject and no occlusion. The one referred to as "intermediate/difficult" has blur, a subject with a moderate amount of movement and no occlusion. The other three sequences are considered difficult since they have blur, subjects with fast movements, and partial or total occlusion of the face by one of the hands. We manually annotated 13 fiducial points on the faces, including the pupils, in all 300 frames. The sequences and the manual annotations are available at http://www.smt.ufrj.br/∼biometria/etr [80]

4.1 IPD as a Prior for Tracking Using Particle Filters

In this section we investigate the use of the IPD detector in the particle filter framework. We start by briefly describing Bayesian state estimation and particle filters. A more thorough description can be found in Section 2.2 or in the literature [65–69]. Let $X = \mathbb{R}^M$ be a state space and $\mathbf{x}(n) \in X$ a state at a discrete time $n \in \mathbb{Z}$. Suppose that we know the states of a process from the beginning up to the present $(\mathbf{x}(0), \mathbf{x}(1), \cdots, \mathbf{x}(n))$ and want to estimate the next state $\mathbf{x}(n+1)$.

Modeling our states as random variables, the optimal estimate of $\hat{\mathbf{x}}(n+1)$ comes from the conditional density

$$\hat{\mathbf{x}}(n+1) = \arg\max_{\mathbf{x}(n+1)} \left\{ p(\mathbf{x}(n+1)|\mathbf{x}(0), \mathbf{x}(1), \cdots, \mathbf{x}(n)) \right\}. \tag{4.1}$$

Let $\mathbf{z}(n) \in \mathbb{R}^N$ be an indirect observation of our state at the discrete time n. The sequence of measurements until the present is $\mathbf{Z}(n) = \{\mathbf{z}(0), \cdots, \mathbf{z}(n)\}$. The model of the sensory system is defined by the conditional probability density $p(\mathbf{z}(n)|\mathbf{x}(0), \cdots, \mathbf{x}(n), \mathbf{z}(0), \cdots, \mathbf{z}(n-1))$. Assuming that our sensory system is memoryless, we can rewrite the conditional probability density of the measurements as

$$p(\mathbf{z}(n)|\mathbf{x}(0), \cdots, \mathbf{x}(n), \mathbf{Z}(n-1)) = p(\mathbf{z}(n)|\mathbf{x}(n)). \tag{4.2}$$

The online estimation starts with a prior distribution $p(\mathbf{x}(0))$ at the discrete time index $n = 0$, and after that, in each time index n, we can obtain the posterior density recursively. The first step of this recursion is the prediction, where the Bayes theorem combined with the Markov condition allows the calculation of $p(\mathbf{x}(n+1)|\mathbf{Z}(n))$. The last step of the recursion is the update, where we increment the variable n so that $p(\mathbf{x}(n+1)|\mathbf{Z}(n))$ becomes $p(\mathbf{x}(n)|\mathbf{Z}(n-1))$ and another measurement $\mathbf{z}(n)$ is collected in order to obtain the next posterior density $p(\mathbf{x}(n)|\mathbf{Z}(n))$. Since our sensory system is memoryless, we can use the Bayes theorem.

This recursion is known as Bayes, or Predictive, filtering [66, 67]. Since it provides the posterior density in each iteration, we need an optimality criterion to obtain an optimal estimate of the current state [65]. Under known constraints the posterior density function can be simplified. The Kalman filter is a classical example – it is the optimal solution when both state and measurement models are linear with Gaussian noises [65–67]. In other cases, the state and measurement models are linear, but the distributions of the states are not Gaussian. A remarkable approximation for this type of problem is the particle filtering, which implements a recursive Bayesian filter through Monte-Carlo simulations [65–67].

In a nutshell, the particle filter starts by randomly drawing K particles $\mathbf{x}^k(0)$ from the cloud of points and initializes their respective weights as $w^k(0) = \frac{1}{K}$. In the update step, it sets the importance weights as $w(\mathbf{x}^k(n))$ by doing $w(\mathbf{x}^k(n)) = p(\mathbf{z}(n)|\mathbf{x}^k(n))$, and calculates the normalized importance weights $w_{\text{norm}}(\mathbf{x}^k(n))$ by using $w_{\text{norm}}(\mathbf{x}^k(n)) = \frac{w(\mathbf{x}^k(n))}{\sum w(\mathbf{x}^k(n))}$. An estimation $\hat{\mathbf{x}}(n)$ is given by $\hat{\mathbf{x}}(n) = \sum_{k=1}^{K} w^k_{\text{norm}}(n)\mathbf{x}^k(n)$. It uses resampling to go from this importance-weighted sample back to a non parametric representation of equally weighted samples.

4.1.1 IPD as Parametric Density Estimation

In the first algorithm to integrate the IPD in the particle filter framework, we suppose that the cloud of points from the IPD output has a Gaussian distribution. Actually, if this assumption holds, a Kalman filter would be enough. However, in next section we will show how to integrate the IPD in the particle filter framework when the points from the cloud have a distribution with unknown shape (a more realistic assumption).

The state (or the particle) is the eye position (x, y) in pixels. At the instant n, the state is:

$$\mathbf{x}(n) = (x(n), y(n)). \tag{4.3}$$

In the initialization step, we choose $K = 300$ particles, that is, we pick up only 300 points from the cloud. According to the algorithm described early in this section, the normalized weights are computed in the update step. To do so, we estimate the average $\mu_c(n)$ and the covariance matrix $\mathbf{\Sigma}_c(n)$ using the complete cloud of points. After that, we compute the likelihood of the particles through a Gaussian modeled by these parameters, that is:

$$p(\mathbf{z}(n)|\mathbf{x}^k(n)) = \frac{\mathrm{e}^{\left[-\frac{1}{2}\mathrm{d}^k(n)^{\mathrm{T}}\mathbf{\Sigma}_c^{-1}(n)\mathrm{d}^k(n)\right]}}{\sqrt{(2\pi)^2|\mathbf{\Sigma}_c(n)|}}, \tag{4.4}$$

where

$$\mathbf{d}^k\,(n) = \mathbf{x}^k(n) - \boldsymbol{\mu}_c(n). \tag{4.5}$$

The normalized weights w_{norm} are obtained by normalizing the outcome of the above equation. The output of the algorithm $\hat{\mathbf{x}}(n)$ is the average of the particles $\mathbf{x}^k(n)$, weighted by $w_{\text{norm}}(\mathbf{x}^k(n))$. To maintain the stability of the algorithm, the covariance matrix $\mathbf{\Sigma}_c(n)$ needs to be full rank. We use a regularization factor to guarantee this property even when the cloud has a small number of points.

In the prediction step of the algorithm we use a random walk with Gaussian noise as follows:

$$\mathbf{x}^k(n+1) = \mathbf{x}^k(n) + \nu_{\text{P}}(n), \tag{4.6}$$

where $\nu_{\text{P}}(n) = \mathcal{N}(0, \mathbf{I}d_{\text{P}}^2)$. Here, $\mathbf{I}$ is a 2×2 identity matrix and d_{P} is a scalar. We find the best value of d_{P} with a leave-one-out cross-validation with grid search procedure in our video dataset [80]. The best average error for the test set was $e_{\text{P}} = 6.94$, with a standard deviation of $\sigma_{\text{P}} = 3.08$. In four of five folds, the value of $d_{\text{P}}^2 = 2.25$ produced the smallest error.

4.1.2 IPD as Nonparametric Density Estimation

In a parametric approach, as in Section 4.1.1, we use the resulting cloud of points to estimate the parameters of a known distribution. The nonparametric approach is even simpler: the whole cloud of points, weighted by their IPD normalized inner product results, represents the nonparametric distribution [103].

In a particle filter, we use the likelihood of the observation to find the weight of each particle before the resampling step. The traditional way to do so is to "interpolate" the density function using a kernel around each point of the nonparametric representation. Using a traditional Gaussian kernel, the unnormalized weights of particles $w(\mathbf{x}^k(n))$ are

$$w(\mathbf{x}^k(n)) = \sum_{i=1}^{M} c_i(n) \mathrm{e}^{\left[-\frac{1}{2}\mathbf{d}_i^k(n)^T (\mathbf{I}d^2)^{-1} \mathbf{d}_i^k(n)\right]}, \tag{4.7}$$

where

$$\mathbf{d}_i^k(n) = \mathbf{x}^k(n) - \mathbf{x}_i(n), \tag{4.8}$$

M is the number of points in the cloud, and $c_i(n)$ is the IPD normalized inner product value of the i-th point of the cloud $\mathbf{x}_i(n)$.

As in the parametric case (see Equation (4.6)), in our experiments we use a simple random walk in the prediction step with Gaussian noise $\nu_{\text{NP}}(n)$ to the particles

$$\mathbf{x}^k(n+1) = \mathbf{x}^k(n) + \nu_{\text{NP}}(n), \tag{4.9}$$

where $\nu_{\text{NP}}(n) = \mathcal{N}(0, \mathbf{I}d_{\text{NP}}^2)$ and d_{NP} was obtained in the same manner as in Section 4.1.1, with error $e_{\text{NP}} = 5.85$ with a standard deviation of $\sigma_{\text{NP}} = 2.23$. In three of five folds the smallest average error was obtained by using $d_{\text{NP}}^2 = 0.3$. The average errors together with the standard deviations of the two algorithms are shown in Table 4.1. There, it can be seen that the nonparametric algorithm has the error with smallest mean and standard deviation. In this case, the error is defined as the Euclidean distance between the manual annotation and the output of the system, expressed as a percentage of the interocular distance.

Table 4.1: Average relative error of the methods described in this paper for the high definition video sequences. The results were obtained by taking the average of the relative errors from the test sequences along the frames.

Method	error [%]	std
Parametric	9.58	5.03
Nonparametric	**7.70**	**2.98**

We show the relative error per frame for the two tracking methods in Figure 4.1. The curves labeled as “P” refer to the parametric method described in the previous section. The curves labeled “NP” are related to the non-parametric method described in this section. Each curve is the average error of the two eyes and the error was obtained by calculating the Euclidean distance between the manual annotation and the tracking result and expressing it as a percentage of the interocular distance. The gaps in the middle of the curves occur when there is no manual annotation, that is, when some occlusion occurs. Examples are when the subject is blinking or when his/her hand is in front of his/her head (this occurs in some frames between frames 50 and 150 of sequence 02 and between frames 200 and 250 of sequence 08). Peaks of error above 10% of the interocular distance are due to momentary loss of tracking, most cases in just one of the eyes. It can be noted that the errors for sequences 01 and 02 are larger. This is so because such sequences are more affected by blurring and compression artifacts. These facts are evidences of why the average error is larger for these sequences.

Figure 4.2 shows the visual tracking result of the nonparametric method for some frames of all sequences of our database[1]. The circles are the ground truth (manually annotated) and the crosses are the tracking results for the nonparametric method. The circles mark the position of the ground truth while the crosses mark the tracking result. The proposed nonparametric method is robust enough to keep tracking even in difficult conditions, such as total or partial occlusions (in the cases where the subject is blinking or putting the hand in front of the head), presence of blurring and compression artifacts. Note the absence of the reference in some frames. This is so because the sequences’ annotators where instructed not to mark a pupil if they were not sure of its position. It is also important to point out that the manual annotation is noisy in all sequences, which contributes to the graphs in Figure 4.1 not being smooth.

[1]The respective videos, as well their visual tracking results, can be watched at http://www.smt.ufrj.br/∼biometria/etr [80].

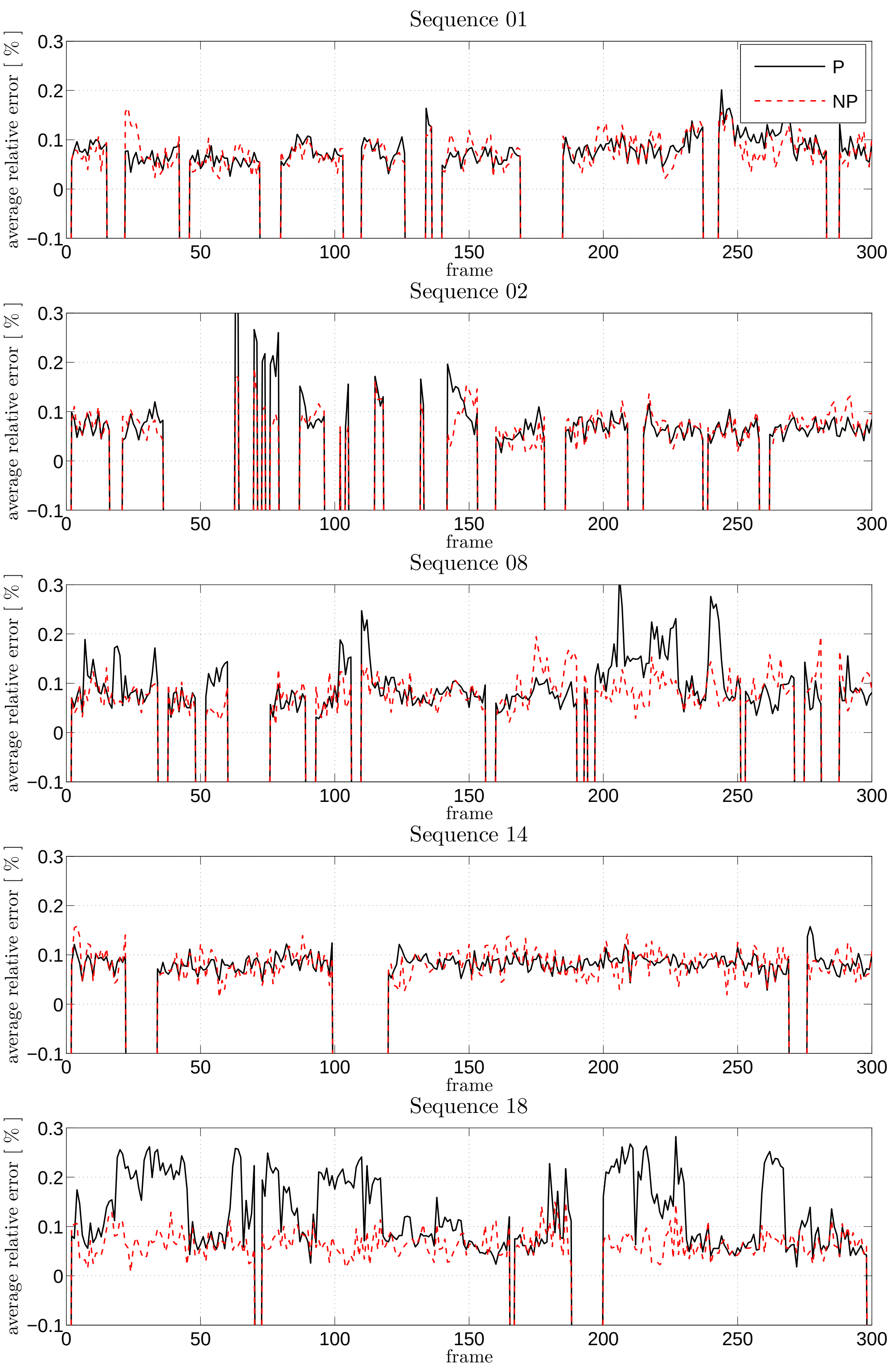


Figure 4.1: Results for the proposed methods on our high definition video sequence database. The error is the distance between the ground truth and the tracking result. Each curve is the average error of the two eyes.

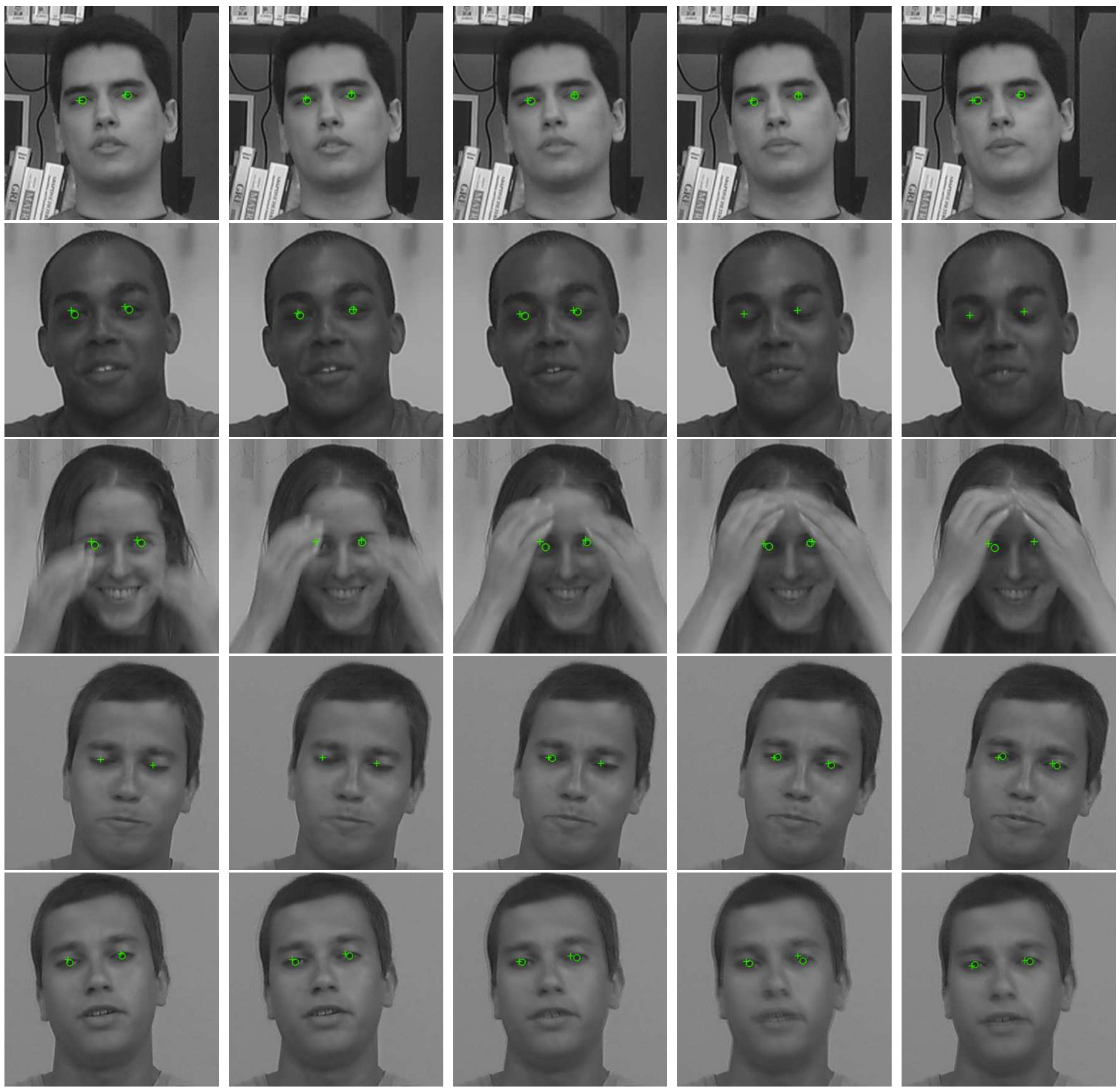


Figure 4.2: Qualitative results for the nonparametric method in our high definition sequences. Note the presence of different subjects, gender, skin color and background in the video sequences. The circles are indicating the ground truth (when available) and the crosses are indicating the tracking results.

4.2 Integration Between Detection and Tracking Using Temporal Consistency and Geometrical Constraints

A good way of improving robustness in video tracking is integrating detection and tracking [1, 7]. One contribution of this work is aligned with this idea. In this section, we describe a way of integrating detection and tracking in order to improve robustness of online eye tracking in video sequences.

One of the characteristics of the detectors employed in this work (the IPD's cascade and SVM with RBF kernel) is that they provide a cloud of points at their output. These points tend to be grouped in small clusters which are close to each other and highly correlated with the desired output [3, 104]. This cloud of points is used to feed a tracker (in this work we use the KL and the Particle Filter). Then, the outputs of the detection and tracking steps are combined to improve robustness.

4.2.1 Temporal Consistency

The temporal consistency has two parts: the histogram strategy and smoothness of the point's trajectory. In the histogram strategy, we use the outputs of the detector and the tracker to keep the reliable points. In addition, the trajectory of the tracked object should be smooth. In the smoothness of the point's trajectory, we propose a simple way to discard points that do not comply with a smooth trajectory. Both steps are described in the next sections.

Histogram Strategy

In the Histogram Strategy, we use the consistency between the detector and tracker outputs in order to discard unreliable detections. To do so, we have to compute histograms from the cloud of points as follows:

(i) Employ a Hierarchical Clustering algorithm [105] at the cascade's output.

(ii) Let j be the index of the frame. Compute the vector median of the cloud of points in order to obtain a single point y_j.

(iii) Feed the tracker with the point y_{j-1} detected in the previous frame and track it in the current one, to get the estimated point y'_j.

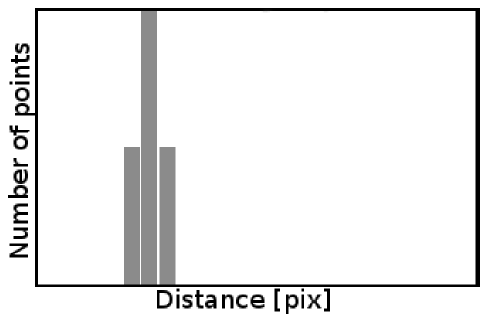
(iv) Let x_i $(i = 1, 2, \cdots, N)$ be a point in a cloud of N points and d_i the distance between the tracked point y'_j and a point x_i. Compute the histogram of the distances d_i.

(v) Analyzing the output of the Hierarchical Clustering algorithm and the histogram, we can distinguish a reliable detection from an unreliable one (see the description below about the histogram analysis). If the detection is reliable, use the vector median of the cloud y_j as output and go to step (ii). If the detection is not, do not provide an output and go to step (i).

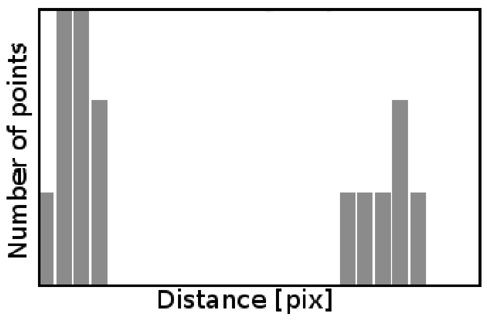
From the analysis of the typical histograms, we can distinguish four types of behaviors, which are illustrated in Figures 4.3(a) to 4.3(d). The histogram in Figure 4.3(a) is unimodal, which tends to correspond to a single cluster close to the desired pattern. This is an indication of a reliable detection. In practice, this behavior is automatically detected if the Hierarchical Clustering provides only one cluster and the variance of the histogram is smaller than a threshold T_h. The histogram in Figure 4.3(b) has two modes. One of them will in general correspond to a cluster close to the desired point (actually, the histogram have more than two modes in most cases, as long as they are clearly defined). This case is a candidate for a reliable detection. This is so because, as discussed later in this section, we can determine the desired cluster by analyzing the temporal behavior of the centroids of these clusters. In practice this behavior is automatically detected if the Hierarchical Clustering provides two or more centroids and the variance of each cluster is smaller than a threshold T_h. The other two typical histograms, depicted in Figures 4.3(c) and 4.3(d), indicate unreliable detections, and the corresponding points should be rejected. The histogram in Figure 4.3(c) has a "uniform" appearance (does not have any clear peak). Such behavior corresponds to a cloud with too many scattered points. In practice, this behavior is automatically detected if the Hierarchical Clustering provides clusters with variances larger than a threshold T_h. The histogram in Figure 4.3(d), that has only a few points, appearing as several modes, is generally related with isolated noisy points. Note that the range of the axis corresponding to the number of pixels varies dynamically. So, the number of points in (a), (b) and (c) are considerably bigger than the respective number in (d). Such situation also indicates an unreliable detection. In practice, this behavior is automatically detected if the number of points in the cloud is smaller than a threshold. We empirically evaluated values for the threshold from 1 up to 30. The best results were obtained when using $T_h = 10$ for all sequences.

Smoothness of the Path

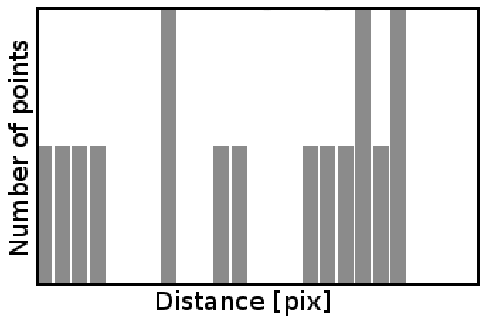
At this point we have a temporal sequence of clusters close to the desired pattern (note that, as a result of the histogram analysis, there can be some frames with no reliable clusters). We can choose the best cluster as well as discard further unreliable clusters by analyzing the temporal evolution of these clusters. We use the



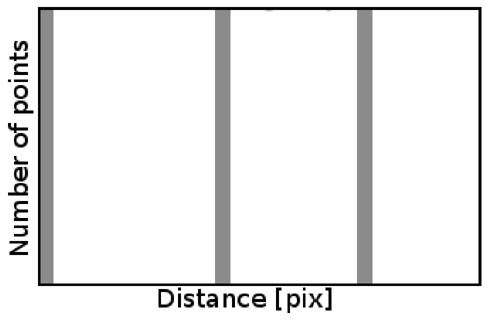
(a) Unimodal histogram.



(b) Bimodal histogram



(c) Uniform histogram.



(d) Histogram with few points.

Figure 4.3: Typical histograms obtained from an intermediate/difficult sequence for the inner corner of the right eye. The range of the axis corresponding to the number of pixels varies dynamically. So, the number of points in (a), (b) and (c) are considerably bigger than the respective number in (d).

strategy illustrated in Figure 4.4. The dots represent the points of the cloud and the crosses, the centroids of these clusters. These centroids are obtained by employing a Hierarchical Clustering algorithm [105] at the cascade's output.

In the example in this figure, we assume that the histogram analyses were made in all frames up to k and therefore all remaining clouds are reliable. The output point from a previous frame (illustrated as a square in frame $k-2$) is projected in the current one and we verify if there is a centroid in it that lies inside a disc centered at the projected point. When there is one or more of such centroid, we pick up the closest to the center of the disk and discard the remaining ones. The radius of the disk depends on the sequence. We experimented values from 1 up to 10 pixels. The best results were obtained by using disks with radius of 3 pixels for the easy sequences and 4 pixels for all the others. If no centroid is inside any disks from the last 5 frames, the cloud corresponding to the current frame is considered as unreliable, and is thus discarded.

The centroids of the clusters that remain reliable after the histogram and temporal consistency evaluation have a high probability of corresponding to the desired features. Therefore, we have a low false positive rate. However, many frames are marked as having unreliable outputs, which gives rise to a high false negative rate

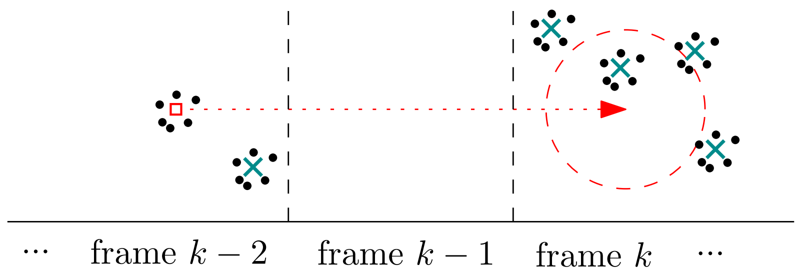


Figure 4.4: Temporal evolution of the cloud of points after the histogram analysis. The output from a previous frame (the square in frame $k-2$) is projected into the current frame (k). A disk of four pixels is centered at the projected point. The centroid in the current frame that lies inside such disk is considered as reliable and the those remaining are discarded. If more than one is in it, the closest to its center is chosen. Note that there is no reliable output at frame $k-1$.

(see the plots on the left part of Figures 4.10, 4.11 and 4.12). To overcome this problem we have devised a method which, whenever there is no reliable detection, can estimate the location of the missing points based on the geometry of the face. This is described in the next section.

4.2.2 Geometrical Constraints

As mentioned previously, the feature points of the eyes that we want to track are the left and right outer corners, left and right inner corners and the two pupils. Considering that real world faces are far enough from the camera, it is reasonable to suppose that such feature points lie on a plane. Then, the correspondence of eyes' feature points in two different frames can be described by a 2D homography $\mathbf{H}$ [106].

To obtain this homography $\mathbf{H}$, we assume that, between frames, the face is translated by $\mathbf{t} = (t_x, t_y)$ and is rotated by θ around an axis parallel to the camera's principal axis. In addition, we assume also a rotation around an axis orthogonal to the 3D scene's horizontal plane, that can be modeled as a scaling s along the camera plane's horizontal direction.

Using this motion model, the desired transformation has four degrees of freedom. As each point correspondence between frames puts two restrictions on the homography $\mathbf{H}$, we need two correspondences to determine $\mathbf{H}$. Since we can assume that the four corners of the eyes comprise a rigid body, we can use these points as references to obtain the desired homography. Therefore, if at least two eye corners have been reliably detected in the current frame, and there is a previous frame in which all eye corners have been reliably detected, the missed points from the current frame can be estimated.

Figure 4.5 depicts the $\mathbf{H}$ parameters $\mathbf{t}$, θ and s, where $\mathbf{x}_i$ is a point in a previous frame and $\mathbf{x}'_i$ is its correspondence in the current one.

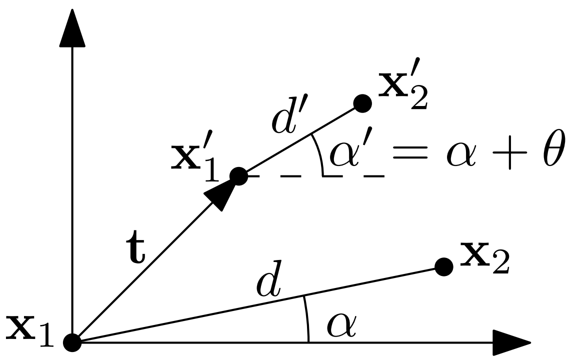


Figure 4.5: Parameters necessary to compute the homography $\mathbf{H}$ between eye corners in two frames. Note that putting the reference on $\mathbf{x}_1$, we have: $\mathbf{t} = \mathbf{x}'_1$, $\theta = \alpha' - \alpha$ and $s = d'/d$.

After obtaining the parameters in Figure 4.5, a missed corner can be estimated using the expression

$$\mathbf{x}'_i = \mathbf{H}_r(\theta)\mathbf{H}_r(\alpha)\mathbf{H}_s\mathbf{H}_r(-\alpha)\mathbf{x}_i + \mathbf{t}, \tag{4.10}$$

where

$$\mathbf{H}_r(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma \\ \sin\gamma & \cos\gamma \end{bmatrix}, \qquad \mathbf{H}_s = \begin{bmatrix} s & 0 \\ 0 & 1 \end{bmatrix}. \tag{4.11}$$

When there are more than two correspondences, each pair is used to compute a different homography. The missed point is the average of the points obtained employing each possible homography.

Since the pupils can move relatively to the eyes' corners, we have to use a different geometrical model for them. It is based on the reasonable supposition that the distance between the pupils is constant, and also that the distances between them and the respective inner or outer corners remains constant. We can estimate the position of a missing pupil in the current frame provided that we know: (i) the position of the eyes' corners in both frames; (ii) the position of the pupils in the previous frame, and (iii) the position of one pupil in the current frame. This model is illustrated in Figure 4.6. In this illustration, the right pupil is represented by a cross and the left pupil by a square. The coordinates (δ_x, δ_y) of the pupil on the right eye relative to the outer right eye corner are the same as the coordinates of the left pupil relative to the left inner eye corner. These coordinates can be used to determine the missing pupil position.

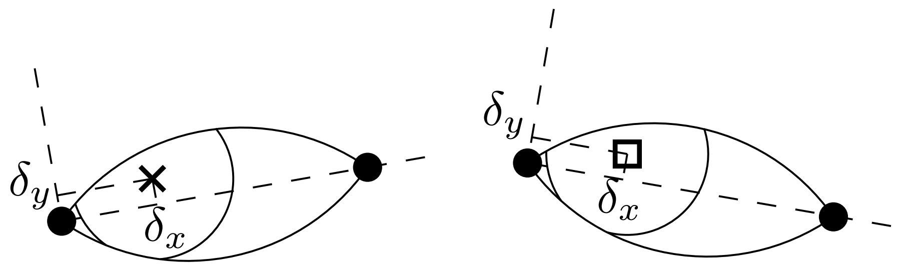


Figure 4.6: Geometric model for the location of the pupils.

Evaluating the Consistency of the Geometrical Model

The Temporal Consistency and Geometrical Constraints presented so far are capable of eliminating unreliable detections while keeping the false positive and false negative ratios. However, if the output of the detection system is consistently wrong along the frames, we would obtain a wrong estimate as well. This can lead to an increase in error rate. Since the homograpies are used to estimate the missing points, we need a way to evaluate their quality. In this section we present a strategy that can be used to evaluate the quality of the estimated points and, as a consequence, the quality of the homographies.

Once we have detected the four corners in the current frame and in a previous one, we can obtain up to six homographies, one for each pair of correspondent points. Each homography leads to a mapping that takes a model point $\mathbf{x}_i$ to a point $\mathbf{x}'_i$ in the current frame. We can use these mappings to obtain estimates $\hat{\mathbf{x}}_{i,n}$ ($n = \{1, \cdots, N\}$ with $N \leq 6$ homographies). Since the homographies are not exactly the same, each estimate $\hat{\mathbf{x}}_{i,n}$ is different. Knowing this, one can evaluate the quality of the geometrical model and obtain better estimates $\hat{\mathbf{x}}_i$ by using Algorithm 2.

The Algorithm 2 can produce an estimate better than the one obtained with the Geometrical Constraints alone and a model that can adapt to a subject facial geometry. Whenever there are missing points, we use the best model up to this moment to obtain the estimates. However, it is possible that an unreliable point passes through the temporal consistency - an error that has some temporal consistency. In this case, even using the best model (the one that produces $e_H = 0$), the produced estimate is wrong. This is so because some of the homographies are not good enough. There are three possible scenarios: i) If there are three or four missing points, there is nothing that can be done since it is not possible to obtain an homography; ii) with two missing points, we have one homography. Even in this case the described algorithm is not applicable; iii) In case of one or no missing point, we have three and six homographies, respectively. If that happens, we can benefit from the algorithm described in this section. We empirically evaluate values for threshold T_H (step 4) from 1 up to 5 pixels. The value that leads to the best hit rate in most of the sequences is $T_H = 3$. We present the results for the high

```
input  : The model points $\mathbf{x}_i$, the points in the current frame $\mathbf{x}'_i$
         $(i = \{1, \cdots, 4\})$, the homographies $\mathbf{H}_n$ $(n = \{1, \cdots, N\}$ with $N \leq 6)$,
         a Threshold $T_H$ and the global error $e_H$.
1  begin
2      for $n = 1 \to N$ do
3          Get an estimate $\hat{\mathbf{x}}_{i,n} = \mathbf{H}_n \mathbf{x}_i$
4          Get the error $e_{i,n} = \|\hat{\mathbf{x}}_{i,n} - \mathbf{x}'_i\|$;
5      Calculate the local error $e_h = \frac{1}{4N} \sum_i \sum_{n=1}^{N} e_{i,n}$;
6      if $e_h < e_H$ and $N = 6$ then
7          Do $\mathbf{x}_i = \mathbf{x}'_i$ to update the model.
8          Do $e_H = e_h$ to set a new global error.
9      if $e_h < T_H$ then
10         Use the estimates $\hat{\mathbf{x}}_i = \frac{1}{N} \sum_{n=1}^{N} \hat{\mathbf{x}}_{i,n}$.
11     else
12         Keep the points with temporal consistency doing $\hat{\mathbf{x}}_i = \mathbf{x}'_i$.
13 end
output: The global error $e_H$ and the final estimates $\hat{\mathbf{x}}_i$
```
Algorithm 2: Evaluation of the geometrical model.

definition video sequences in next section.

4.2.3 Experiments Using a High Definition Video Sequence Database

To assess the performance of our method, we employ the local accuracy measure described in Equation (3.33), Section 3.6.1. In addition, both IPD and SVM classifiers used in the video sequences were trained using the BioID database. The training procedures are the same as described in Section 3.6.1. The method described in this chapter was implemented in C++ using the OpenCV library, and can run in real time on a fast PC, as specified on Page 47.

As can be seen in Section 3.6.1 (Figure 3.6) the RoIs for features on the right side of the face can be quite different from the ones on left side. However, as can be seen in the Appendices A and B, the detection results tend to be equivalent for the left and right eyes. So, in this section we only show the results for the right side of the face. The remaining curves (all points of all sequences) can be seen in Appendix B. First of all, note that the experiments with SVM were not included in the pictures. We decided to do this because the SVM classifiers have a very small hit rate in our HD database. Since SVM is a strong classifier, it was not able to generalize enough to be used with our HD Sequences.

The results in Figures 4.7, 4.8 and 4.9 were obtained using an easy, an intermediate/difficult and a hard sequence, respectively. Unlike the rest of this section, in this case we are showing the results for both eyes. This is so because in this way the reader can better appreciate how a reliable detection of one eye landmark can help correcting the detection of the other. In both figures, we compare the IPD (referred to as IPD in our plots), IPD feeding a Particle Filter using Temporal Constraints (referred to as IPD + TC) and the IPD with Particle Filter using Temporal and Geometrical Constraints (referred to as IPD + TGC in our plots). In all graphs, we considered both false positives and false negatives as errors. The hit rate curves in the plots show that there is a decrease in hit rate when we employ the Temporal Constraints (actually a dramatic decrease in some cases). This is related with the increase in false negative error. As a matter of fact, this is expected, since the detection in many frames is considered as unreliable after using the Histogram Strategy and the Smoothness of the Path. However, since the remaining points have a good reliability, the easy and intermediate/difficult sequences can benefit from using the Geometrical Constraints (together with the geometrical model consistency evaluation - see Section 4.2.2). In both cases, the remaining points that were considered as reliable detections were enough to assure an increase in robustness and, consequently, in hit rate. This fact is even more evident in the graphs corresponding to the intermediate/difficult sequence. The same does not occur for the hard sequences (Figure 4.9). This is so because there is a large number of frames where no reliable points are detected in these sequences. As a consequence, the results are not good for the hard sequences, regardless of using the temporal and geometrical constraints, or the evaluation of the homographies. This suggests that one should not exceed a maximum error rate to benefit from these constraints. However, is important to point out that in many practical cases the sequences used will be more like the easy and the intermediate/difficult sequences. In addition, the majority of known methods also tends to fail in our difficult sequences (since they have blur, fast movements, and occlusion by one of the hands).

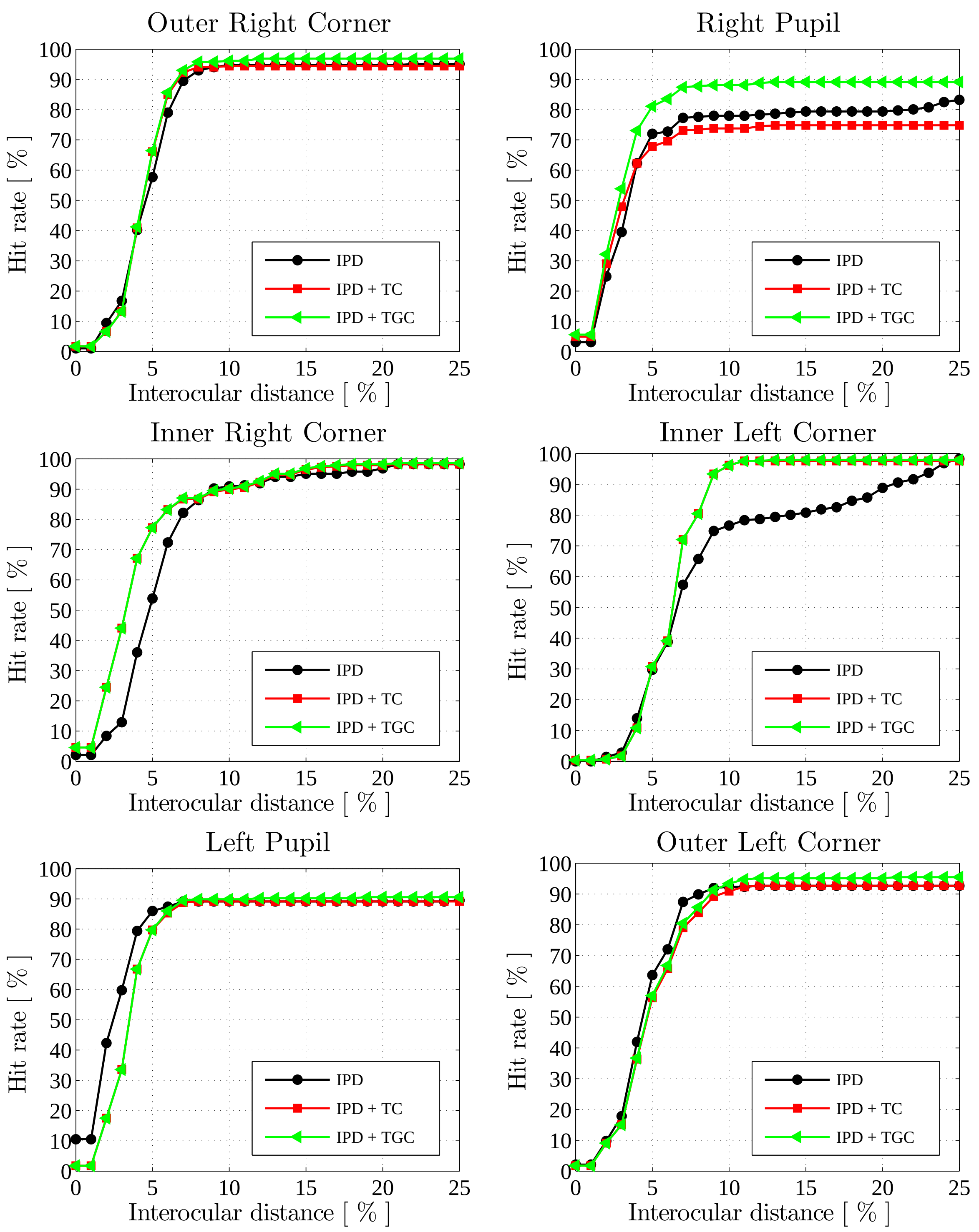


Figure 4.7: Comparison between IPD (referred to as IPD), IPD feeding a Particle Filter using Temporal Constraints (referred to as IPD + TC) and the IPD with Particle Filter using the Temporal and Geometrical Constraints (referred to as IPD + TGC). The results were obtained using an easy sequence.

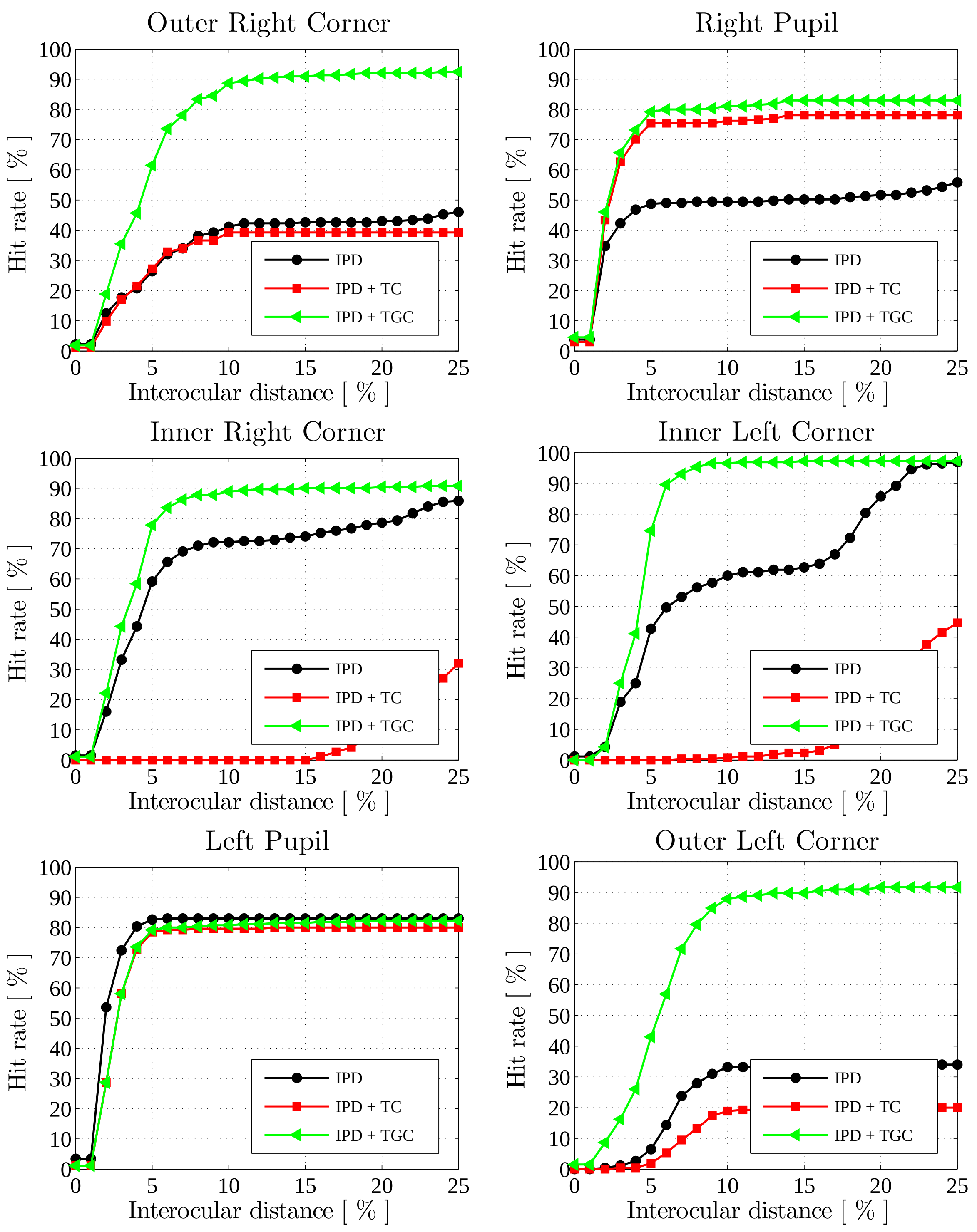


Figure 4.8: Comparison between IPD (referred to as IPD), IPD feeding a Particle Filter using Temporal Constraints (referred to as IPD + TC) and the IPD with Particle Filter using the Temporal and Geometrical Constraints (referred to as IPD + TGC). The results were obtained using an intermediate/difficult sequence.

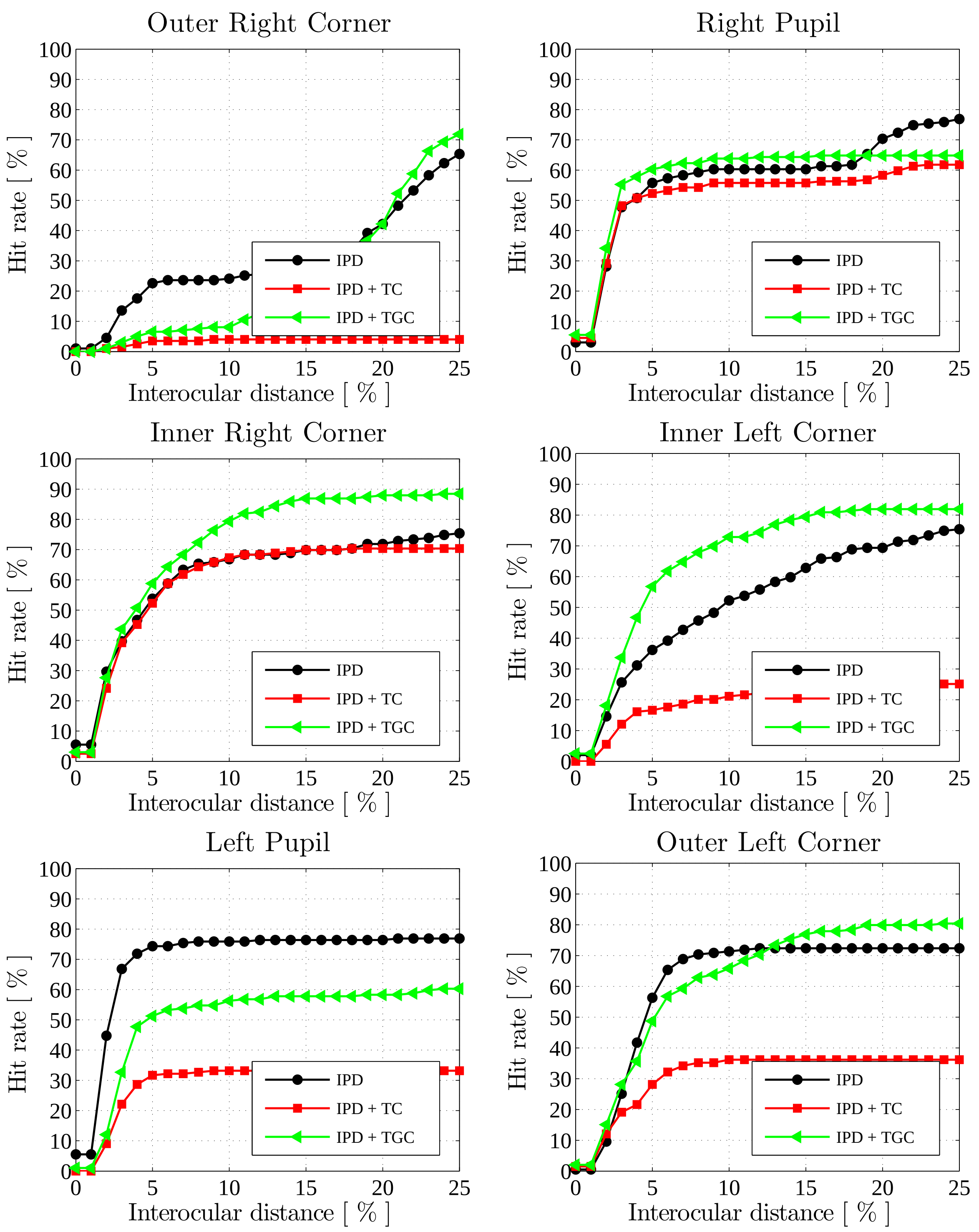


Figure 4.9: Comparison between IPD (referred to as IPD), IPD feeding a Particle Filter using Temporal Constraints (referred to as IPD + TC) and the IPD with Particle Filter using the Temporal and Geometrical Constraints (referred to as IPD + TGC). The results were obtained using a hard sequence.

In Figures 4.10 to 4.12, we evaluate the behavior of the errors when using only the Temporal Constraints (referred to as TC in our plots) and using both the Temporal and Geometrical Constraints (referred to as TGC). To do so we plot, for each method, two different definitions of hit rate. One considers only False Positives (referred to as FP in our plots) as errors and other considers both False Positives and False Negatives (referred to as FP + FN) as errors. These plots show that the use of histogram analysis and temporal consistency was capable of providing a low false positive rate, but at the expense of a high false negative rate. By comparing them with the plots for the case when Geometrical Constraints are added, we can see that the addition of geometrical constraints significantly decrease the false negative rate, while maintaining the low false positive rate. The results are consistent for both easy (Figure 4.10) and intermediate/difficult (Figure 4.11) sequences. However, the analysis of the results for the hard sequence (Figure 4.12) shows that, although the use of the histogram analysis and temporal consistency are able to provide a reasonably low false positive rate, the use of geometric consistency is not as effective as in the cases of easy and intermediate/difficult sequences. The reason is the same as stated before: there is a large number of frames where no reliable points are detected. Also, since the false positive rates are not as low as in the easier cases, points in error can lead to wrong geometrical models, which could place a wrong point in place of a missing one, increasing the false positive rates.

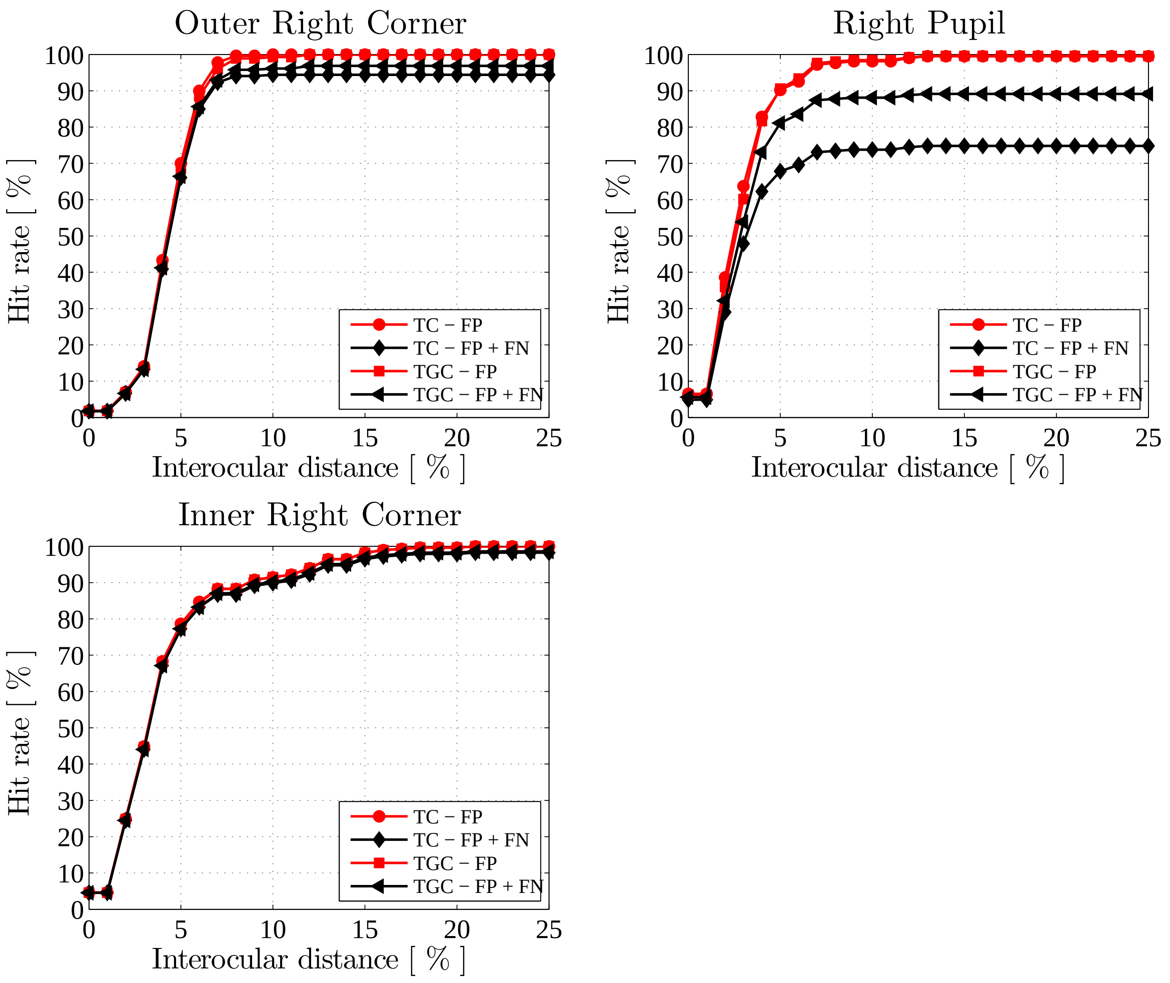


Figure 4.10: Results for an easy sequence. FP stands for False Positive and FN stands for False Negative. The curves refereed to as TC were obtained by using Temporal consistency only. The curves referred to as TGC were obtained by employing both Temporal and Geometrical Constraints (together with the strategy to evaluate the obtained model)

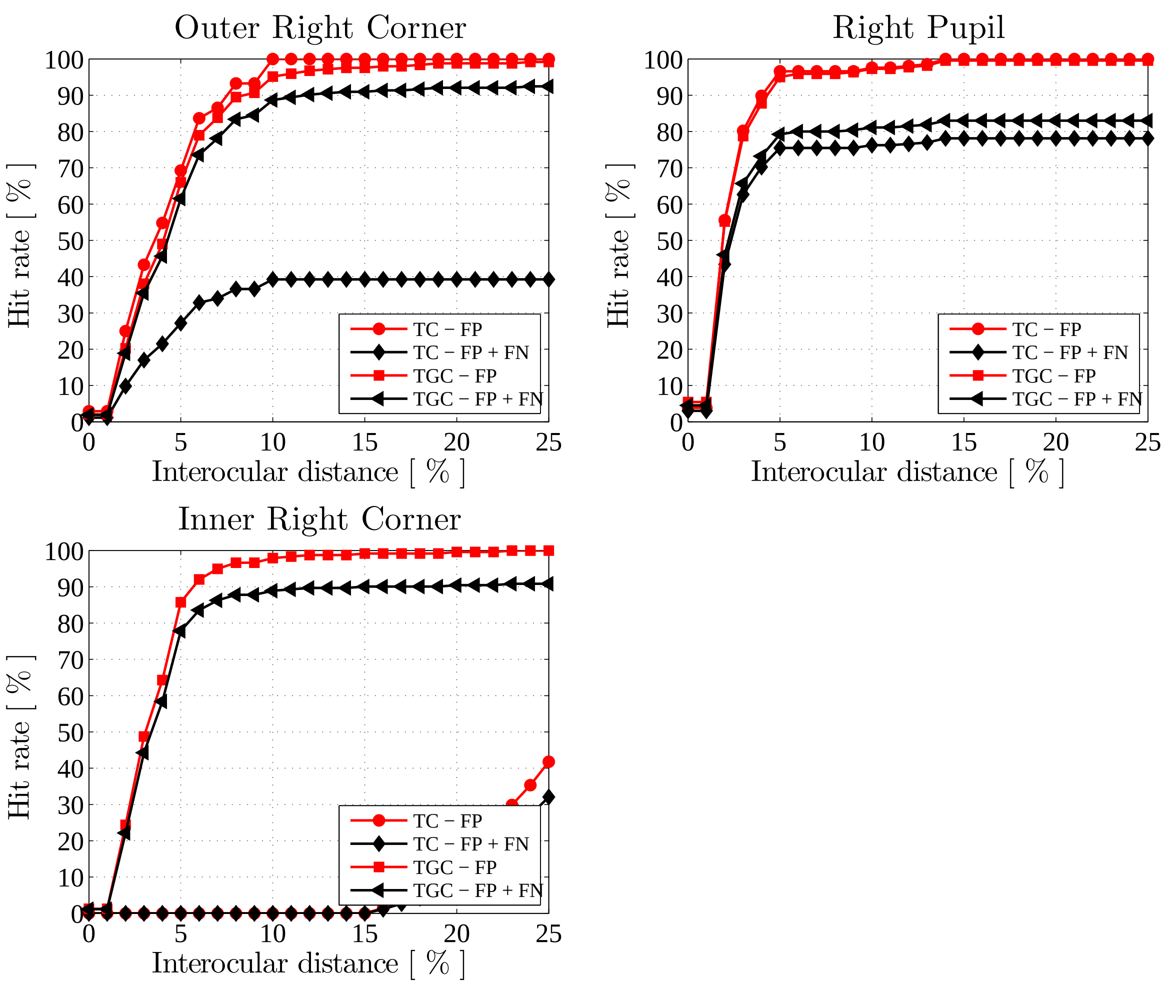


Figure 4.11: Results for an intermediate/difficult sequence. FP stands for False Positive and FN stands for False Negative. The curves referred to as TC were obtained by using Temporal consistency only. The curves referred to as TGC were obtained by employing both Temporal and Geometrical Constraints (together with the strategy to evaluate the obtained model)

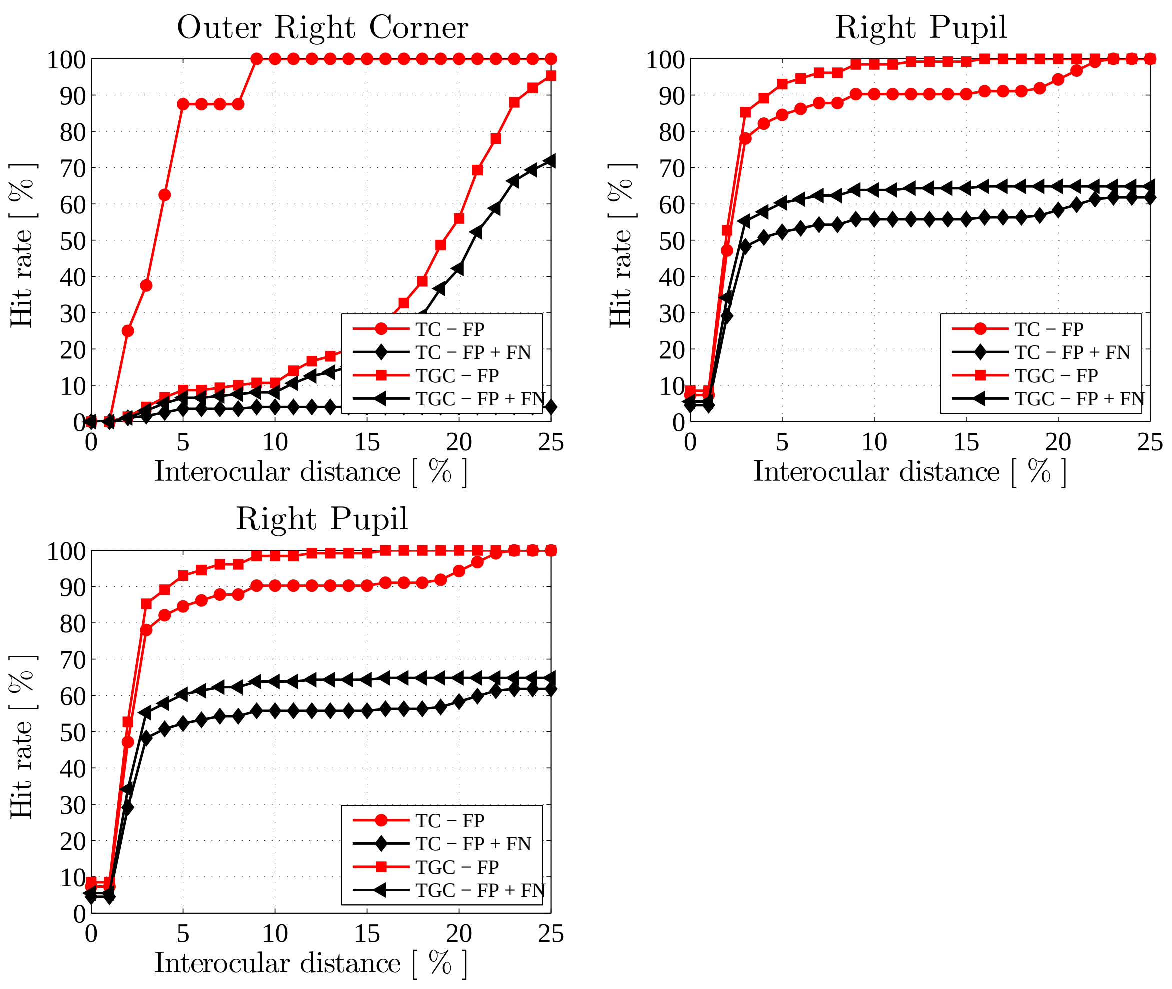


Figure 4.12: Results for a hard sequence. FP stands for False Positive and FN stands for False Negative. The curves referred to as TC were obtained by using Temporal consistency only. The curves referred to as TGC were obtained by employing both Temporal and Geometrical Constraints (together with the strategy to evaluate the obtained model)

Our method was designed to work with different types of trackers. From Figures 4.13 to 4.15, we evaluate the impact of changing the tracking method. In both figures, we compare the IPD (referred to as IPD in our plots), IPD feeding a Kanade-Lucas tracker using Temporal Constraints (referred to as IPD + TC) and the IPD with kanade-Lucas using Temporal and Geometrical Constraints (referred to as IPD + TGC in our plots). In all plots, we considered both false positives and false negatives as errors. In Figures 4.13 and 4.14 we have the results for an easy and an intermediate/difficult sequence, respectively. There is no significant difference between these results and the ones in Figures 4.7 and 4.8, in which a Particle Filter was used. This is a strong evidence that our method can work well with different types of trackers. Since our method as stated above is not adequate for hard sequences, the results for hard sequences in Figure 4.15 are inconclusive.

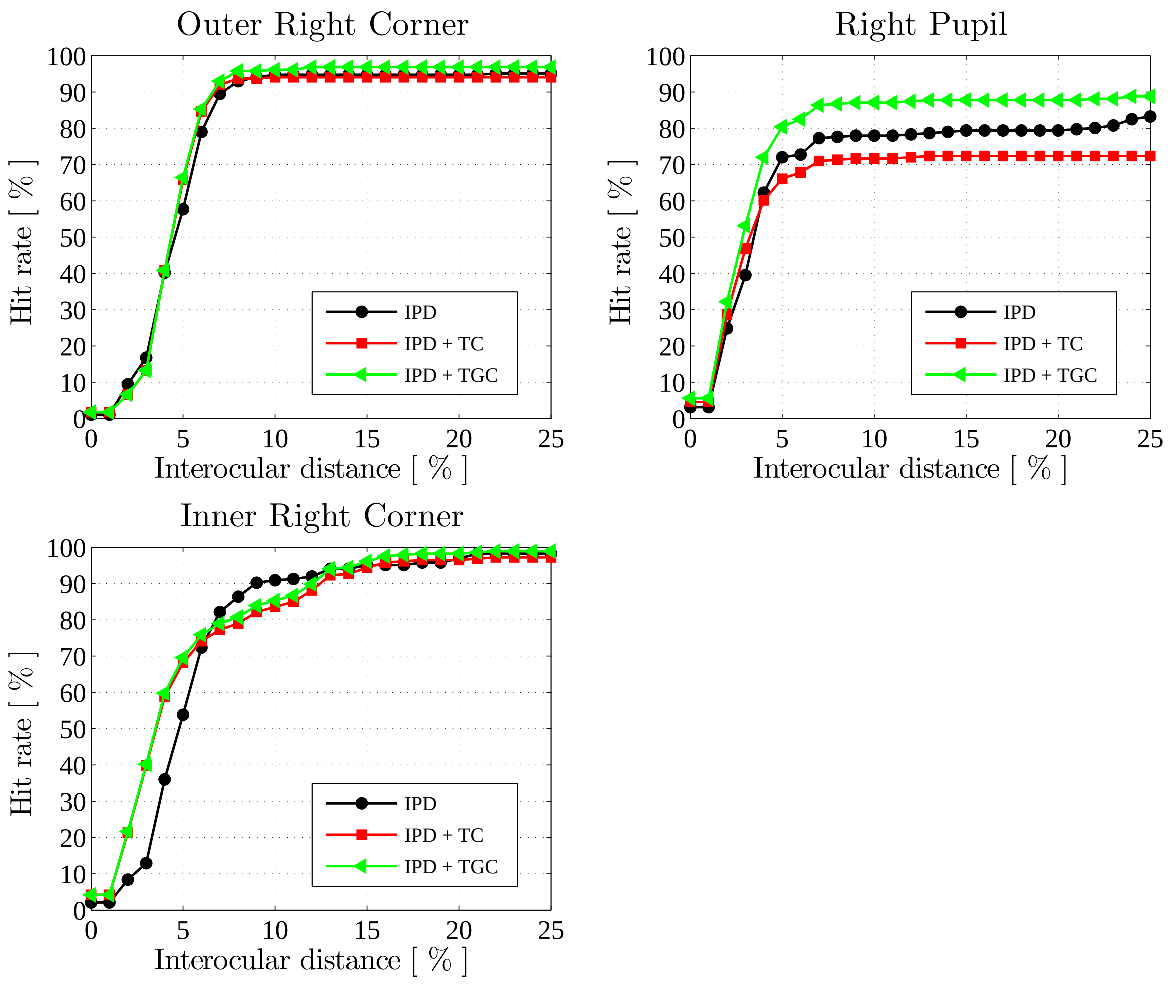


Figure 4.13: Comparison between IPD (referred to as IPD), IPD feeding a Kanade-Lucas tracker using Temporal Constraints (referred to as IPD + TC) and the IPD with Kanade-Lucas using the Temporal and Geometrical Constraints (referred to as IPD + TGC). The results were obtained using an easy sequence and correspond to the right eye's landmarks.

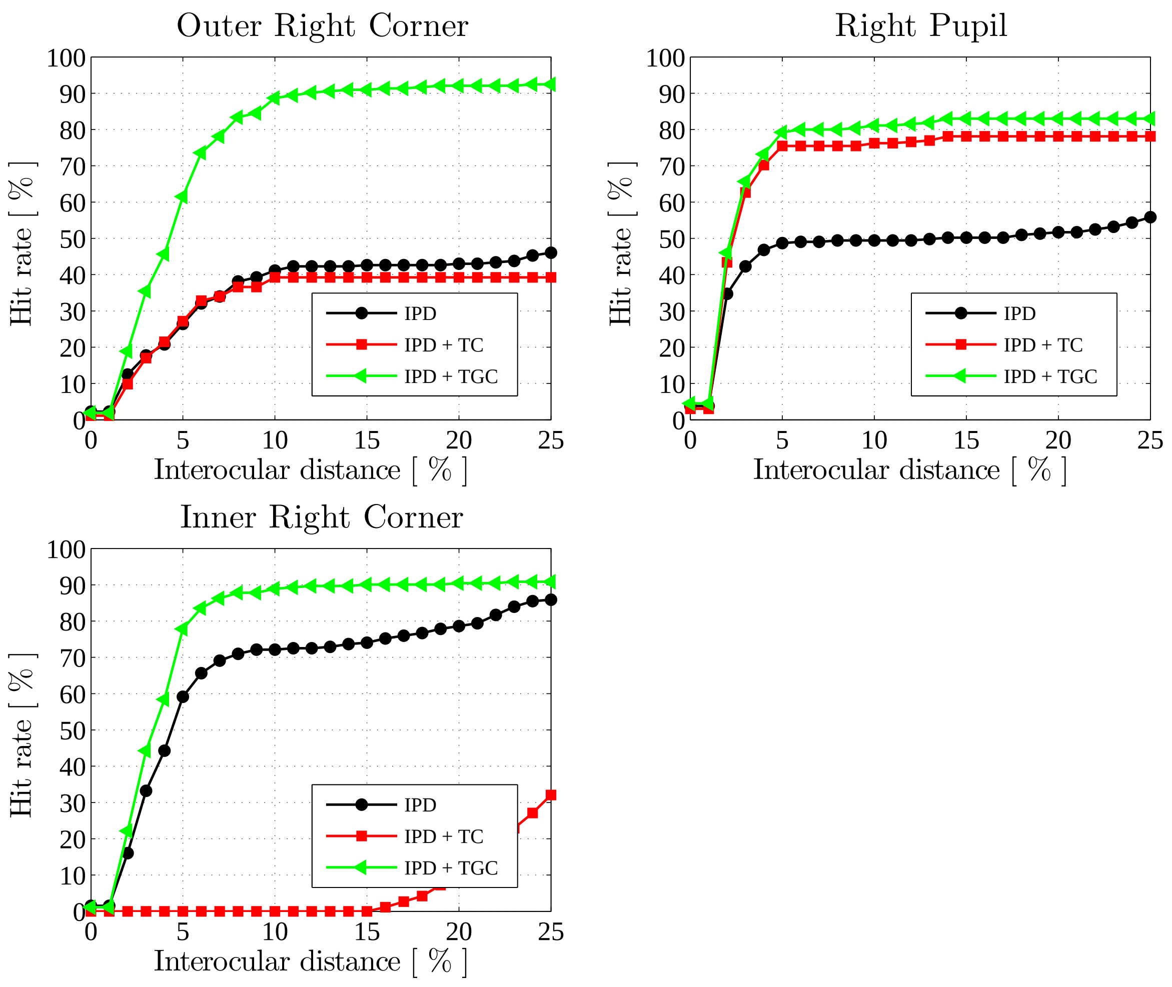


Figure 4.14: Comparison between IPD (referred to as IPD), IPD feeding a Kanade-Lucas tracker using Temporal Constraints (referred to as IPD + TC) and the IPD with Kanade-Lucas using the Temporal and Geometrical Constraints (referred to as IPD + TGC). The results were obtained using an intermediate/difficult sequence and correspond to the right eye's landmarks.

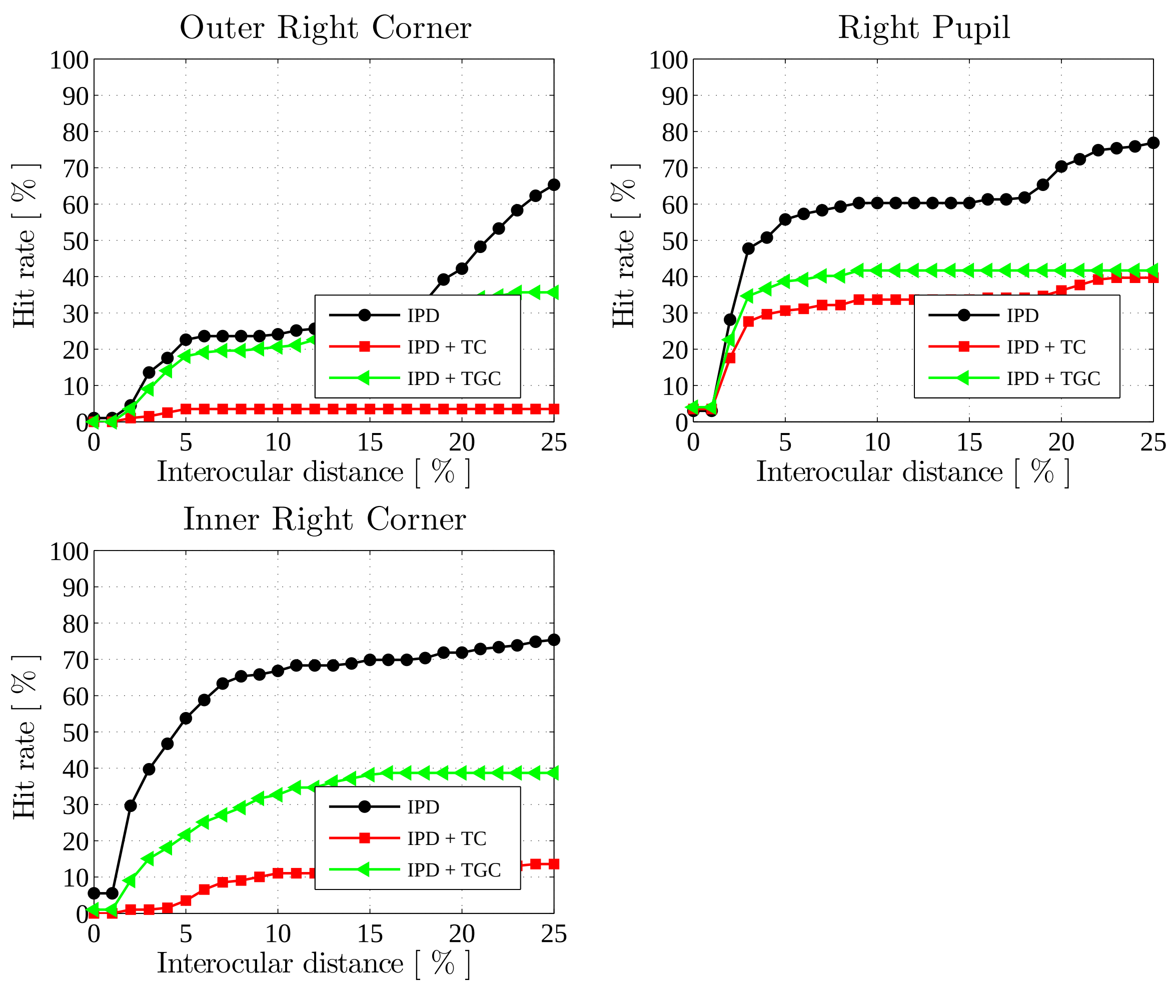


Figure 4.15: Comparison between IPD (referred to as IPD), IPD feeding a Kanade-Lucas tracker using Temporal Constraints (referred to as IPD + TC) and the IPD with Kanade-Lucas using the Temporal and Geometrical Constraints (referred to as IPD + TGC). The results were obtained using a hard sequence and correspond to the right eye's landmarks.

Chapter 5

Conclusions and Future work

In this work, we addressed the problem of robustly detecting and tracking facial landmarks. We used two types of detectors, the Support Vector Machine (SVM) with RBF kernel and the Inner Product Detector (IPD). Both trainings were performed in the BioID database. We also used the Kanade-Lucas (KL) and the Particle Filter (PF) in order to find temporal correlation between detections along the frames and then perform tracking. Strategies like the ones proposed in this work are known in the literature as tracking by detection.

We have contributions in the detection part and in the integration between detection and tracking. In the detection, the contributions are:

(i) the changes that we have made in the discriminant function makes the IPD more restrictive. It allows the selection of a smaller volume of the space – a desirable feature when the classes are unbalanced. Such unbalancing is exactly the case of feature/object detection in images;

(ii) we conducted several experiments in order to verify the effect of the detector size in the performance of the IPD;

(iii) the stopping criterion used to automatically determine the number of stages of the IPD cascade.

We also proposed novel approaches to combine detection and tracking:

(i) we used the IPD to find the probability distribution function of desired features in images. There are two variations of such method, one parametric and other nonparametric. They are simple, fast and fit in a Bayesian framework, such as the Kalman or particle filter, for tracking the features in video sequences;

(ii) in another contribution, based on the analysis of the histogram of the distance between tracking and the detection, as well as temporal consistency, we can find candidate points with low probability of false positives and reasonable accuracy. However, this is obtained at the expense of a high false negative rate;

(iii) other contribution of this paper aims at reducing this false negative rate by employing geometrical consistency between frames. We use the geometry of the face to impose localization constraints and recover some missed points;

(iv) as a last contribution in tracking part, we proposed a method to periodically evaluate the consistency of the model and avoid tracking wrong features that have some temporal consistency.

Also, in this work we have provided indirect, but useful contributions to the research community. We manually annotated 13 landmarks in all 2003 images from FERET in which subjects were in near frontal pose and made available in the web. In addition, we also have made available the video sequences used in our experiments, along with the manual annotation of 13 landmarks on the face for all 300 frames of each sequence.

In the detection part, the IPD has shown competitive results. It can detect eyes in 88.3% of the BioID images with an error less than 5% of the interocular distance. This result is comparable to the ones of other methods in the literature, with the added advantage of not using a global face model. We also show that the IPD is tolerant to small variations of the desired pattern, has good generalization potential, is fast enough to be used in real time applications, and can be implemented in parallel. Although the SVM could reach a higher hit rate in the BioID database, the IPD performed better in our high definition video sequences.

We also obtained promising results in the tracking part. When integrating the IPD with particle filter to track local features (the pupils, more specifically), we obtained an average relative error of 7.7% of the interocular distance in our high definition dataset. On the other hand, our global method to jointly track a set of features on eyes performs well for easy and intermediate/difficult sequences. It needs to be improved especially to deal with hard sequences, where we have a great deal of fast movement, blur and occlusions. However, in many practical cases the sequences used will be more like the easy and the intermediate/difficult sequences. In addition, the majority of known methods also tend to fail in our difficult sequences (due to blur, fast movements, and occlusion by one of the hands).

It is important to point out that the integration of detection and tracking procedure developed in this work can be used with most other tracking methods. It can also be used with other types of detectors, as long as they can be adapted to output a cloud of points instead of a single point. Note that these results are achieved with reasonably low computational complexity. The proposed detectors have been implemented in C++ using the OpenCV library, and run in real time on a fast PC.

5.1 Future work

In Chapters 3 and 4, we have described our method for object detection and tracking in video sequences. We also have shown competitive detection/tracking results. However, some modifications can improve the reliability of our methods. In this section, we introduce these suggested improvements. In Section 5.2 we present a pre-processing technique that can improve the hit rate in the classification step. In Section 5.3, we describe a possible way to improve our state model and obtain better results when tracking with the Particle Filter. In Section 4.2, we employed a geometric constraint to infer the position of missing points. Unfortunately, this strategy cannot be generalized to be used with generic shapes. So, in Section 5.4, we suggested an alternative that could allow it.

5.2 Inner Product Detector and Histogram of Oriented Gradients

As described in Chapter 3, before detecting the landmarks in the pixel domain, we employed the pre-processing described in [81, 82]. This pre-processing is an illumination normalization method that consists in the following sequence of techniques: Gamma Correction, Difference of Gaussian (DoG) filtering and Contrast Equalization.

On the other hand, we can improve the hit rate if we extract more descriptive features and detect the landmarks in that feature space. Even better if this feature descriptors could be invariant to illumination changes or shadowing. The Histogram of Oriented Gradients (HOG) has these characteristics [107]. To obtain the HOG descriptors, we first split the image in cells and compute one-dimensional histograms of gradient directions over the pixels of each cell. After that, the histograms are contrast-normalized by a measure of intensity in blocks larger than the cells. In other words, the contrast is normalized over overlapping spatial blocks The descriptors are a combination of the normalized histograms

5.3 Integration Between Particle Filters and IPD

In Section 4.1, we have shown that each landmark in each frame has a cloud of points at the cascade's output. Each point of this cloud has a respective inner product value. This scalar can be viewed as a confidence rate. We also have shown how to use this information to find the probability distribution function of desired features' in images.

On the other hand, our state modeling consists only in the position of the features and the velocity and acceleration are considered as noise. Our method can benefit from incorporating velocity in our space state model while keeping the acceleration as noise. In addition, we just employ a simple random walk to obtain the estimate used as output of our method. The prediction step can also provide a more robust outcome if we employ a more elaborate state modeling.

5.4 A Global Method for Locating a Set of Landmarks

The use of geometric constraints in our integration method is a good way for estimating the missing points. However, it is strongly related with the problem of eye tracking and cannot be generalized for using with generic shapes. An alternative way would be to use a generic deformable template for this task.

The method in [108], or some variation of it, has a potential to be used with our work. It represents the object by a set of parts arranged in a deformable template. Each part is modeled separately and the template is composed by spring-like connections. They also address the problem of learning the model from the samples and matching it through energy minimization or MAP estimate.

Bibliography

[1] KALAL, Z., MIKOLAJCZYK, K., MATAS, J. “Tracking-Learning-Detection”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 34, n. 7, pp. 1409–1422, July 2012.

[2] DA SILVA JÚNIOR, W. S. *Reconhecimento de padrões utilizando filtros de correlação com análise de componentes principais.* Tese de D.Sc., Universidade Federal do Rio de Janeiro (COPPE/UFRJ), Rio de Janeiro, RJ, Brasil, 2011.

[3] ARAUJO, G. M., DA SILVA JÚNIOR, W. S., DA SILVA, E. A. B., et al. “Facial Landmarks Detection Based on Correlation Filters”. In: *International Telecommunications Symposium (ITS)*, pp. 1–5, 2010.

[4] ARAUJO, G. M. *Algoritmo para reconhecimento de características faciais baseado em filtros de correlação.* M.Sc. dissertation, Universidade Federal do Rio de Janeiro (COPPE/UFRJ), Rio de Janeiro, RJ, Brasil, 2010.

[5] RIBEIRO, F. M. L., ARAUJO, G. M., DA SILVA, E. A. B., et al. “Detecção de pontos fiduciais sobre a face em tempo real”. In: *XXX Simpósio Brasileiro de Telecomunicações (SBRT)*, pp. 1–5, 2012.

[6] ARAUJO, G. M., RIBEIRO, F. M. L., SILVA, E. A. B., et al. “Fast Eye Localization Without a Face Model Using Inner Product Detectors”. In: *IEEE International Conference on Image Processing (ICIP)*, pp. 1366–1370, 2014.

[7] ARAUJO, G. M., SILVA, E. A. B., CIANCIO, A. G., et al. “Integration of Eye Detection and Tracking in Videoconference Sequences Using Temporal Consistency and Geometrical Constraints”. In: *IEEE International Conference on Image Processing (ICIP)*, pp. 421–424, 2012.

[8] ARAUJO, G. M., SILVA, E. A. B., RIBEIRO, F. M. L., et al. “Rastreamento Robusto de Olhos Usando Consistência Temporal e Restries Geométricas”. In: *XXXI Sompósio Brasileiro de Telecomunicações (SBRT)*, pp. 1–5, 2013.

[9] LI TIAN, Y., KANADE, T., COHN, J. F. "Recognizing Action Units for Facial Expression Analysis", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 23, pp. 97–115, 2001.

[10] ZHAO, S., GAO, Y. "Automated Face Pose Estimation Using Elastic Energy Models". In: *Proceedings of the International Conference on Pattern Recognition*, pp. 618–621, Washington, DC, USA, 2006.

[11] FANG, T., ZHAO, X., OCEGUEDA, O., et al. "3D facial expression recognition: A perspective on promises and challenges". In: *Proceedings of the IEEE International Conference on Automatic Face Gesture Recognition*, pp. 603–610, Santa Barbara, CA, USA, march 2011.

[12] NAIR, P., CAVALLARO, A. "3D Face Detection, Landmark Localization, and Registration using a Point Distribution Model", *IEEE Transactions on Multimedia*, v. 11, n. 4, pp. 611–623, 2009.

[13] AL-RAHAYFEH, A., FAEZIPOUR, M. "Eye Tracking and Head Movement Detection: A State-of-Art Survey", *Translational Engineering in Health and Medicine, IEEE Journal of*, v. 1, pp. 2100212–2100212, 2013.

[14] HARISCHANDRA, J., PERERA, M. "Intelligent emotion recognition system using brain signals (EEG)". In: *IEEE EMBS Conference on Biomedical Engineering and Sciences (IECBES)*, pp. 454–459, Dec 2012.

[15] FU, X., GUAN, X., PELI, E., et al. "Automatic Calibration Method for Driver's Head Orientation in Natural Driving Environment", *IEEE Transactions on Intelligent Transportation Systems*, v. 14, n. 1, pp. 303–312, March 2013.

[16] HORAK, K. "Fatigue features based on eye tracking for driver inattention system". In: *Telecommunications and Signal Processing (TSP), 2011 34th International Conference on*, pp. 593–597, Aug 2011.

[17] ZHANG, C., YIN, Z., FLORENCIO, D. "Improving depth perception with motion parallax and its application in teleconferencing". In: *IEEE International Workshop on Multimedia Signal Processing (MMSP)*, pp. 1–6, 2009.

[18] HANSEN, D., JI, Q. "In the Eye of the Beholder: A Survey of Models for Eyes and Gaze", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 32, n. 3, pp. 478–500, March 2010.

[19] DUCHOWSKI, A. T. *Eye Tracking Methodology: Theory and Practice.* Springer, jul 2007.

[20] YOO, D. H., KIM, J. H., LEE, B. R., et al. "Non-contact eye gaze tracking system by mapping of corneal reflections". In: *IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 94–99, May 2002.

[21] YOO, D. H., CHUNG, M. J. "A novel non-intrusive eye gaze estimation using cross-ratio under large head motion." *Computer Vision and Image Understanding*, v. 98, n. 1, pp. 25–51, 2005.

[22] COETZER, R. C., HANCKE, G. P. "Eye detection for a real-time vehicle driver fatigue monitoring system". In: *IEEE Intelligent Vehicles Symposium*, pp. 66–71, June 2011.

[23] VALENTI, R., GEVERS, T. "Accurate Eye Center Location through Invariant Isocentric Patterns", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 34, n. 9, pp. 1785–1798, Sept 2012.

[24] DU, C., WU, Q., YANG, J., et al. "SVM based ASM for facial landmarks location". In: *Proceedings of the International Conference on Computer and Information Technology*, pp. 321–326, july 2008.

[25] COOTES, T. F., TAYLOR, C. J., COOPER, D. H., et al. "Active Shape Models-Their Training and Application", *Computer Vision and Image Understanding*, v. 61, n. 1, pp. 38–59, jan 1995.

[26] COOTES, T. F., EDWARDS, G. J., TAYLOR, C. J. "Active appearance models", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 23, n. 6, pp. 681–685, jun 2001.

[27] VAPNIK, V. N. *The Nature of Statistical Learning Theory.* 1 ed. New York, NY, USA, Springer-Verlag New York, Inc., 1995.

[28] ZHOU, M., LIANG, L., SUN, J., et al. "AAM based face tracking with temporal matching and face segmentation". In: *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pp. 701–708, 2010.

[29] CRISTINACCE, D., COOTES, T. F. "Feature Detection and Tracking with Constrained Local Models". In: *Proceedings of the British Machine Vision Conference (BMVC)*, pp. 95.1–95.10, 2006.

[30] VUKADINOVIC, D., PANTIC, M. "Fully automatic facial feature point detection using Gabor feature based boosted classifiers". In: *Proceedings of the International Conference on Systems, Man and Cybernetics*, pp. 1692–1698, october 2005.

[31] TIMM, F., BARTH, E. "Accurate eye centre localisation by means of gradients". In: *Proceedings of the International Conference on Computer Theory and Applications*, pp. 125–130, Algarve, Portugal, 2011.

[32] VALENTI, R., GEVERS, T. "Accurate eye center location and tracking using isophote curvature". In: *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pp. 1–8, june 2008.

[33] LOWE, D. G. "Distinctive Image Features from Scale-Invariant Keypoints", *International Journal of Computer Vision*, v. 60, n. 2, pp. 91–110, 2004.

[34] ASADIFARD, M., SHANBEZADEH, J. "Automatic adaptive center of pupil detection using face detection and cdf analysis". In: *Proceedings of the International MultiConference of Engineers and Computer Scientists (IMECS)*, pp. 1–5, 2010.

[35] ZHOU, Z.-H., GENG, X. "Projection functions for eye detection", *Pattern Recognition*, v. 37, n. 5, pp. 1049–1056, may 2004.

[36] S.ASTERIADIS, N.NIKOLAIDIS, A.HAJDU, et al. "An Eye Detection Algorithm Using Pixel to Edge Information". In: *European Signal Processing Conference (EUSIPCO)*, pp. 1–5, Florence, Italy, 2006.

[37] SIVIC, J., EVERINGHAM, M., ZISSERMAN, A. "Who Are You? - Learning Person Specific Classifiers from Video". In: *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1145–1152, Florida, US, jun 2009.

[38] BELHUMEUR, P., JACOBS, D., KRIEGMAN, D., et al. "Localizing parts of faces using a consensus of exemplars". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 545–552, 2011.

[39] FISCHLER, M. A., BOLLES, R. C. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography", *Commun. ACM*, v. 24, n. 6, pp. 381–395, jun 1981.

[40] OKUMA, K., TALEGHANI, A., DE FREITAS, N., et al. "A Boosted Particle Filter: Multitarget Detection and Tracking". In: *European Conference on Computer Vision (ECCV)*, pp. 28–39, 2004.

[41] GRABNER, H., BISCHOF, H. "On-line Boosting and Vision". In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, v. 1, pp. 260–267, June 2006.

[42] LEIBE, B., SCHINDLER, K., VAN GOOL, L. "Coupled Detection and Trajectory Estimation for Multi-Object Tracking". In: *IEEE 11th International Conference on Computer Vision (ICCV)*, pp. 1–8, Oct 2007.

[43] AVIDAN, S. "Ensemble Tracking", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 29, n. 2, pp. 261–271, Feb 2007.

[44] WU, B., NEVATIA, R. "Detection and Tracking of Multiple, Partially Occluded Humans by Bayesian Combination of Edgelet Based Part Detectors", *International Journal of Computer Vision*, v. 75, n. 2, pp. 247–266, nov 2007.

[45] ANDRILUKA, M., ROTH, S., SCHIELE, B. "People-tracking-by-detection and people-detection-by-tracking". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–8, June 2008.

[46] BREITENSTEIN, M., REICHLIN, F., LEIBE, B., et al. "Robust tracking-by-detection using a detector confidence particle filter". In: *IEEE International Conference on Computer Vision (CVPR)*, pp. 1515–1522, Sept 2009.

[47] KALAL, Z., MATAS, J., MIKOLAJCZYK, K. "Online learning of robust object detectors during unstable tracking". In: *IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pp. 1417–1424, Sept 2009.

[48] LUCAS, B. D., KANADE, T. "An Iterative Image Registration Technique with an Application to Stereo Vision". In: *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 674–679, 1981.

[49] DOUCET, A., DE FREITAS, N., GORDON, N. *Sequential Monte Carlo Methods in Practice*. Information Science and Statistics. Springer, 2001.

[50] BERCLAZ, J., FLEURET, F., FUA, P. "Robust People Tracking with Global Trajectory Optimization". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, v. 1, pp. 744–750, June 2006.

[51] PERERA, A., SRINIVAS, C., HOOGS, A., et al. "Multi-Object Tracking Through Simultaneous Long Occlusions and Split-Merge Conditions". In:

IEEE Conference on Computer Vision and Pattern Recognition (CVPR), v. 1, pp. 666–673, June 2006.

[52] CAI, Y., DE FREITAS, N., LITTLE, J. J. "Robust Visual Tracking for Multiple Targets". In: *Proceedings of the 9th European Conference on Computer Vision - Volume Part IV*, European Conference on Computer Vision (ECCV), pp. 107–118, Berlin, Heidelberg, 2006. Springer-Verlag.

[53] HUANG, C., WU, B., NEVATIA, R. "Robust Object Tracking by Hierarchical Association of Detection Responses". In: *European Conference on Computer Vision (ECCV)*, pp. 788–801, Berlin, Heidelberg, 2008.

[54] VIOLA, P. A., JONES, M. J. "Robust Real-Time Face Detection", *International Journal of Computer Vision*, v. 57, n. 2, pp. 137–154, 2004.

[55] SHI, J., TOMASI, C. "Good Features to Track". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 593 – 600, 1994.

[56] YUN, T., GUAN, L. "Automatic landmark point detection and tracking for human facial expressions", *EURASIP J. Image and Video Processing*, v. 2013, pp. 8, 2013.

[57] EVERINGHAM, M., ZISSERMAN, A. "Regression and classification approaches to eye localization in face images". In: *International Conference on Automatic Face and Gesture Recognition*, pp. 441–446, April 2006.

[58] DORINI, L., GOLDENSTEIN, S. "Unscented Feature Tracking", *Computer Vision and Image Understanding*, v. 115, n. 1, pp. 8–11, 2011.

[59] MORAIS, E., FERREIRA, A., CUNHA, S., et al. "A Multiple Camera Methodology for Automatic Localization and Tracking of Futsal Players", *Pattern Recognition Letters*, v. 39, pp. 21–30, 2014.

[60] HASTIE, T., TIBSHIRANI, R., FRIEDMAN, J. H. *The elements of statistical learning: data mining, inference, and prediction.* New York: Springer-Verlag, 2009.

[61] JOACHIMS, T. "Making large-Scale SVM Learning Practical". In: Schölkopf, B., Burges, C., Smola, A. (Eds.), *Advances in Kernel Methods - Support Vector Learning*, MIT Press, cap. 11, pp. 169–184, Cambridge, MA, 1999.

[62] DUDA, R. O., HART, P. E., STORK, D. G. *Pattern Classification.* Wiley-Interscience Publication, 2000.

[63] BISHOP, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. 1 ed. Secaucus, NJ, USA, Springer-Verlag New York, Inc., 2006.

[64] THEODORIDIS, S., KOUTROUMBAS, K. *Pattern Recognition*. 4 ed. San Diego, California, USA, Academic Press, 2009.

[65] VAN DER HEIJDEN, F., DUIN, R., DE RIDDER, D., et al. *Classification, Parameter Estimation and State Estimation: An Engineering Approach Using MATLAB*. Wiley, 2004.

[66] MAGGIO, E., CAVALLARO, A. *Video Tracking: Theory and Practice*. Wiley, 2011.

[67] GOLDENSTEIN, S. "A Gentle Introduction to Predictive Filters", *Revista de Informatica Terica e Aplicada*, v. XI, n. 1, pp. 61–89, 2004.

[68] ISARD, M., BLAKE, A. "CONDENSATION - conditional density propagation for visual tracking", *International Journal of Computer Vision*, v. 29, pp. 5–28, 1998.

[69] CHEN, Z. *Bayesian Filtering - From Kalman Filters to Particle Filters, and Beyond*. Relatório técnico, McMaster University, 2003.

[70] KUMAR, B. V. K. V., MAHALANOBIS, A., JUDAY, R. D. *Correlation Pattern Recognition*. 1 ed. New York, NY, USA, Cambridge University Press, 2005.

[71] XIE, C., SAVVIDES, M., KUMAR, B. V. K. V. "Redundant Class-Dependence Feature Analysis Based on Correlation Filters Using FRGC2.0 Data". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) - Workshops*, pp. 153–153, june 2005.

[72] MAHALANOBIS, A., KUMAR, B. V. K. V., CASASENT, D. "Minimum Average Correlation Energy Filters", *Applied Optics*, v. 26, n. 17, pp. 3633–3640, 1987.

[73] LAI, H., RAMANATHAN, V., WECHSLER, H. "Reliable Face Recognition Using Adaptive and Robust Correlation Filters", *Computer Vision and Image Understanding*, v. 111, n. 3, pp. 329–350, 2008.

[74] MOYA, M., HUSH, D. "Network Constraints and Multi-Objective Optimization for One-Class Classification", *Neural Networks*, v. 9, n. 3, pp. 463–474, 1996.

[75] SCHLKOPF, B., SMOLA, A. *Learning with Kernels Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, 2002.

[76] LIU, B. *Web Data Mining.* Springer, 2007.

[77] JAIN, A. K. *Fundamentals of Digital Image Processing.* 1 ed. Upper Saddle River, NJ, USA, Prentice-Hall, Inc., 1989.

[78] BIOID TECHNOLOGY RESEARCH. "The BioID Face Database". http://www.bioid.com, 2011.

[79] PHILLIPS, P., MOON, H., RIZVI, S., et al. "The FERET evaluation methodology for face-recognition algorithms", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 22, n. 10, pp. 1090–1104, Oct 2000.

[80] EYE TRACKING RESEARCH. "Robust Eye Tracking in High Definition Videoconference Sequences". http://www.lps.ufrj.br/∼biometria/etr/, 2012.

[81] TAN, X., TRIGGS, B. "Enhanced Local Texture Feature Sets for Face Recognition under Difficult Lighting Conditions". In: *IEEE International Workshop on Analysis and Modeling of Faces and Gestures (AMFG)*, pp. 168–182, 2007.

[82] TAN, X., TRIGGS, B. "Enhanced Local Texture Feature Sets for Face Recognition Under Difficult Lighting Conditions", *IEEE Transactions on Image Processing*, v. 19, n. 6, pp. 1635–1650, June 2010.

[83] JESORSKY, O., KIRCHBERG, K. J., FRISCHHOLZ, R. "Robust Face Detection Using the Hausdorff Distance". In: *International Conference on Audio and Video-Based Biometric Person Authentication*, pp. 90–95, 2001.

[84] BEHNKE, S. "Learning Face Localization Using Hierarchical Recurrent Networks". In: *Proceedings of the International Conference on Artificial Neural Networks*, pp. 1319–1324, 2002.

[85] CRISTINACCE, D., COOTES, T., SCOTT, I. "A Multi-Stage Approach to Facial Feature Detection". In: *Proceedings of the British Machine Vision Conference (BMVC)*, pp. 30.1–30.10, 2004.

[86] HAMOUZ, M., KITTLER, J., KAMARAINEN, J., et al. "Feature-based affine-invariant localization of faces", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 27, n. 9, pp. 1490 –1495, sept. 2005.

[87] BAI, L., SHEN, L., WANG, Y. "A Novel Eye Location Algorithm based on Radial Symmetry Transform". In: *ICPR*, 2006.

[88] CAMPADELLI, P., LANZAROTTI, R., LIPORI, G. "Precise Eye Localization through a General-to-specific Model Definition". In: *Proceedings of the British Machine Vision Conference (BMVC)*, pp. 187–196, 2006.

[89] CHEN, D., TANG, X., OU, Z., et al. "A Hierarchical FloatBoost and MLP Classifier for Mobile Phone Embedded Eye Location System". In: *Advances in Neural Networks (ISNN)*, pp. 20–25, 2006.

[90] NIU, Z., SHAN, S., YAN, S., et al. "2D Cascaded AdaBoost for Eye Localization". In: *International Conference on Pattern Recognition (ICPR)*, pp. 511–514, 2006.

[91] TÜRKAN, M., PARDÀS, M., ÇETIN, A. E. "Human eye localization using edge projections." In: *International Conference on Computer Vision Theory and Applications (VISAPP)*, pp. 410–415.

[92] KROON, B., HANJALIC, A., MAAS, S. M. "Eye localization for face matching: is it always useful and under what conditions?" In: *International Conference on Content-based Image and Video Retrieval (CIVR)*, pp. 379–388, 2008.

[93] REN, Y., WANG, S., HOU, B., et al. "A Novel Eye Localization Method With Rotation Invariance", *IEEE Transactions on Image Processing*, v. 23, n. 1, pp. 226–239, Jan 2014.

[94] MARTINEZ, B., VALSTAR, M., BINEFA, X., et al. "Local Evidence Aggregation for Regression-Based Facial Point Detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 35, n. 5, pp. 1149–1163, 2013.

[95] ENGAUGE DIGITIZER. "Engauge Digitizer - Digitizing software". http://digitizer.sourceforge.net/, 2012.

[96] VALSTAR, M., MARTINEZ, B., BINEFA, X., et al. "Facial point detection using boosted regression and graph models". In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 2729–2736, 2010.

[97] MILBORROW, S., NICOLLS, F. "Locating Facial Features with an Extended Active Shape Model". In: Forsyth, D., Torr, P., Zisserman, A. (Eds.), *Eu-*

ropean Conference on Computer Vision (ECCV), v. 5305, *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 504–513, 2008.

[98] RAPP, V., SENECHAL, T., BAILLY, K., et al. "Multiple kernel learning SVM and statistical validation for facial landmark detection". In: *IEEE International Conference on Automatic Face Gesture Recognition and Workshops*, pp. 265–271, 2011.

[99] CRISTINACCE, D., COOTES, T. "Automatic feature localisation with constrained local models", *Pattern Recognition*, v. 41, n. 10, pp. 3054 – 3067, 2008.

[100] BELHUMEUR, P., JACOBS, D., KRIEGMAN, D., et al. "Localizing Parts of Faces Using a Consensus of Exemplars", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 35, n. 12, pp. 2930–2940, 2013.

[101] EFRATY, B., HUANG, C., SHAH, S., et al. "Facial landmark detection in uncontrolled conditions". In: *Biometrics (IJCB), 2011 International Joint Conference on*, pp. 1–8, 2011.

[102] DIBEKLIOGLU, H., SALAH, A., GEVERS, T. "A Statistical Method for 2-D Facial Landmarking", *IEEE Transactions on Image Processing*, v. 21, n. 2, pp. 844–858, 2012.

[103] GIBBONS, J. *Nonparametric Statistics: An Introduction*. SAGE Publications, 1992.

[104] ARAUJO, G. M., DA SILVA JÚNIOR, W. S., DA SILVA, E. A. B., et al. "Detecção de Landmarks Faciais Usando SVM". In: *XXIX Sompósio Brasileiro de Telecomunicações (SBRT)*, pp. 1–5, 2011.

[105] HASTIE, T., TIBSHIRANI, R., FRIEDMAN, J. "Additive logistic regression: a statistical view of boosting", *Annals of Statistics*, v. 28, pp. 1–5, 2000.

[106] HARTLEY, R. I., ZISSERMAN, A. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[107] DALAL, N., TRIGGS, B. "Histograms of Oriented Gradients for Human Detection". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 886–893, 2005.

[108] FELZENSZWALB, P. F., HUTTENLOCHER, D. P. "Pictorial Structures for Object Recognition", *International Journal of Computer Vision*, v. 61, pp. 2005, 2003.

Appendix A

Complementary Results on Landmarks Detection

This Appendix is a complement of the Chapter 3. It contains the complete results on landmarks detection, which consists of the hit rate curves for all landmarks of both datasets, BioID and FERET.

A.1 The effect of the face's and template's sizes on the performance of the IPD

In this section we have the complete results of the experiment employed to obtain the sizes of the template and face with best performance. First of all, Table A.1 show all faces and blocks sizes used in this work. The entries of the tables are the absolute sizes of the block.

Table A.1: Evaluated relative block sizes and faces sizes

		Size of the face (in pixels)				
		25×25	50×50	75×75	100×100	125×125
Block's size (w.r.t. to the face's size)	$\sim 15\%$	3×3	7×7	11×11	15×15	19×19
	$\sim 20\%$	5×5	11×11	15×15	21×21	25×25
	$\sim 28\%$	7×7	15×15	21×21	27×27	35×35
	$\sim 35\%$	9×9	17×17	27×27	35×35	45×45
	$\sim 38\%$	-	19×19	29×19	37×37	47×47

From Figures A.1 to A.10 we show the performance of the IPD when varying the face's and template's sizes.

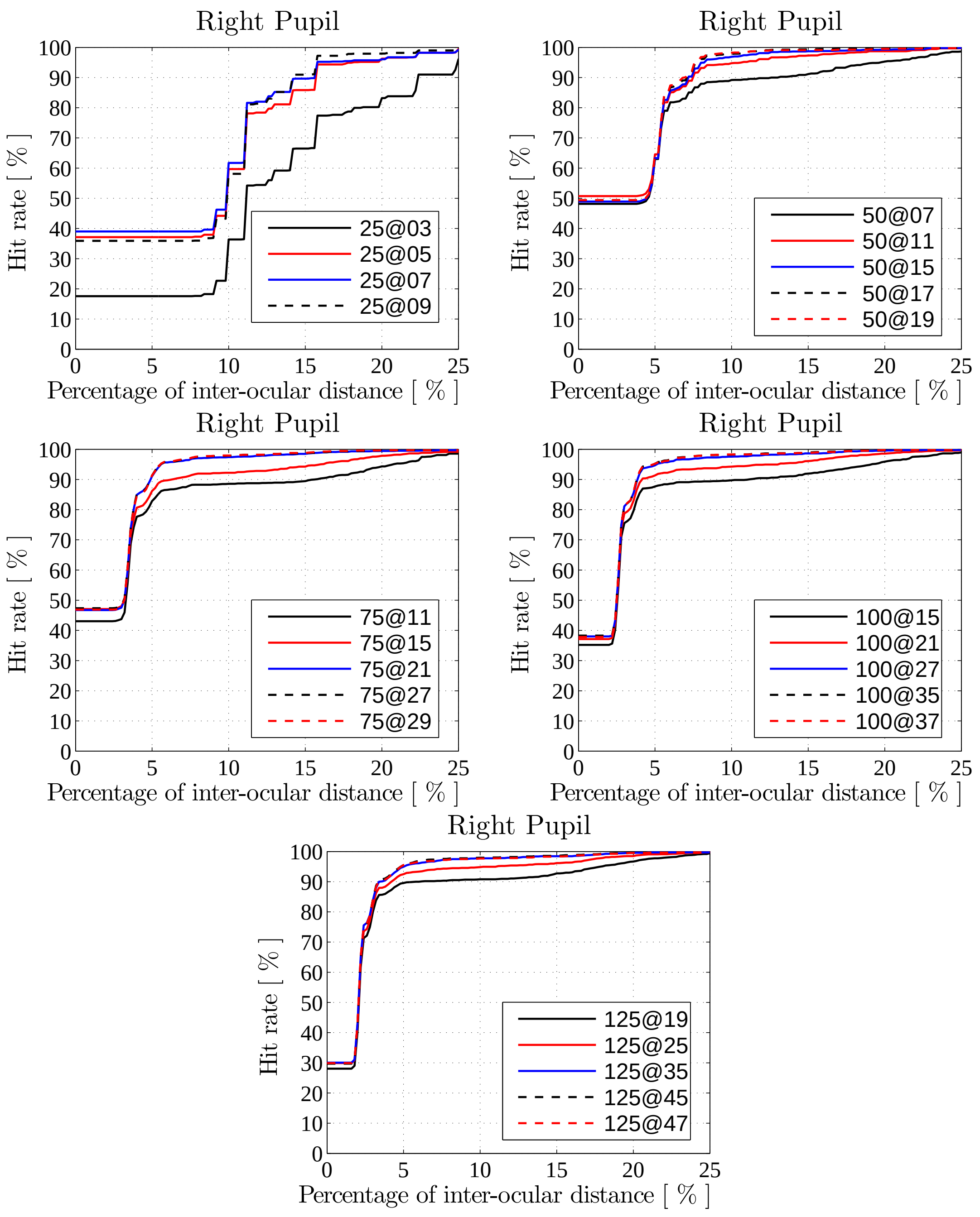


Figure A.1: Average hit rate for the right pupil using faces from 25×25 up to 125×125 pixels and several block sizes.

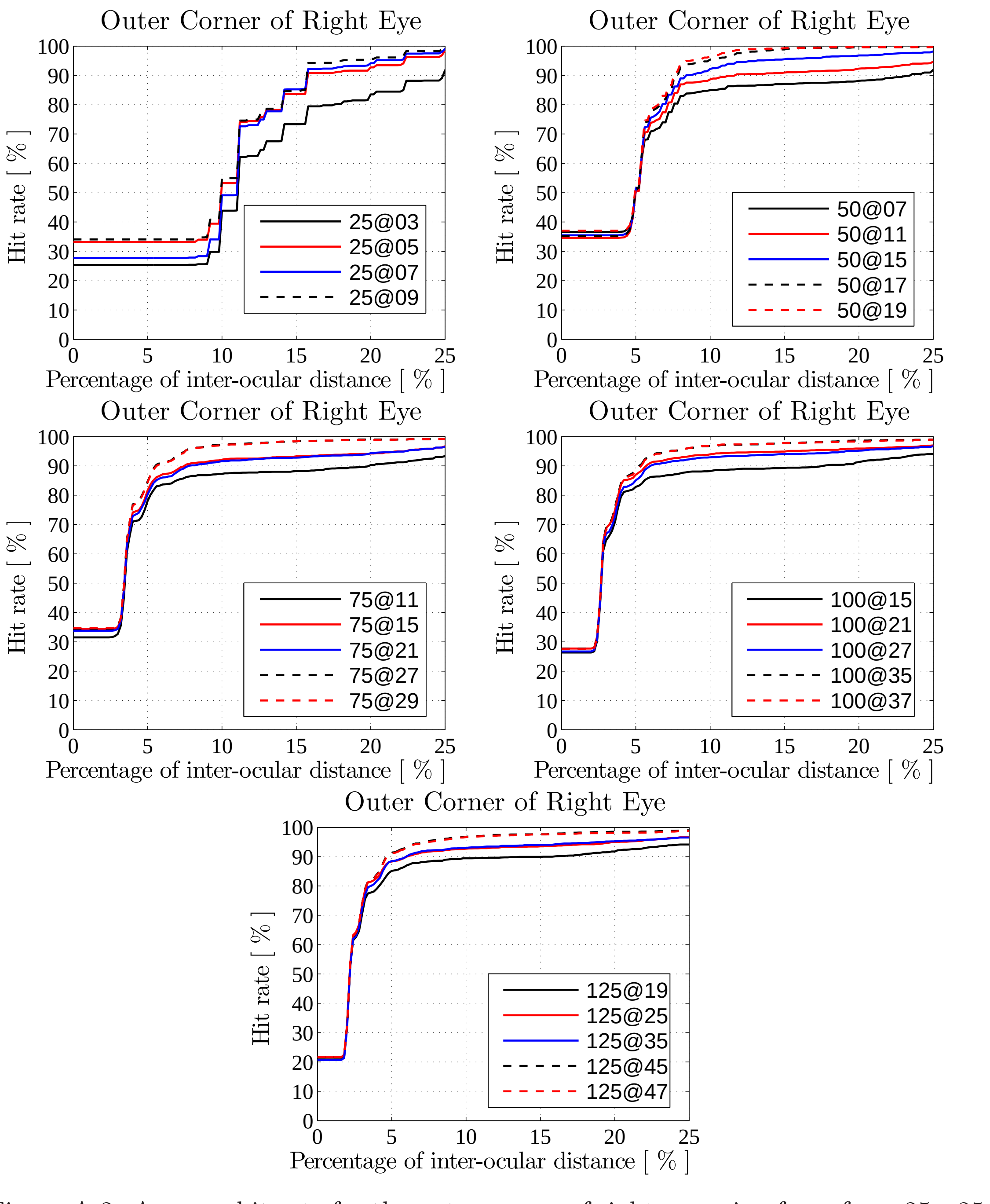


Figure A.2: Average hit rate for the outer corner of right eye using faces from 25×25 up to 125×125 pixels and several block sizes.

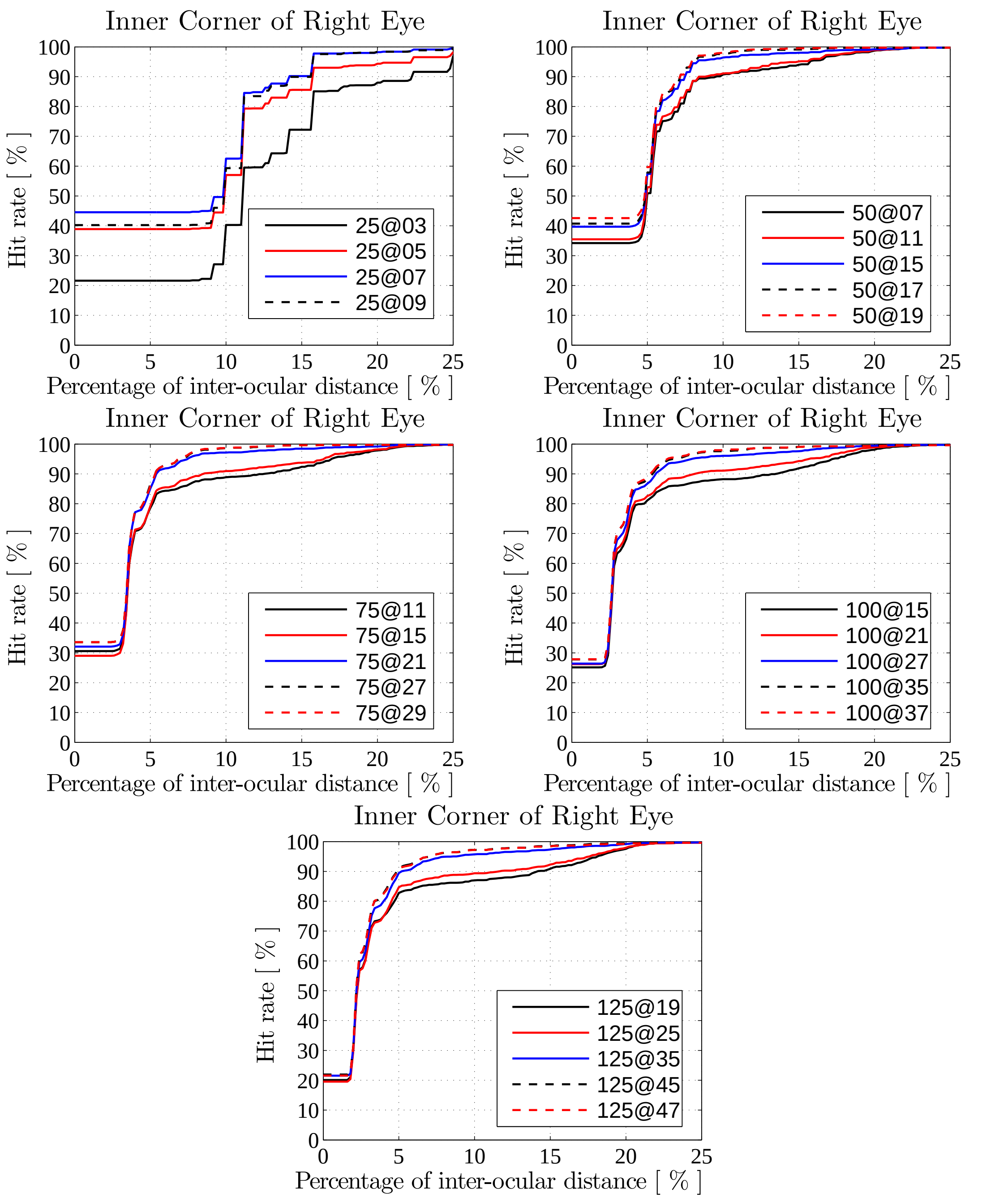


Figure A.3: Average hit rate for the inner corner of right eye using faces from 25×25 up to 125×125 pixels and several block sizes.

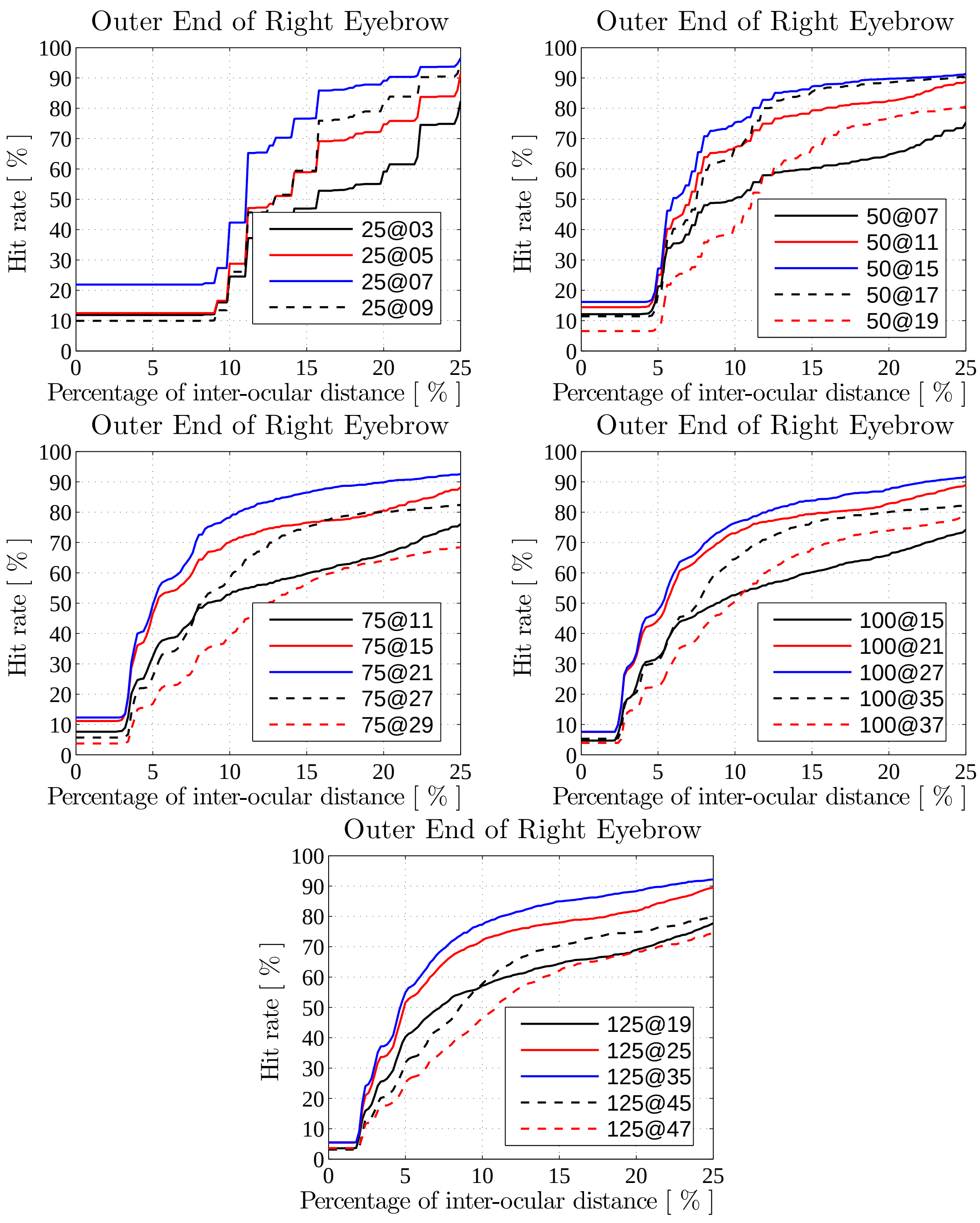


Figure A.4: Average hit rate for the outer end of right eyebrow using faces from 25×25 up to 125×125 pixels and several block sizes.

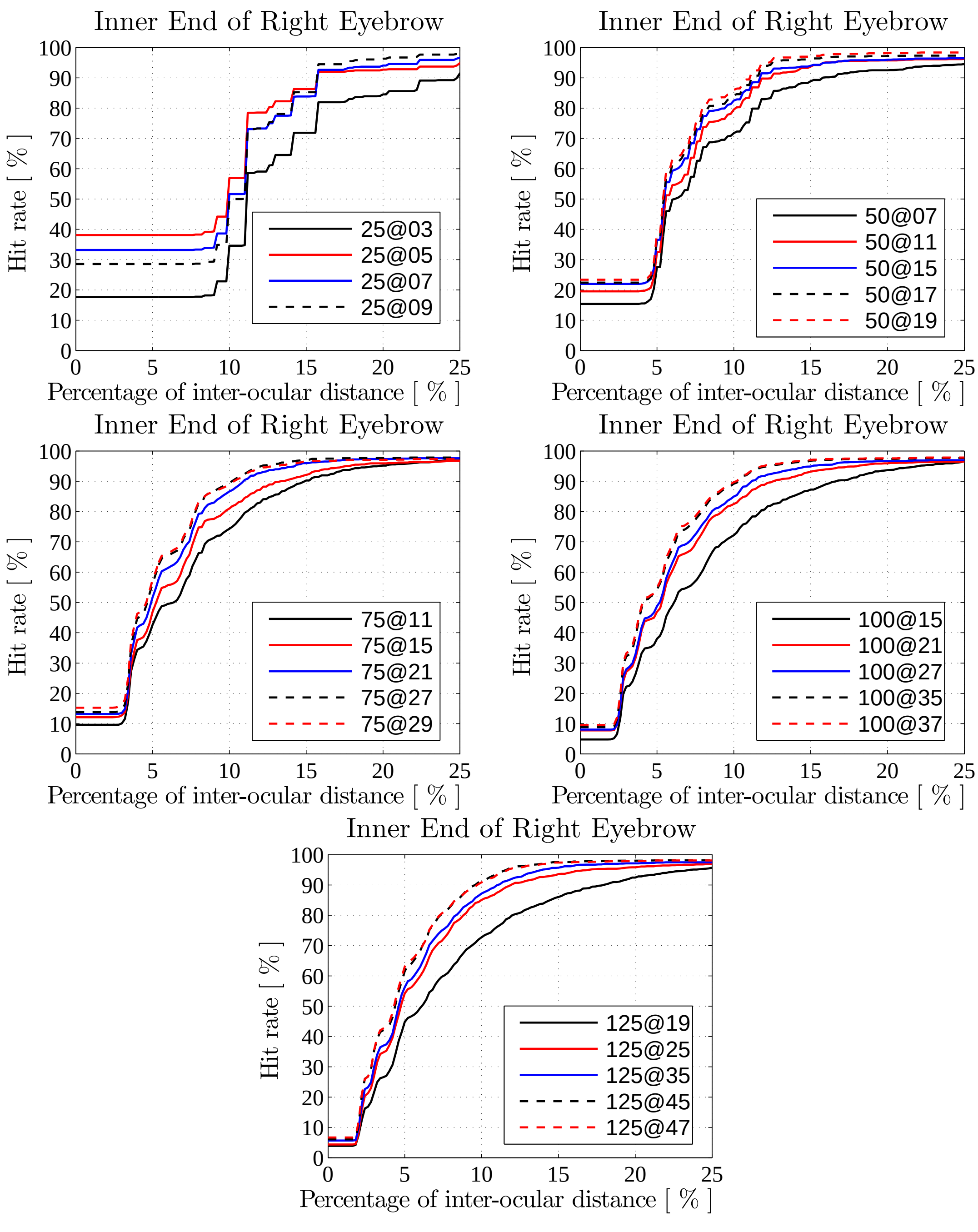


Figure A.5: Average hit rate for the inner end of right eyebrow using faces from 25×25 up to 125×125 pixels and several block sizes.

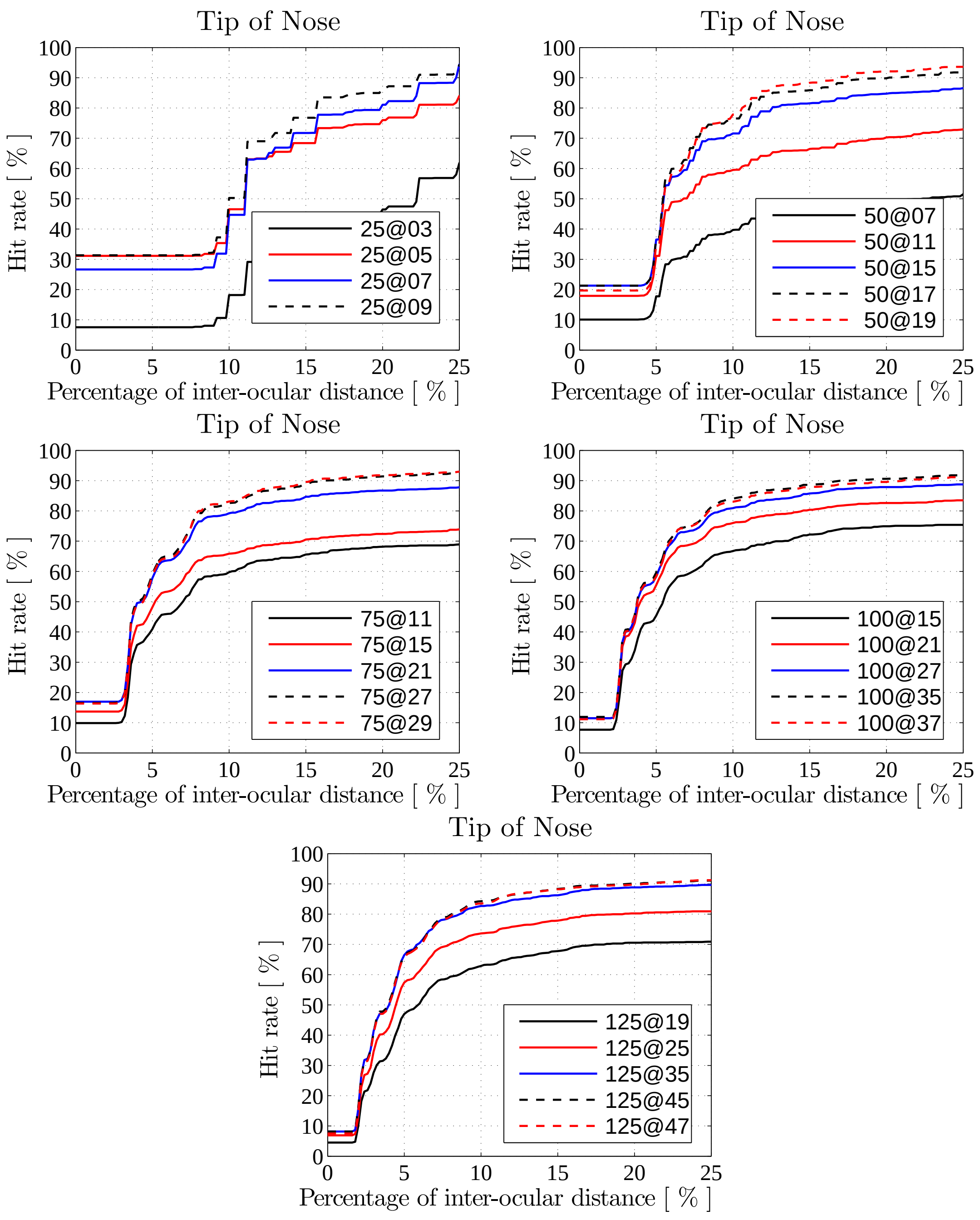


Figure A.6: Average hit rate for the tip of nose using faces from 25×25 up to 125×125 pixels and several block sizes.

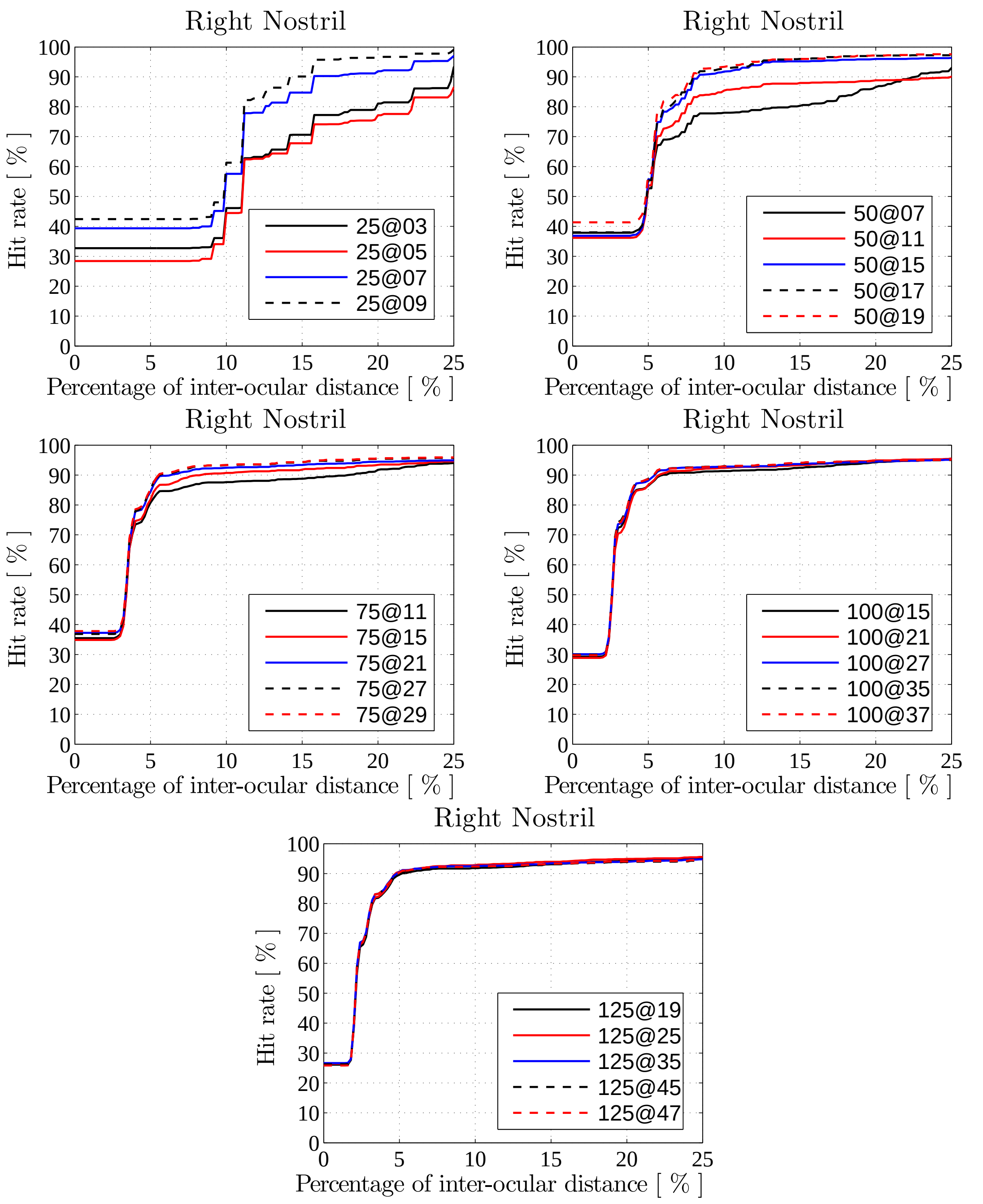


Figure A.7: Average hit rate for the right nostrill using faces from 25 × 25 up to 125 × 125 pixels and several block sizes.

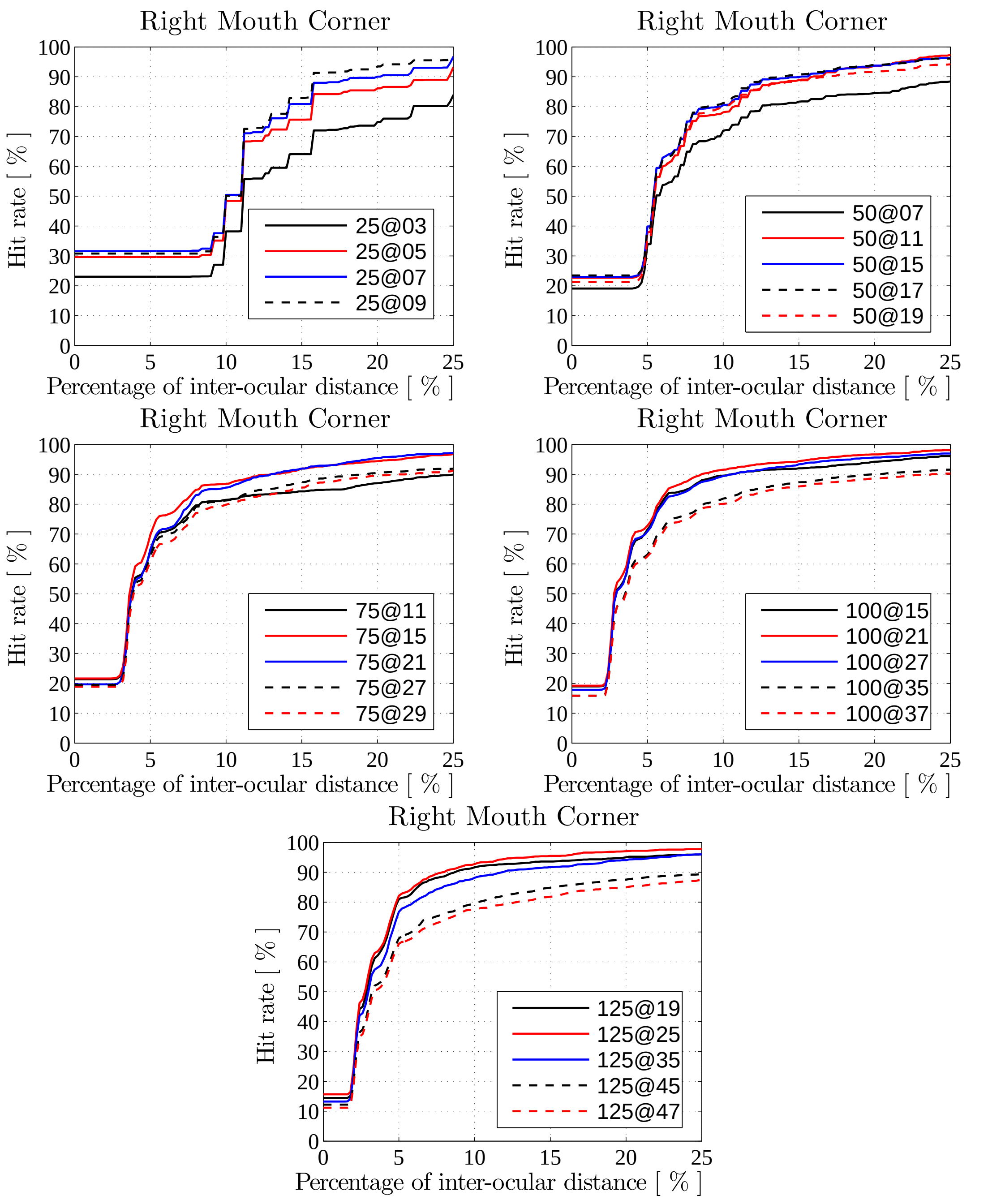


Figure A.8: Average hit rate for the right mouth corner using faces from 25×25 up to 125×125 pixels and several block sizes.

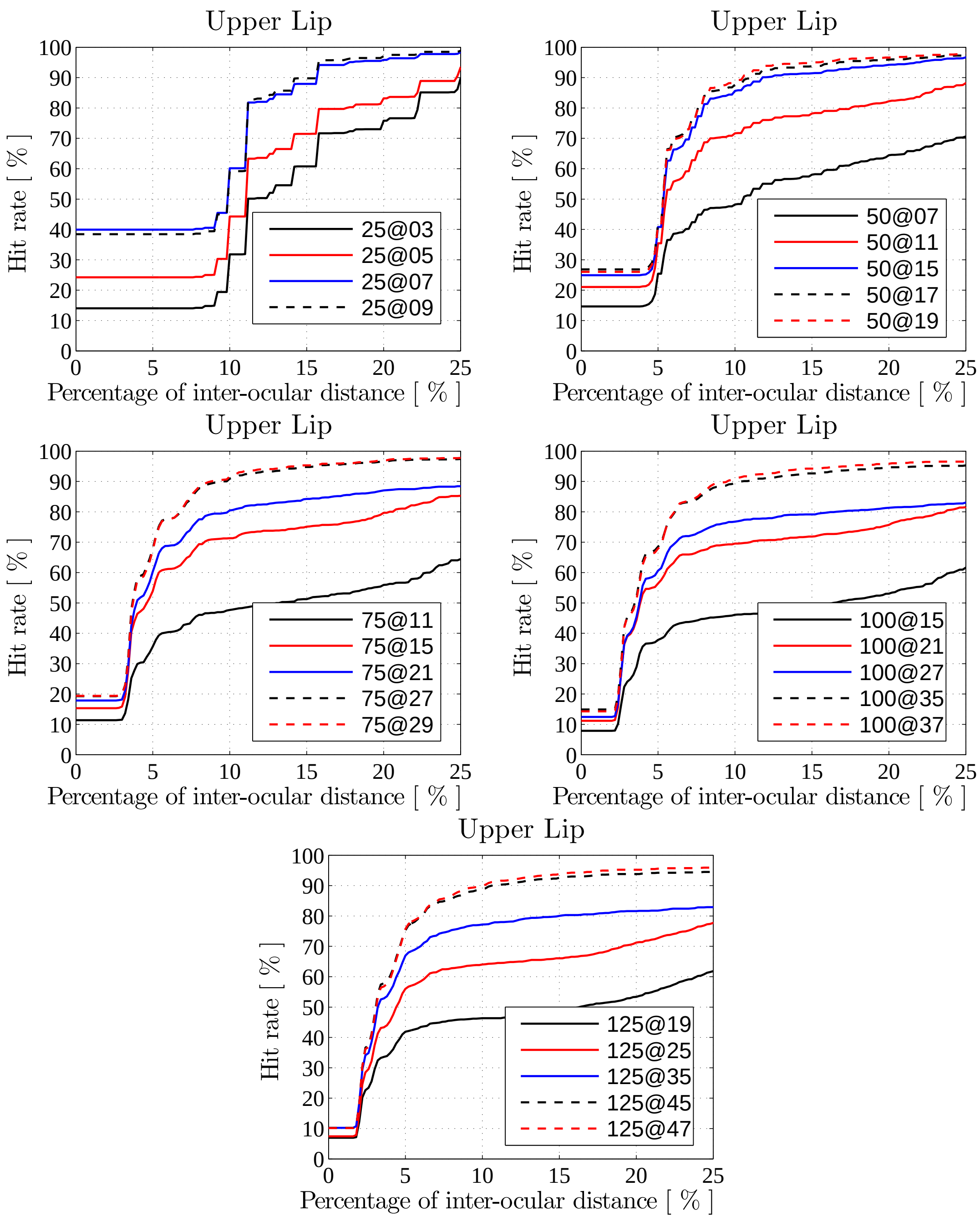


Figure A.9: Average hit rate for the upper lip using faces from 25×25 up to 125×125 pixels and several block sizes.

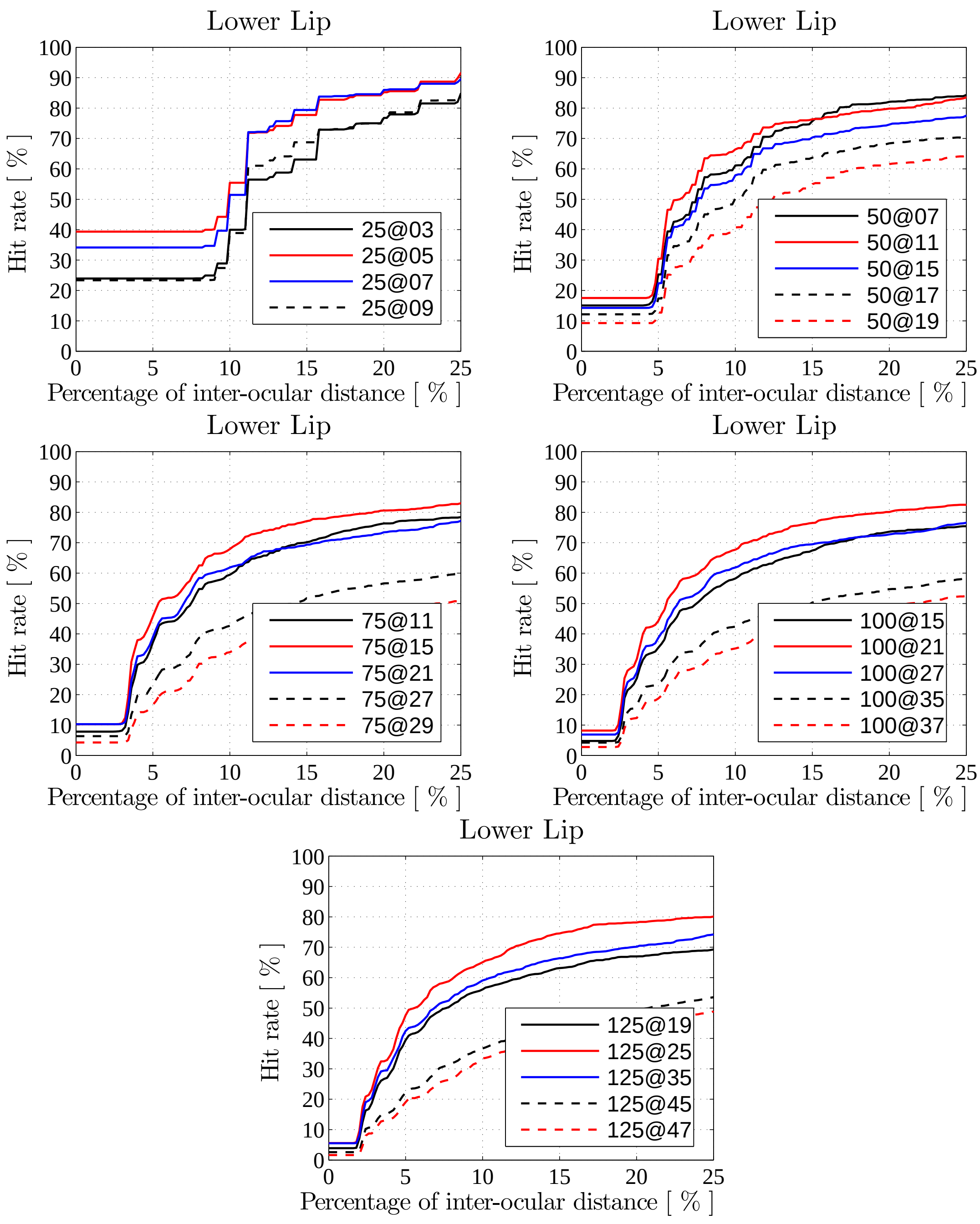


Figure A.10: Average hit rate for the lower lip using faces from 25 × 25 up to 125 × 125 pixels and several block sizes.

In FiguresA.11 to A.30 we have a performance comparison when the relative size of the block is kept fixed. The performance comparison when the relative block's size is approximately 15% of the face's size is shown in Figures A.11 to A.14.

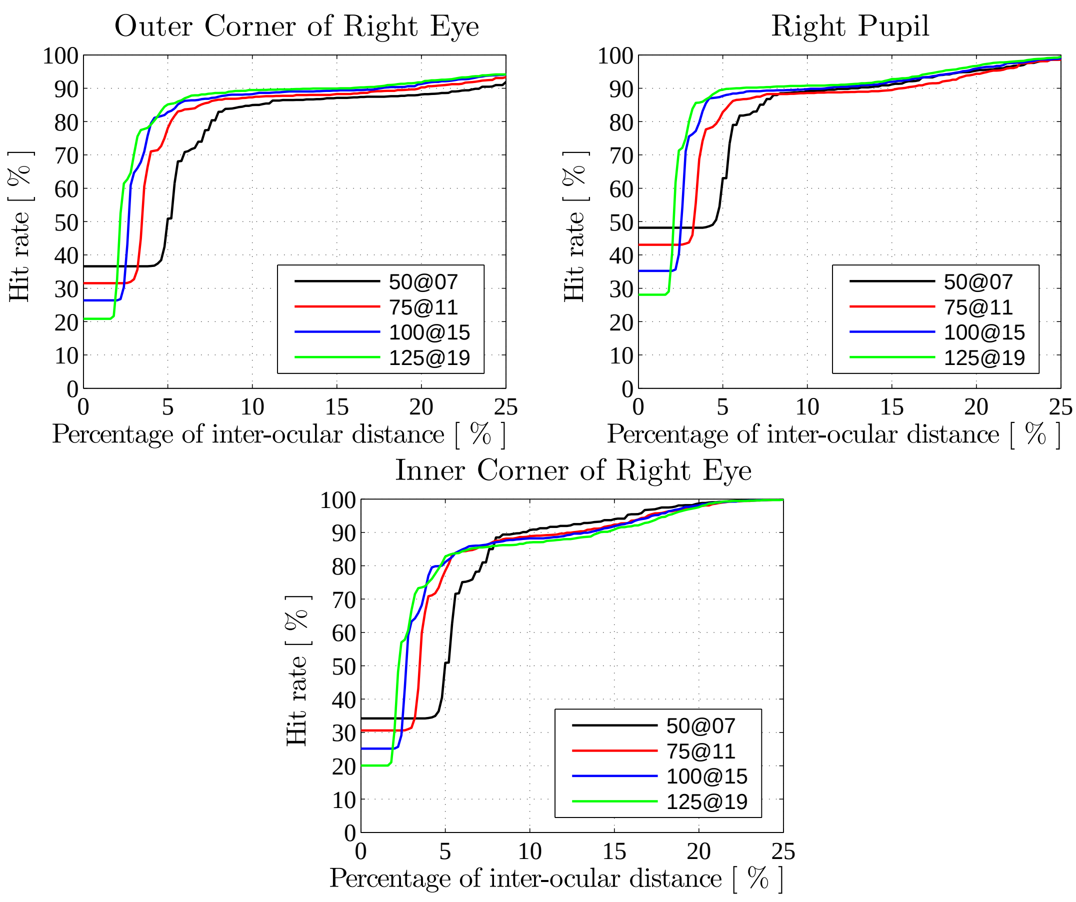


Figure A.11: Average hit rate for the outer corner right eye, right pupil, and inner coner of right eye using faces from 50 × 50 up to 125 × 125 pixels and blocks with approximately 15% of the faces's size

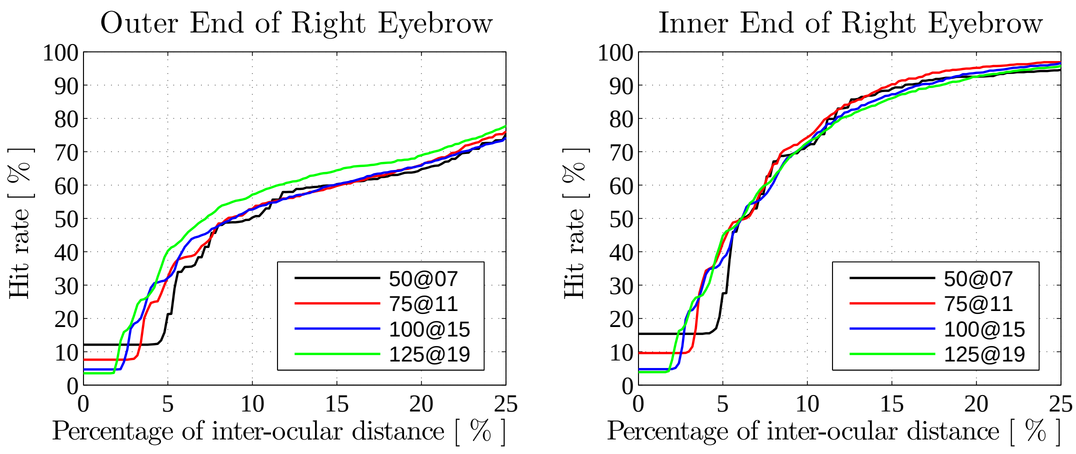
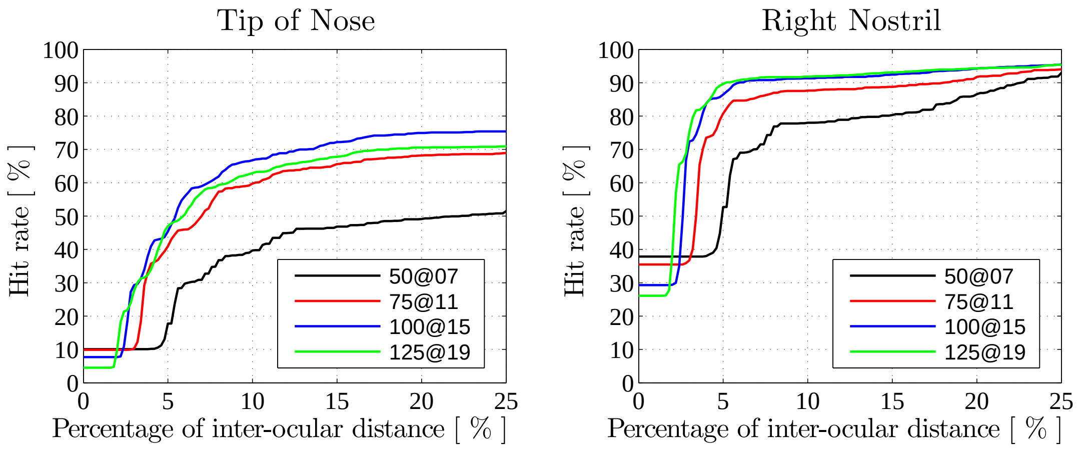


Figure A.12: Average hit rate for the outer end of right eyebrow and outer end of right eyebrow using faces from 50 × 50 up to 125 × 125 pixels and blocks with approximately 15% of the faces's size



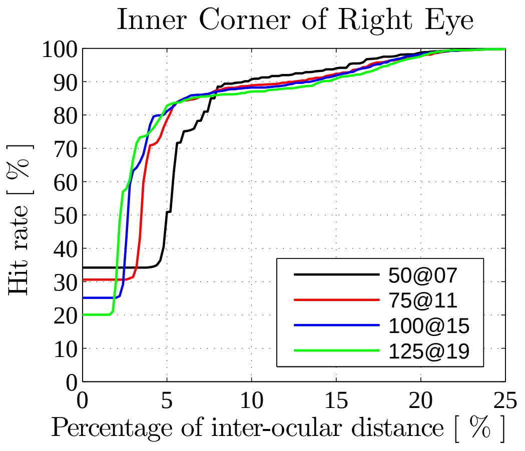


Figure A.13: Average hit rate for the tip of nose and right nostril using faces from 50×50 up to 125×125 pixels and blocks with approximately 15% of the faces's size

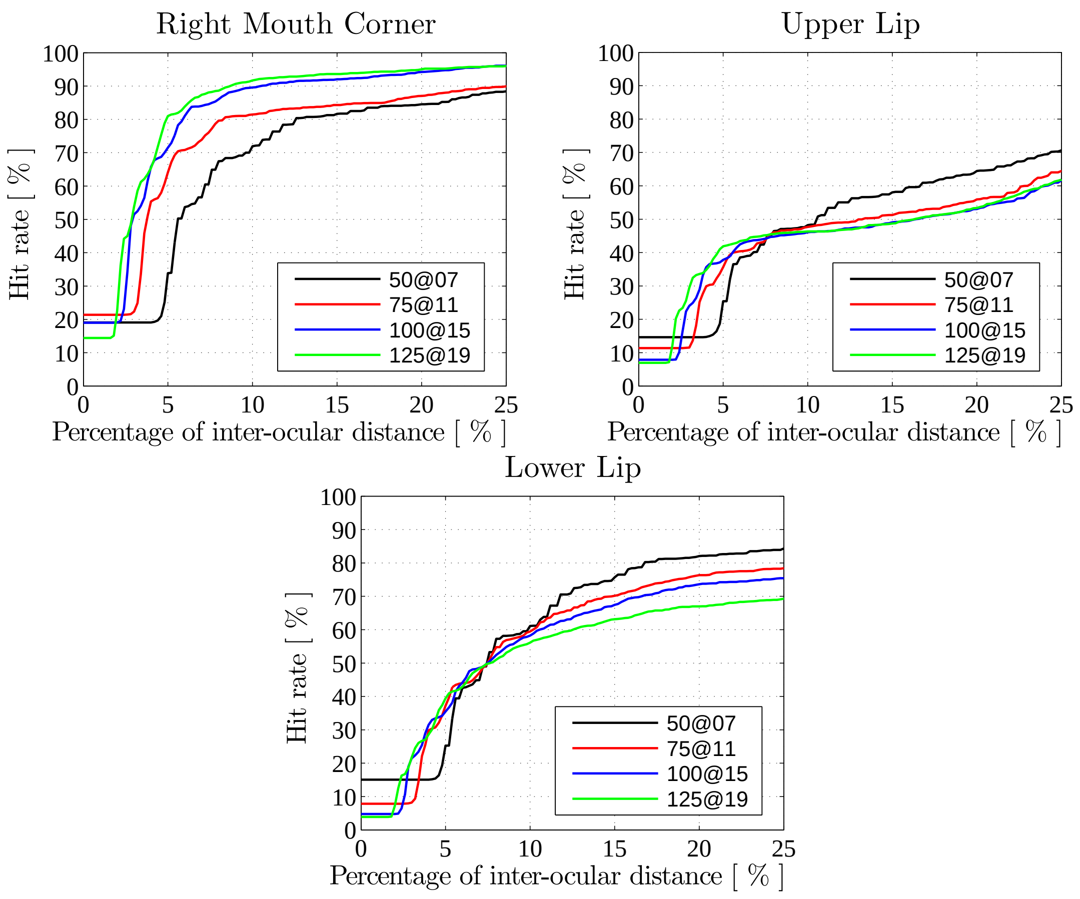
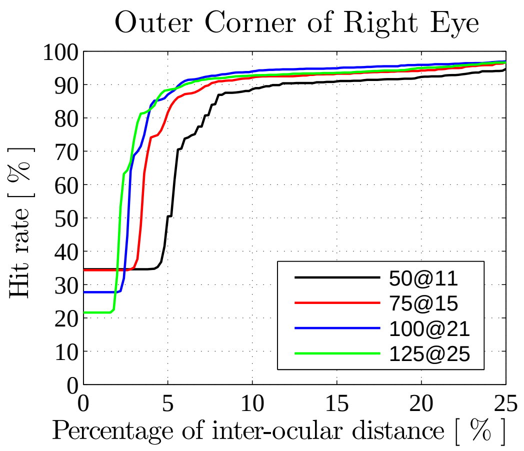
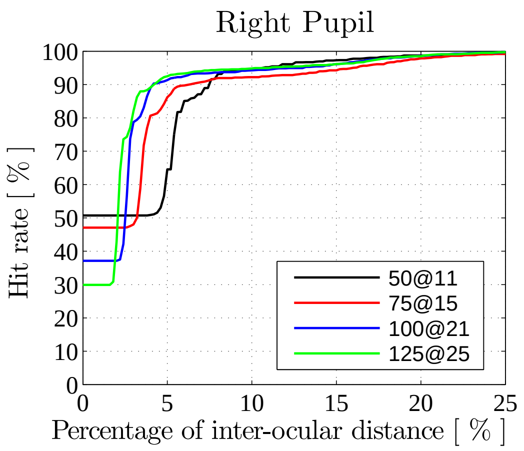


Figure A.14: Average hit rate for right mouth corner, upper lip and lower lip using faces from 50×50 up to 125×125 pixels and blocks with approximately 15% of the faces's size

The performance comparison when the relative block's size is approximately 20% of the face's size is shown in Figures A.15 to A.18.





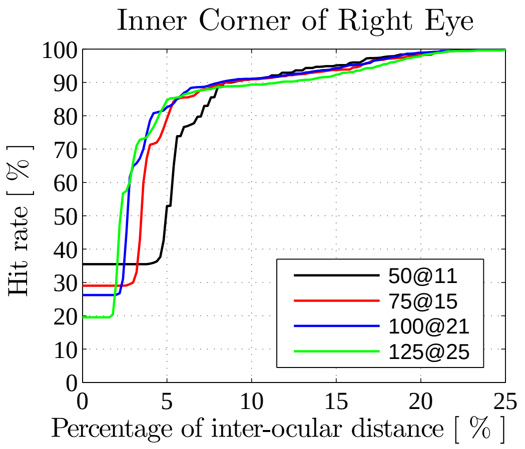
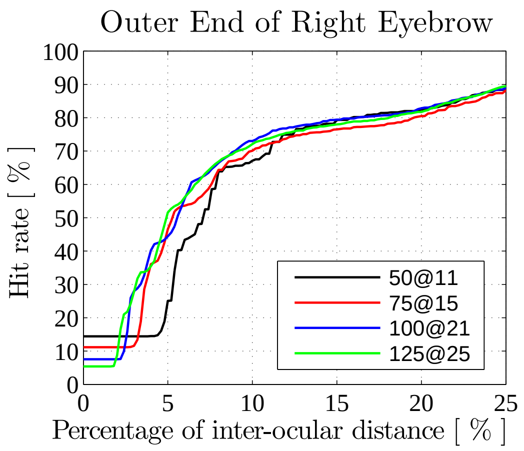


Figure A.15: Average hit rate for the outer corner right eye, right pupil, and inner coner of right eye using faces from 50 × 50 up to 125 × 125 pixels and blocks with approximately 20% of the faces's size



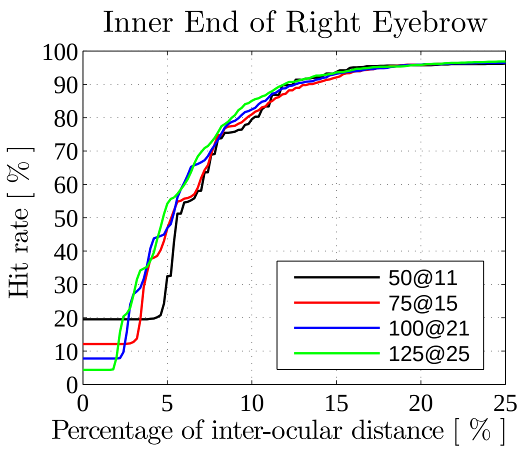


Figure A.16: Average hit rate for the outer end of right eyebrow and outer end of right eyebrow using faces from 50 × 50 up to 125 × 125 pixels and blocks with approximately 20% of the faces's size

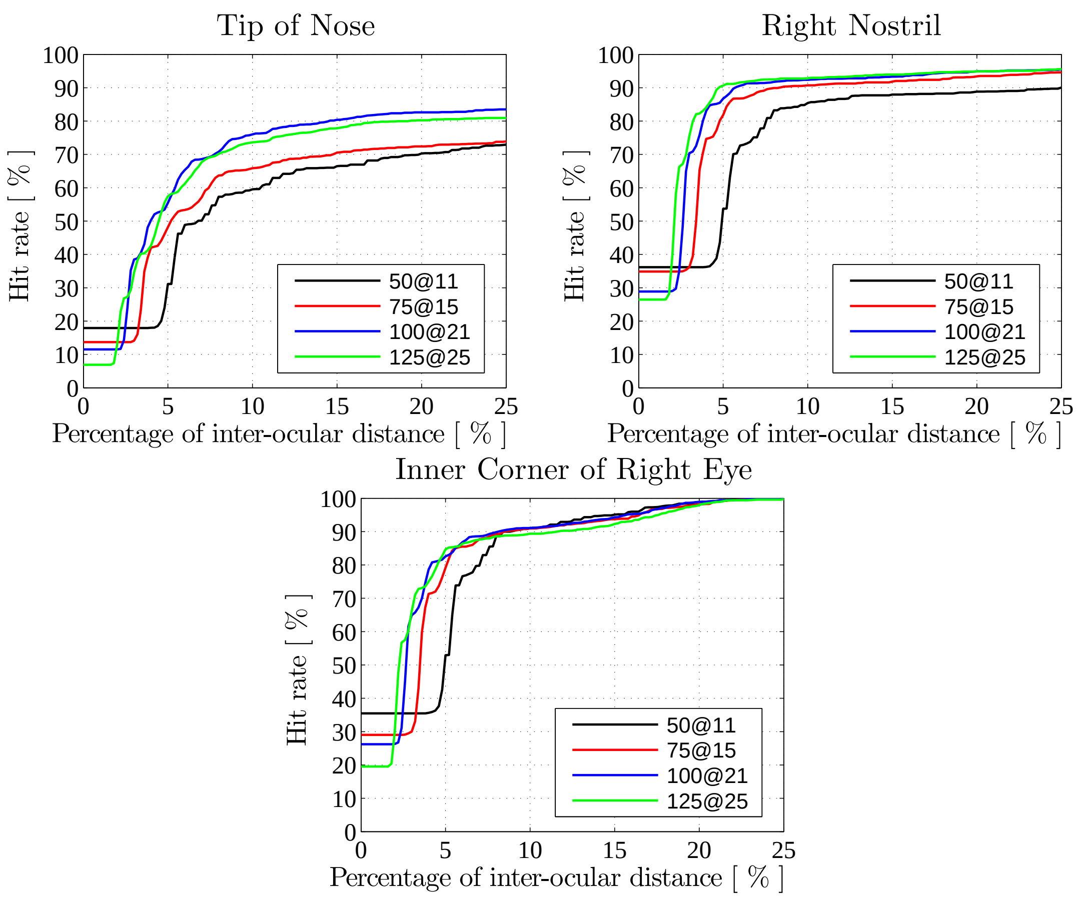


Figure A.17: Average hit rate for the tip of nose and right nostril using faces from 50×50 up to 125×125 pixels and blocks with approximately 20% of the faces's size

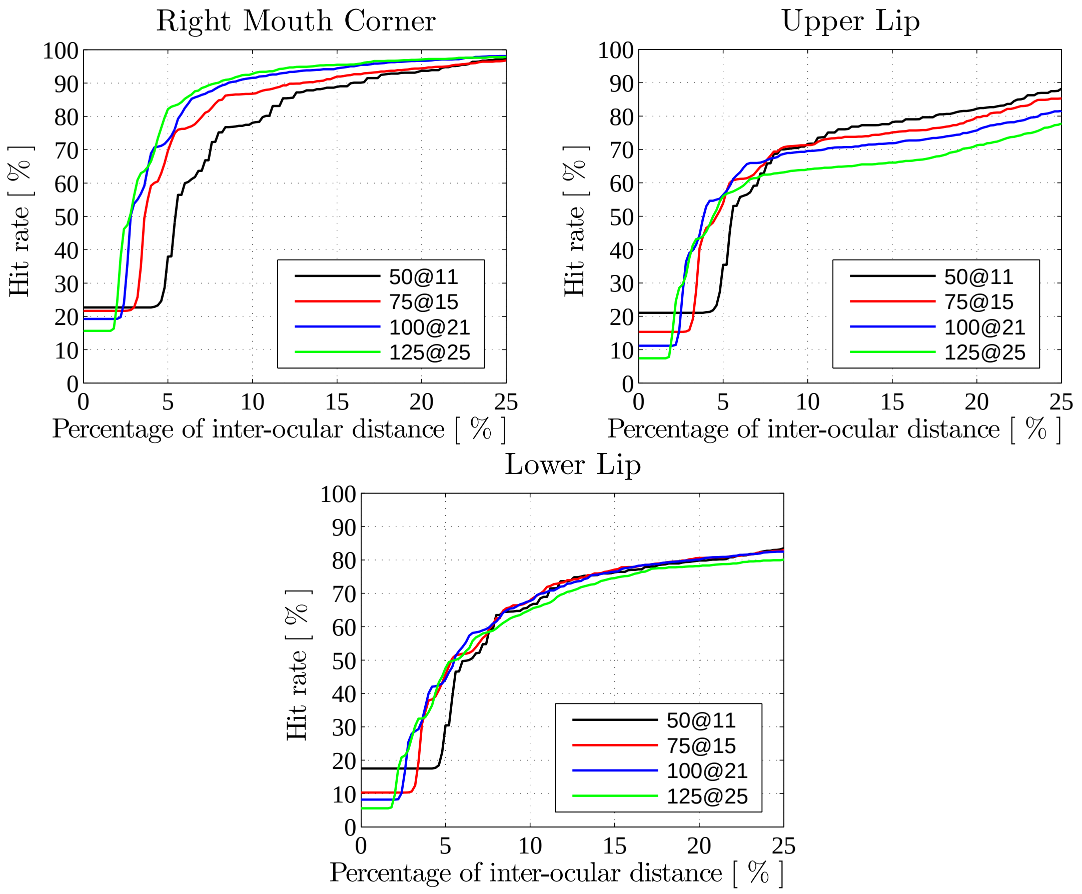


Figure A.18: Average hit rate for right mouth corner, upper lip and lower lip using faces from 50×50 up to 125×125 pixels and blocks with approximately 20% of the faces's size

The performance comparison when the relative block's size is approximately 28% of the face's size is shown in Figures A.19 to A.22.

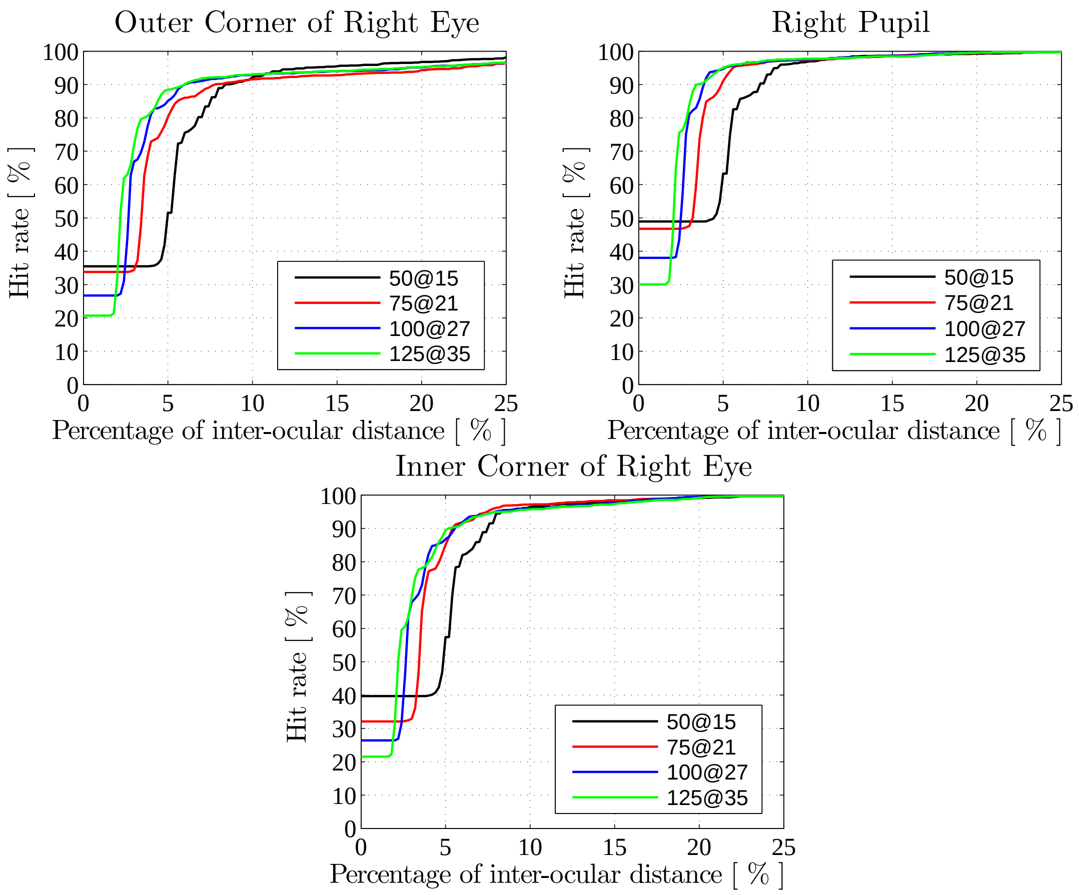


Figure A.19: Average hit rate for the outer corner right eye, right pupil, and inner coner of right eye using faces from 50 × 50 up to 125 × 125 pixels and blocks with approximately 28% of the faces's size

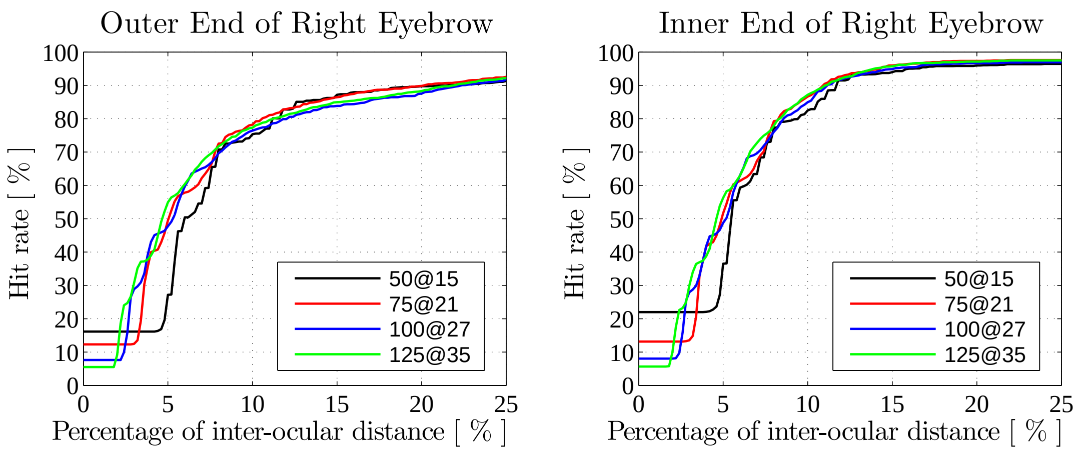


Figure A.20: Average hit rate for the outer end of right eyebrow and outer end of right eyebrow using faces from 50 × 50 up to 125 × 125 pixels and blocks with approximately 28% of the faces's size

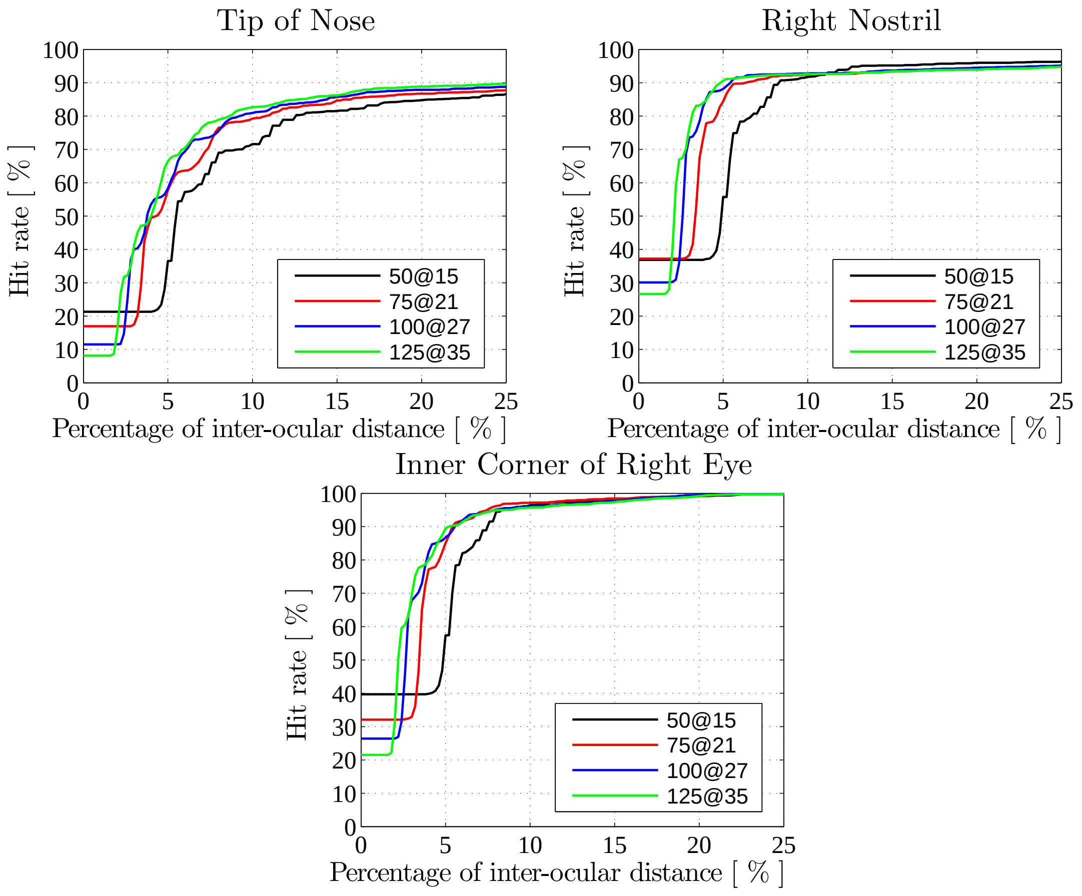


Figure A.21: Average hit rate for the tip of nose and right nostril using faces from 50×50 up to 125×125 pixels and blocks with approximately 28% of the faces's size

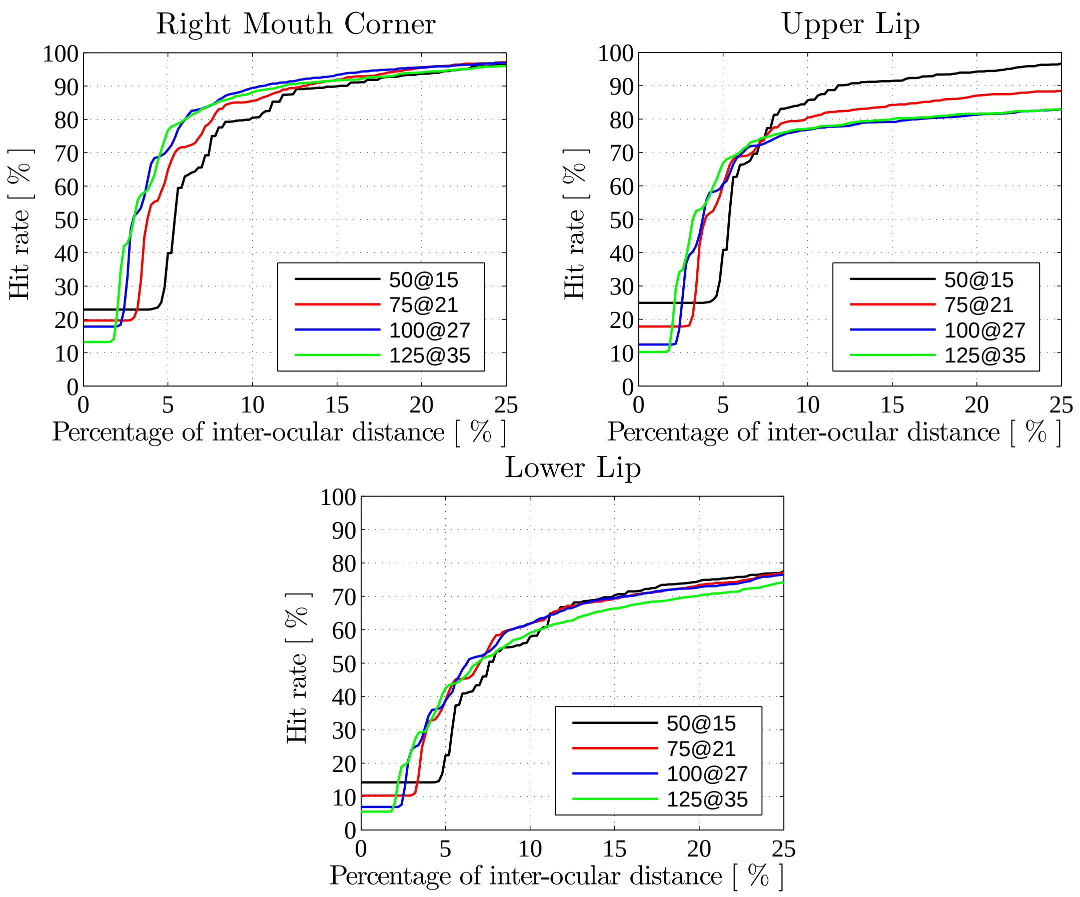


Figure A.22: Average hit rate for right mouth corner, upper lip and lower lip using faces from 50×50 up to 125×125 pixels and blocks with approximately 28% of the faces's size

The performance comparison when the relative block's size is approximately 35% of the face's size is shown in Figures A.24 to A.26.

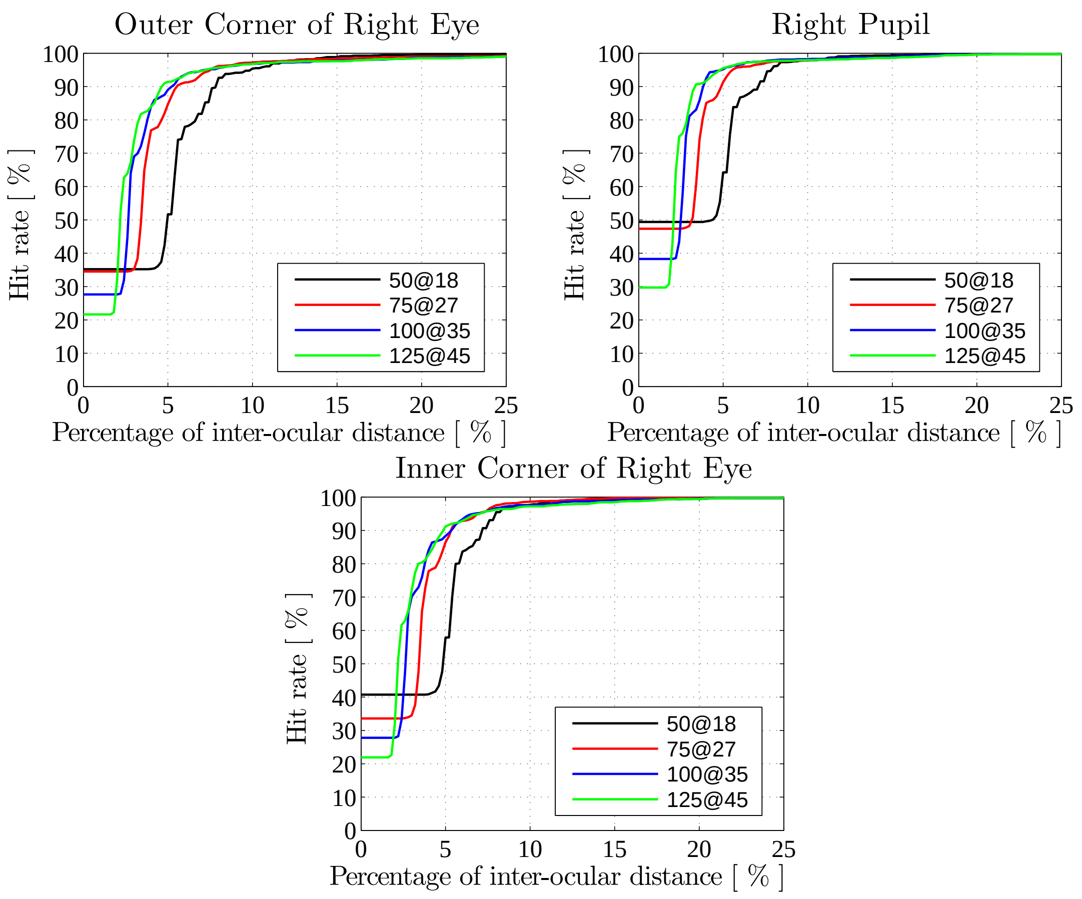


Figure A.23: Average hit rate for the outer corner right eye, right pupil, and inner coner of right eye using faces from 50 × 50 up to 125 × 125 pixels and blocks with approximately 35% of the faces's size

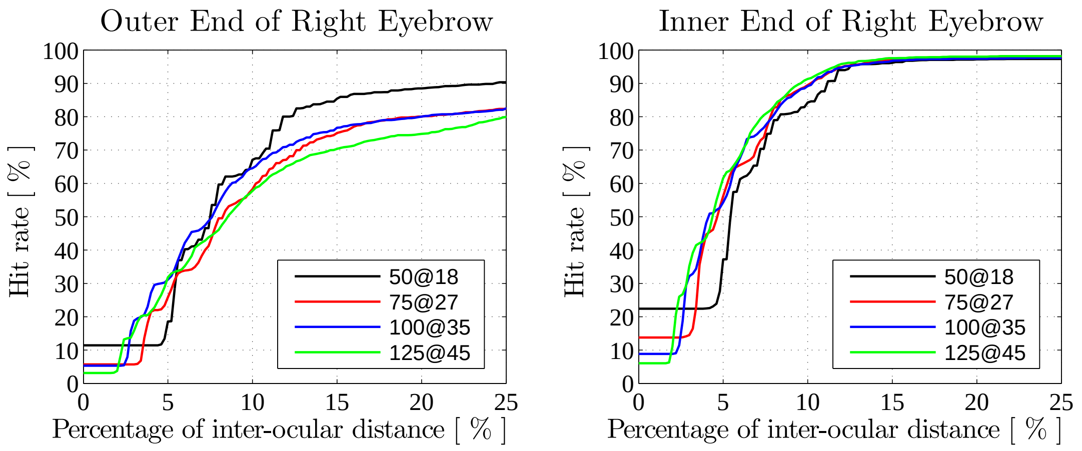


Figure A.24: Average hit rate for the outer end of right eyebrow and outer end of right eyebrow using faces from 50 × 50 up to 125 × 125 pixels and blocks with approximately 35% of the faces's size

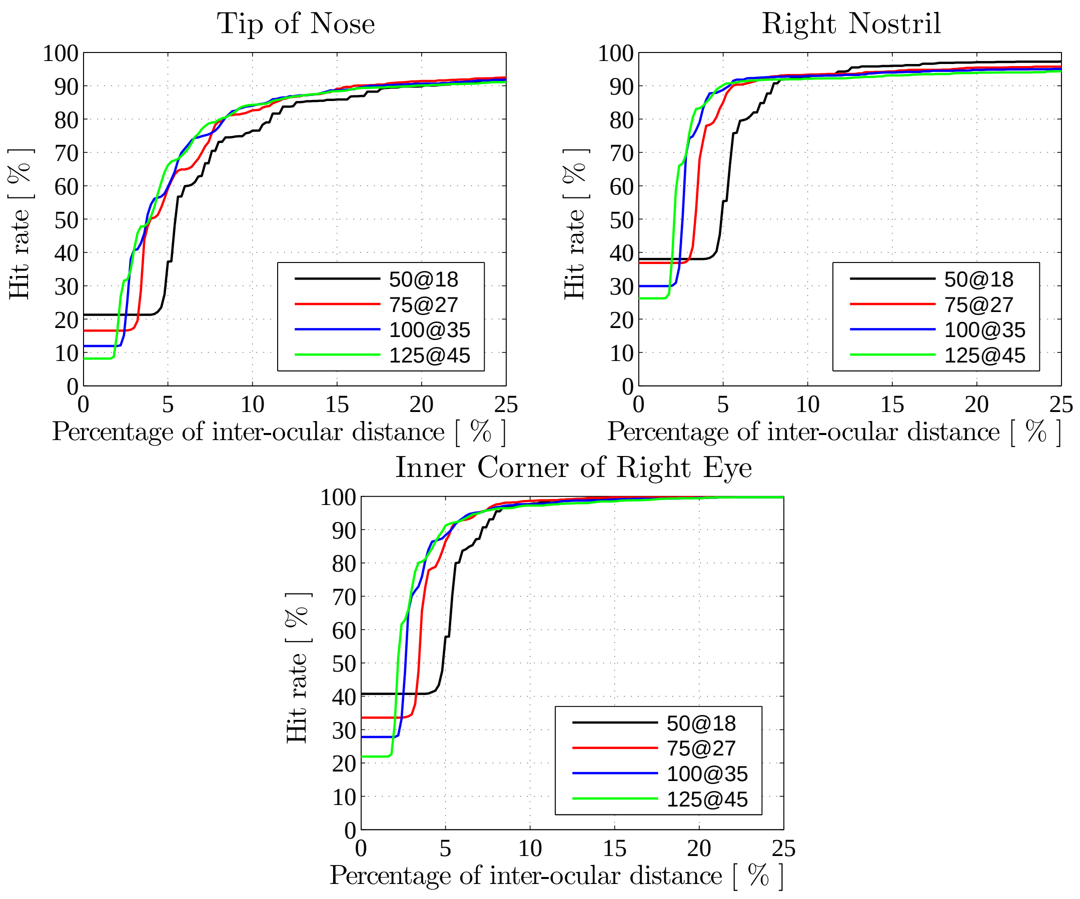


Figure A.25: Average hit rate for the tip of nose and right nostril using faces from 50×50 up to 125×125 pixels and blocks with approximately 35% of the faces's size

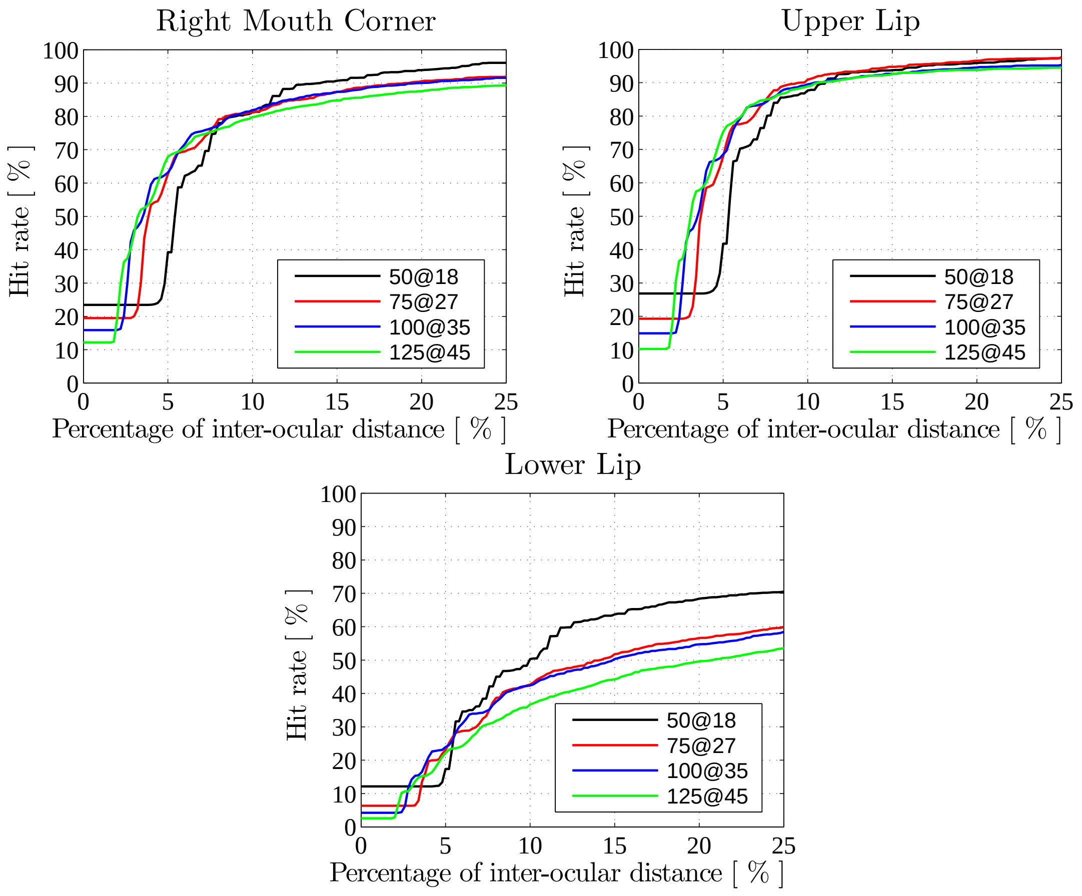
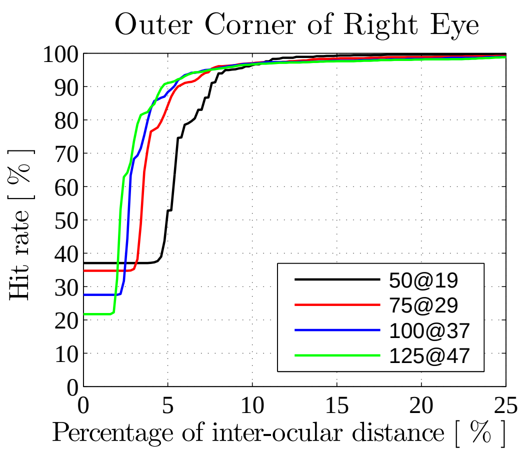
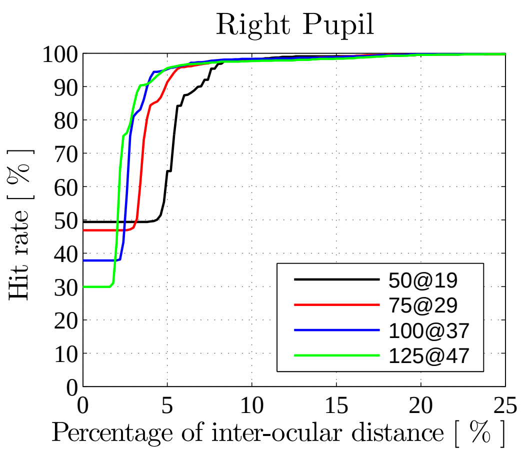


Figure A.26: Average hit rate for right mouth corner, upper lip and lower lip using faces from 50×50 up to 125×125 pixels and blocks with approximately 35% of the faces's size

The performance comparison when the relative block's size is approximately 38% of the face's size is shown in Figures A.27 to A.30.





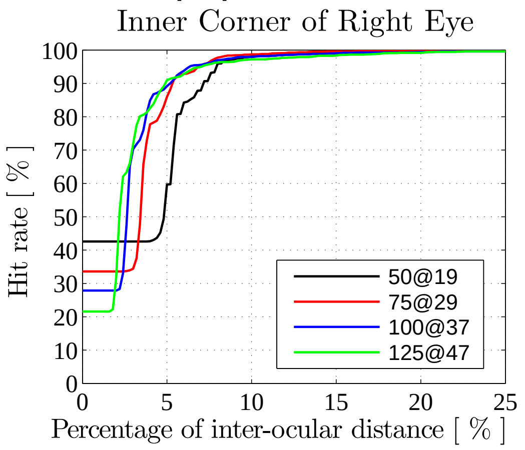
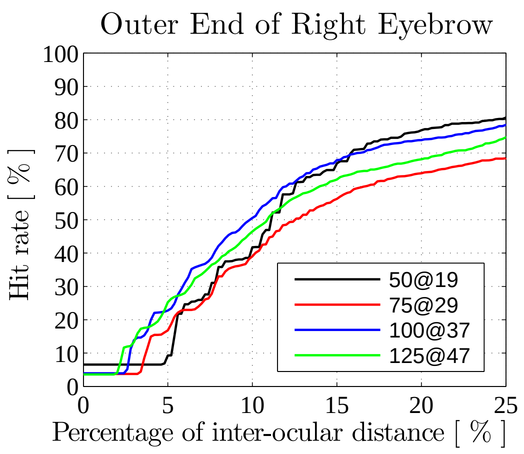


Figure A.27: Average hit rate for the outer corner right eye, right pupil, and inner coner of right eye using faces from 50 × 50 up to 125 × 125 pixels and blocks with approximately 38% of the faces's size



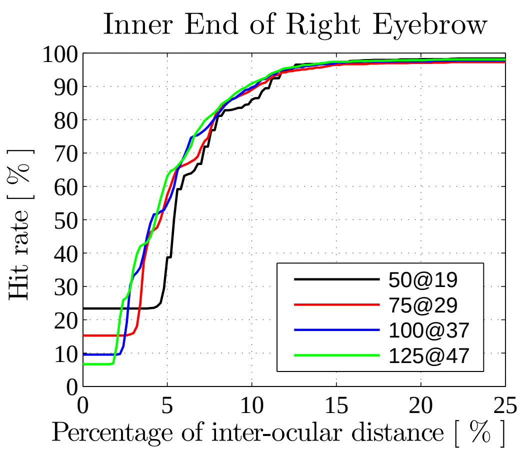


Figure A.28: Average hit rate for the outer end of right eyebrow and outer end of right eyebrow using faces from 50 × 50 up to 125 × 125 pixels and blocks with approximately 38% of the faces's size

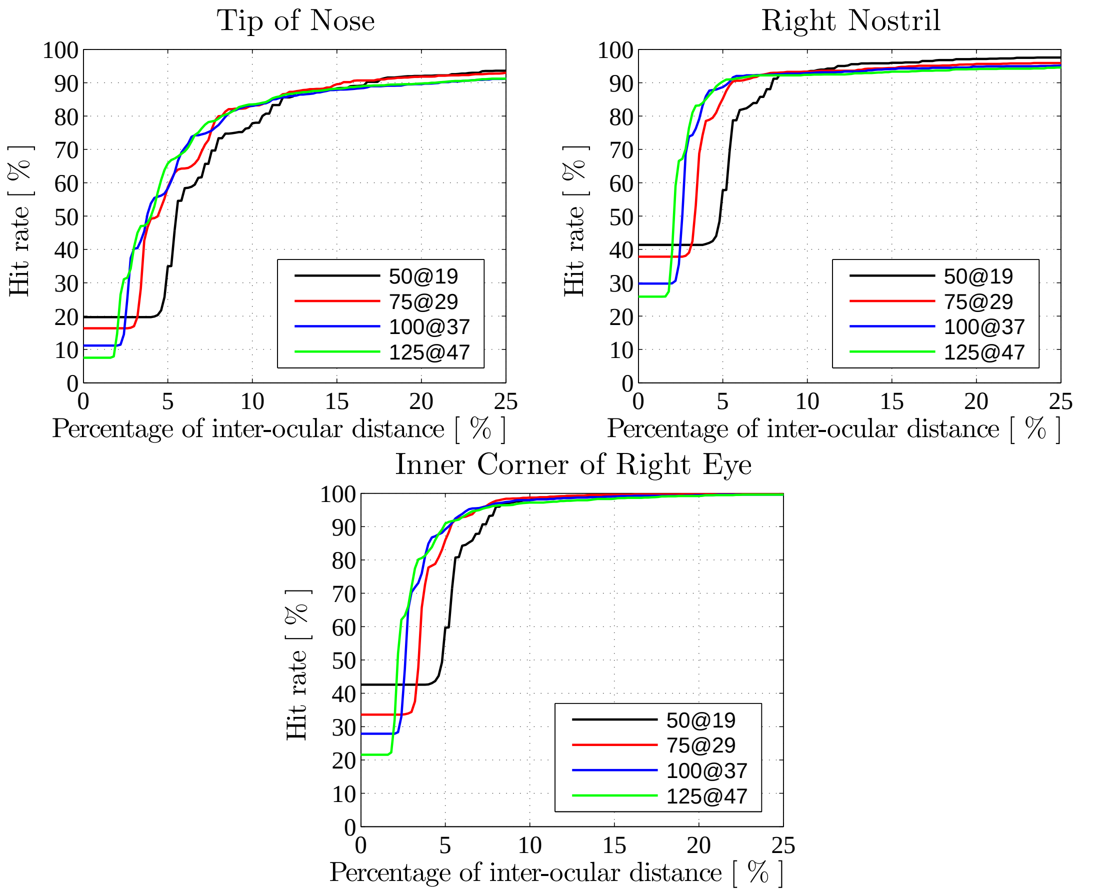
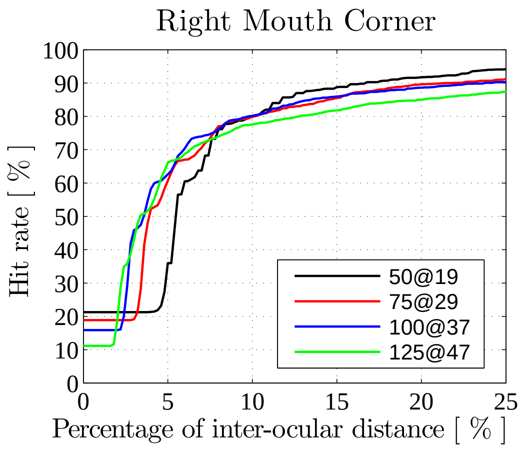
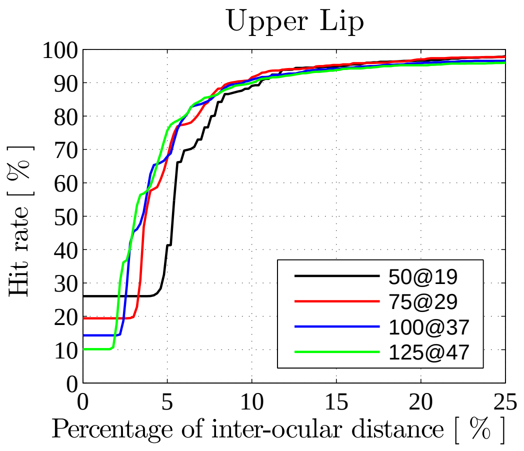


Figure A.29: Average hit rate for the tip of nose and right nostril using faces from 50×50 up to 125×125 pixels and blocks with approximately 38% of the faces's size





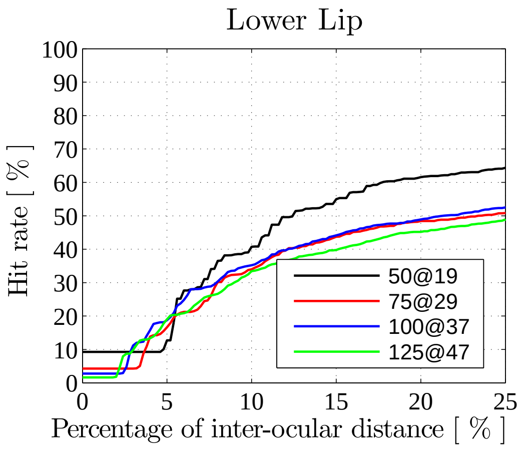


Figure A.30: Average hit rate for right mouth corner, upper lip and lower lip using faces from 50×50 up to 125×125 pixels and blocks with approximately 38% of the faces's size

A.2 Cross-dataset validation between BioID and Color FERET

In order to evaluate the capability of generalization of our method, we employed a cross dataset validation between BioID and FERET. The results are shown in Figures A.31 to Figures A.34. The curves labeled as FERET represent the experiments in which we trained the IPD using BioID and ran the test in the FERET. The converse has been made for the curves labeled as BioID.

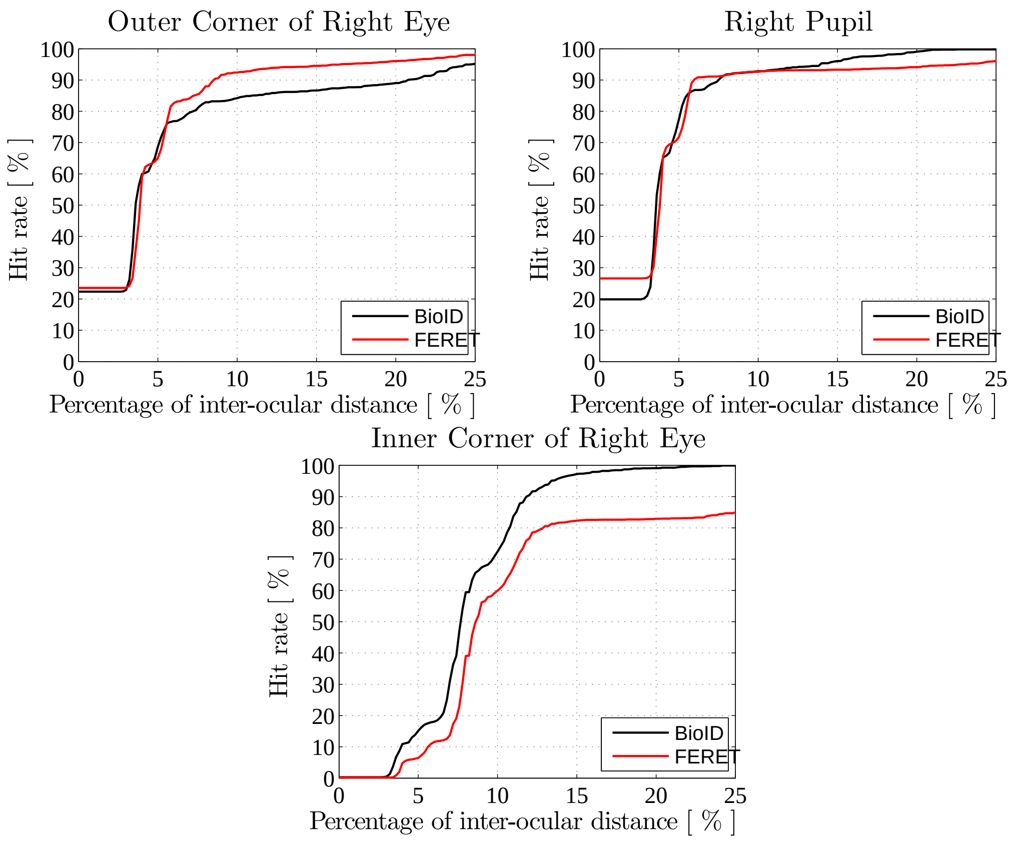


Figure A.31: Cross dataset validation results for points on the right eye. Red curves: training with BioID and testing on FERET. Black curves: trainig with FERET and testing on BioID.

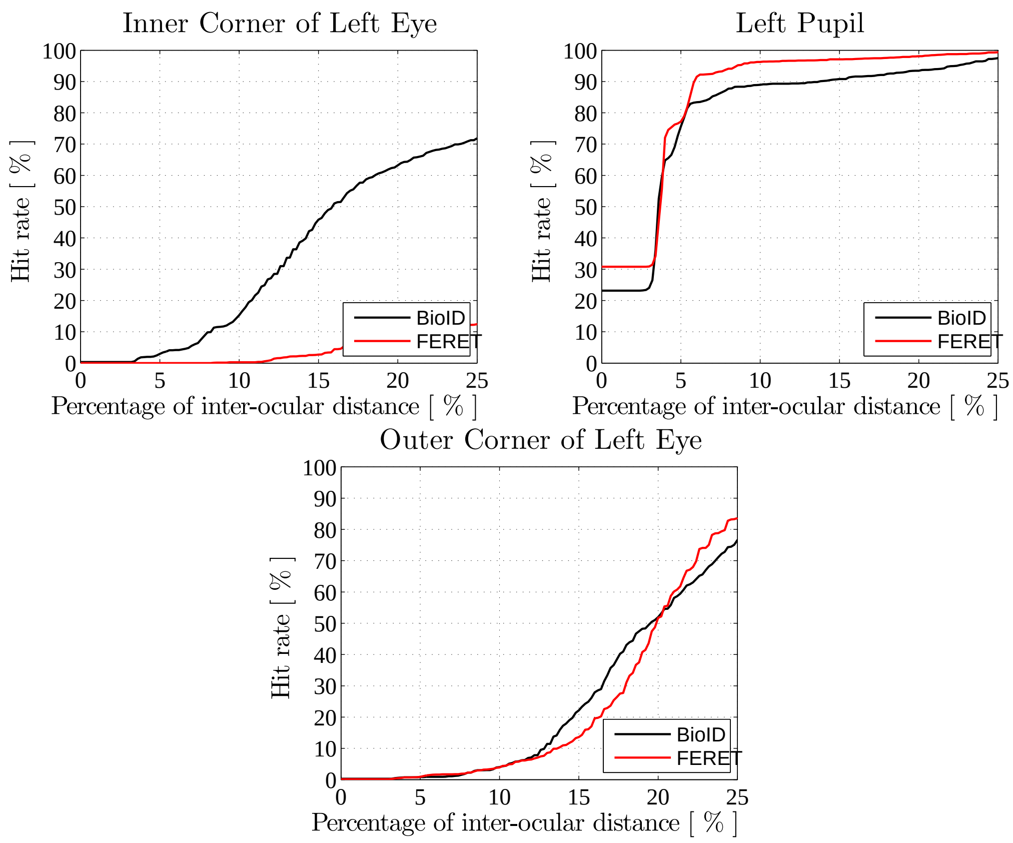


Figure A.32: Cross dataset validation results for points on the left eye. Red curves: training with BioID and testing on FERET. Black curves: trainig with FERET and testing on BioID.

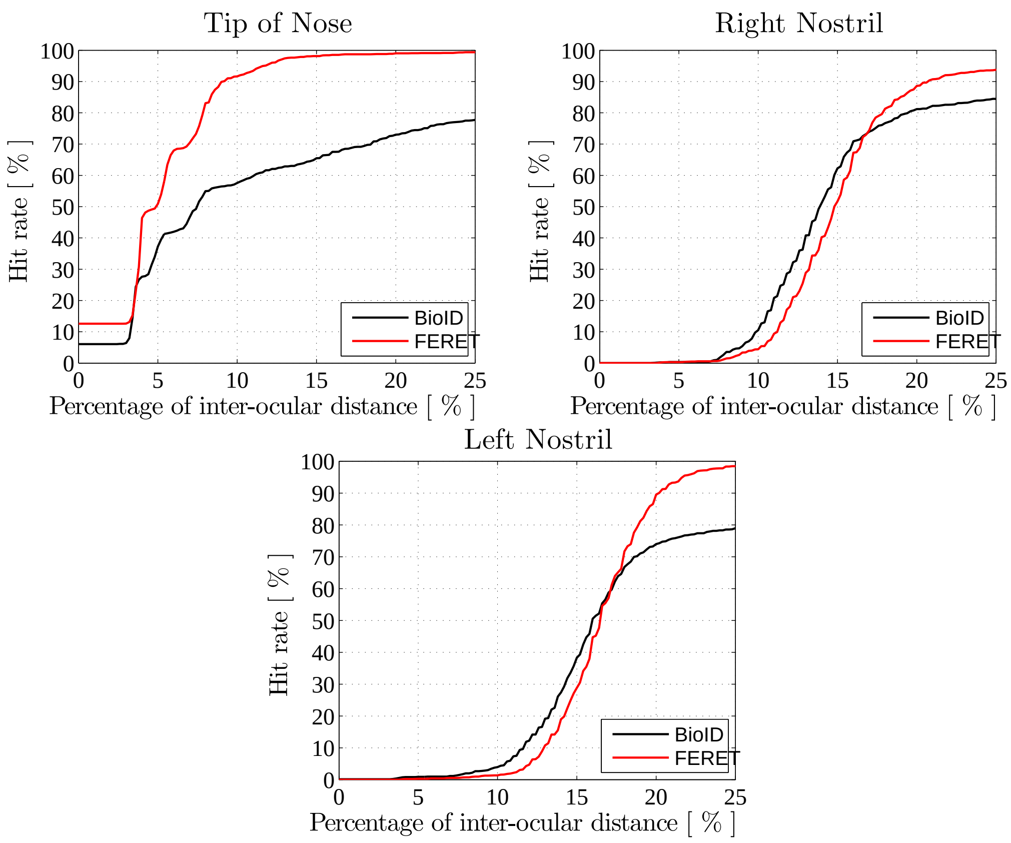


Figure A.33: Cross dataset validation results for points on the nose. Red curves: training with BioID and testing on FERET. Black curves: trainig with FERET and testing on BioID.

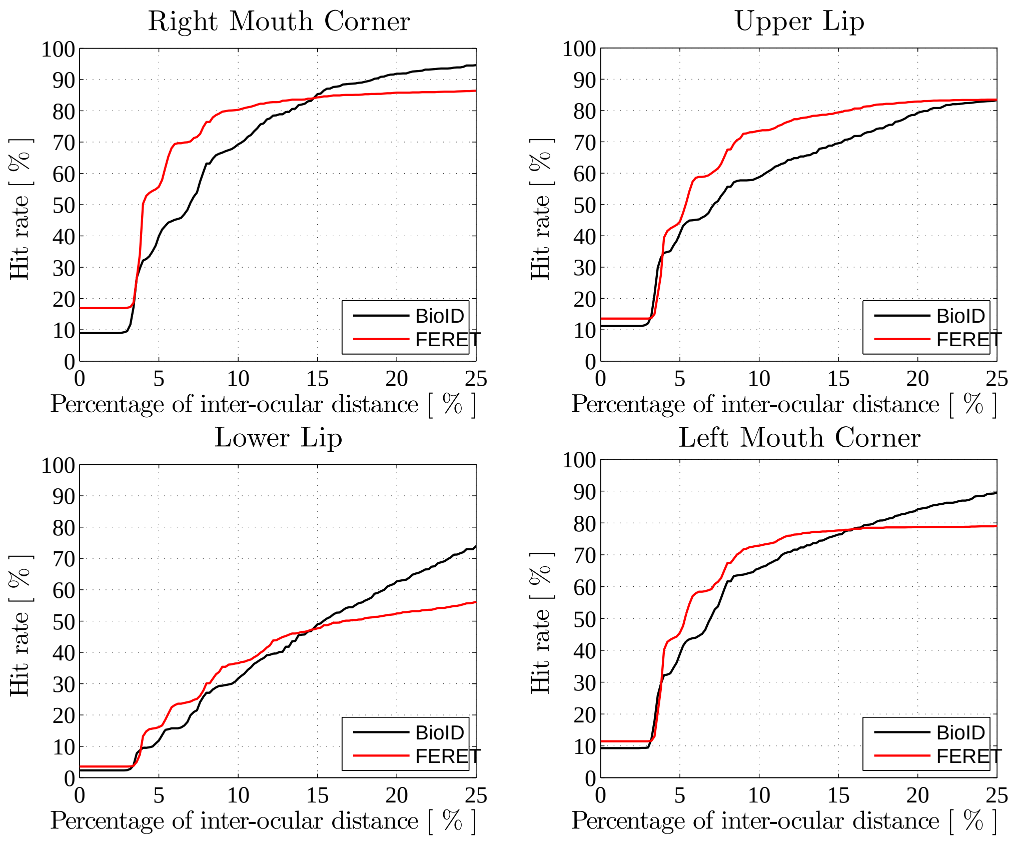


Figure A.34: Cross dataset validation results for points on the mouth. Red curves: training with BioID and testing on FERET. Black curves: trainig with FERET and testing on BioID.

A.2.1 Comparison Between IPD and SVM-RBF

In Figures A.35 to A.43 we show the local results of all 17 inner landmarks for both IPD and SVM (using the local metric as explained in Subsection 3.6.1). In the Figures A.35 to A.37, we have a comparison between IPD and the SVM for the landmarks of the eye.

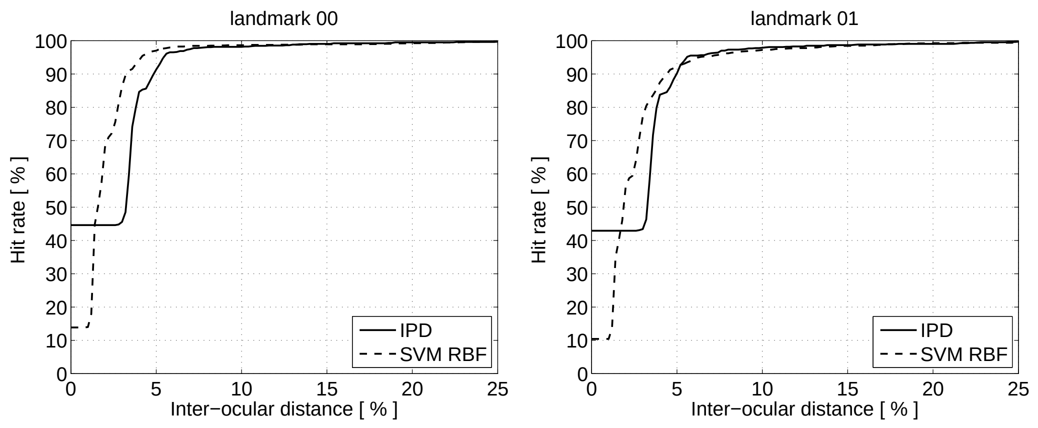


Figure A.35: *Left*: the results for the right pupil. *Right*: the results for the left pupil.

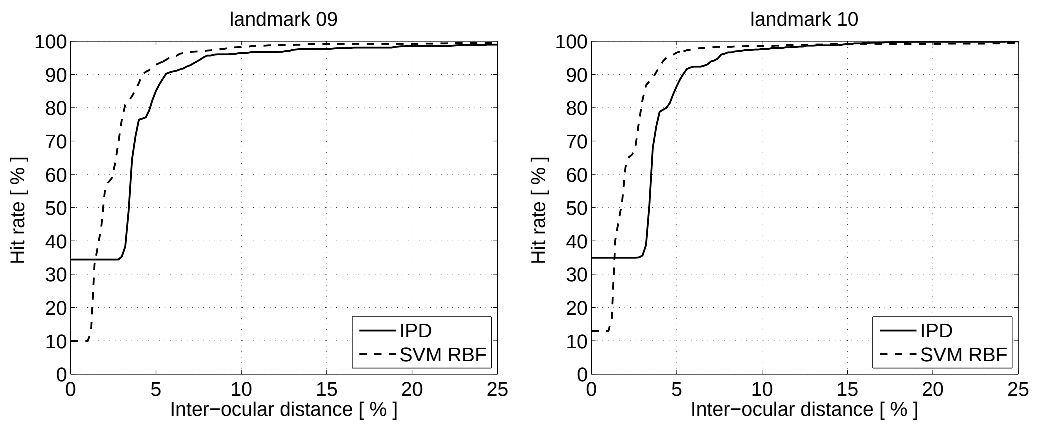


Figure A.36: *Left*: the results for the outer corner of right eye. *Right*: the results for the inner corner of right eye.

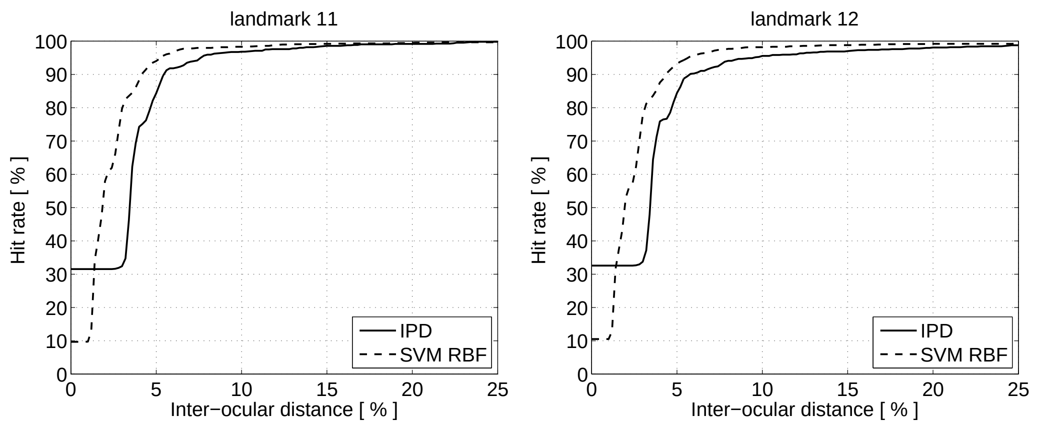


Figure A.37: *Left*: the results for the inner corner of left eye. *Right*: the results for the outer corner of left eye.

In the Figures A.38 and A.39, we have a comparison between IPD and the SVM for the landmarks on the eyebrows.

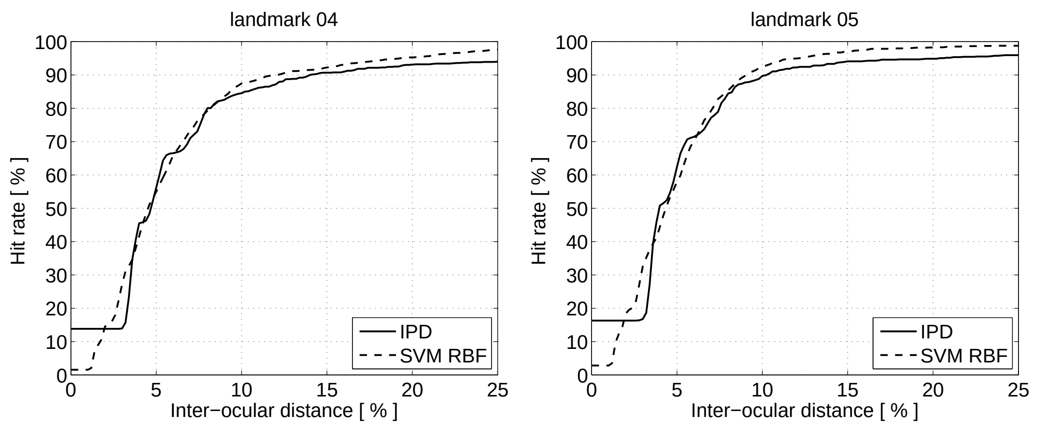


Figure A.38: *Left*: the results for the outer end of right eye brow. *Right*: the results for the inner end of right eye brow.

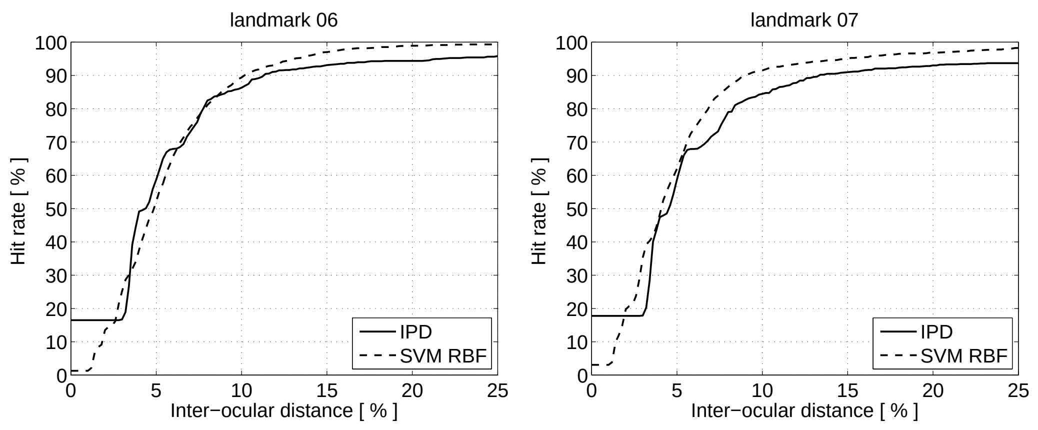


Figure A.39: *Left*: the results for the inner end of left eye brow. *Right*: the results for the outer end of left eye brow.

In the Figures A.38 and A.39, we have a comparison between IPD and the SVM for the landmarks on the nose.

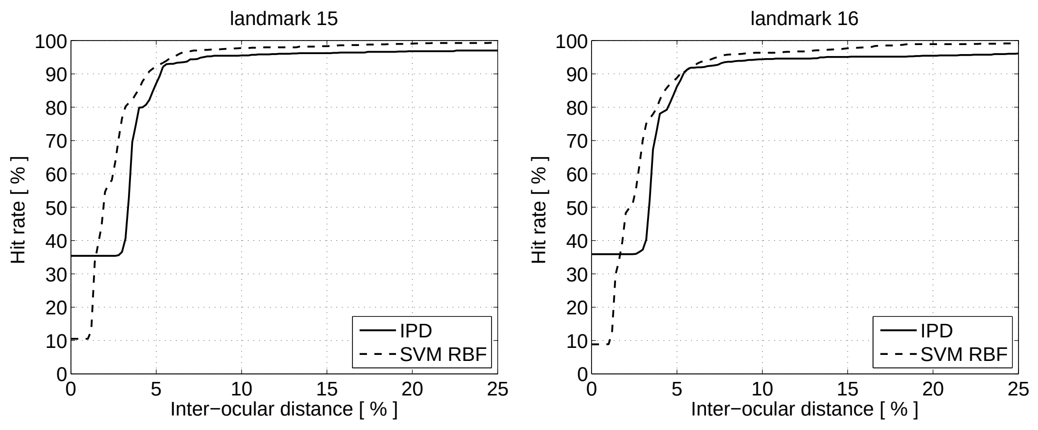


Figure A.40: *Left*: the results for the right nostril. *Right*: the results for the left nostril.

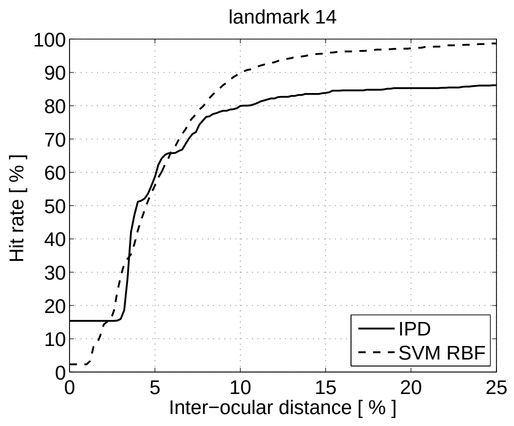


Figure A.41: Results for the tip of nose.

In the Figures A.42 and A.43, we have a comparison between IPD and the SVM for the landmarks on the mouth.

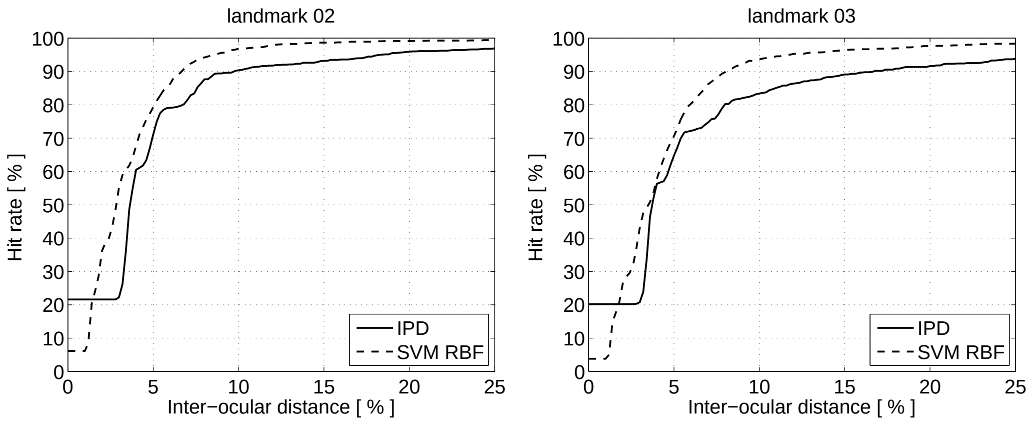


Figure A.42: *Left*: the results for the right mouth corner. *Right*: the results for the left mouth corner.

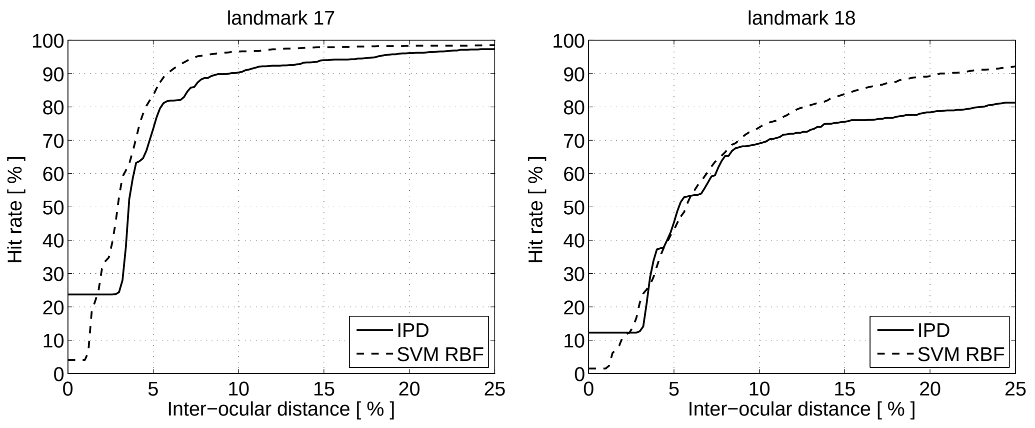


Figure A.43: *Left*: the results for the central point on outer edge of upper lip. *Right*: the results for central point on outer edge of lower lip.

Appendix B

Complementary Results on Landmarks Tracking

In Figures B.1 to B.5, we compare the IPD (referred to as IPD in our graphics), IPD feeding a Particle Filter using Temporal Constraints (referred to as IPD + TC) and the IPD with Particle Filter using Temporal and Geometrical Constraints (referred to as IPD + TGC in our graphs). In all graphs, we considered both, false positives and false negatives as errors.

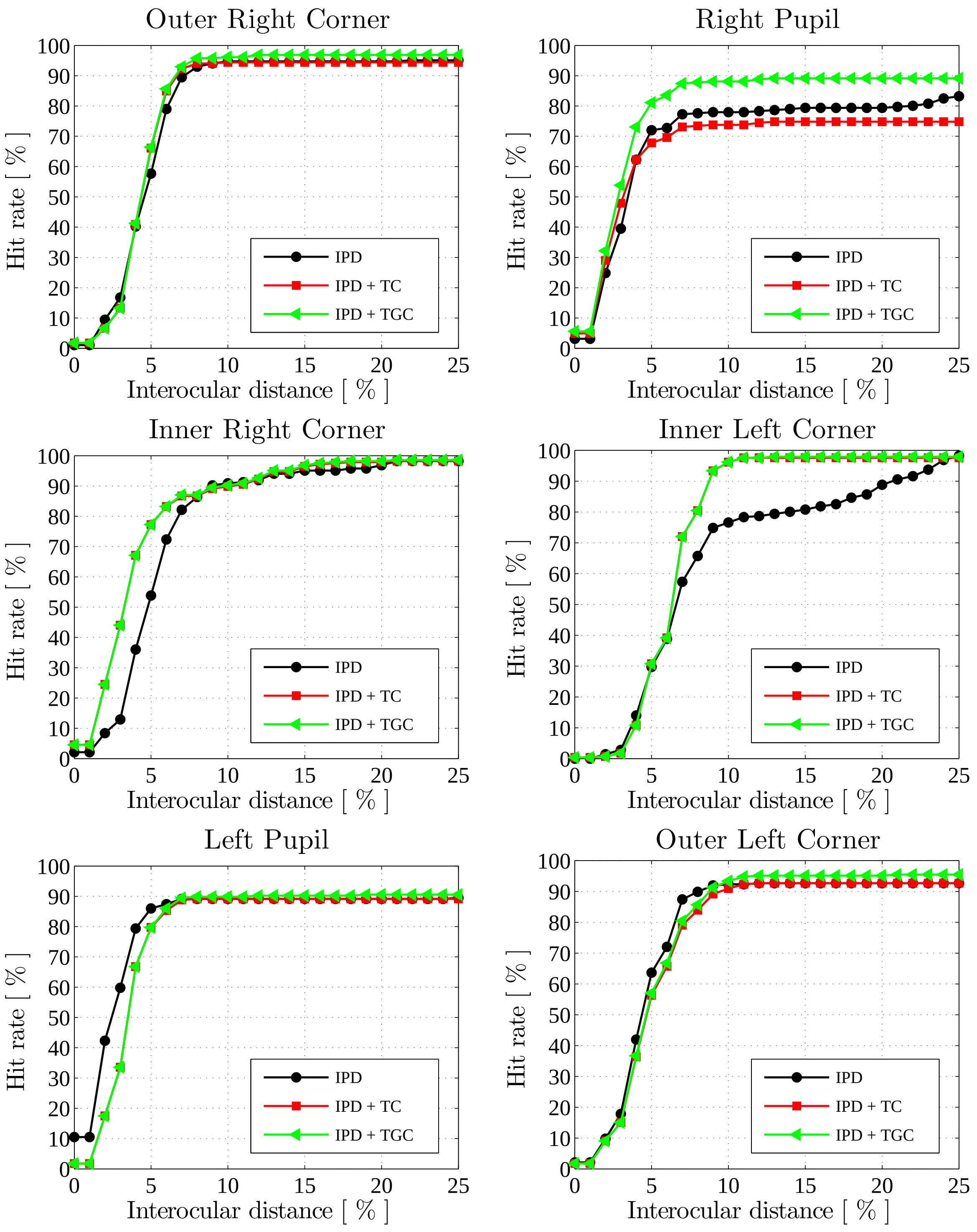


Figure B.1: Comparison between IPD (referred to as IPD), IPD feeding a Particle Filter using Temporal Constraints (referred to as IPD + TC) and the IPD with Particle Filter using the Temporal and Geometrical Constraints (referred to as IPD + TGC). The results were obtained using Sequence 18 (an easy sequence).

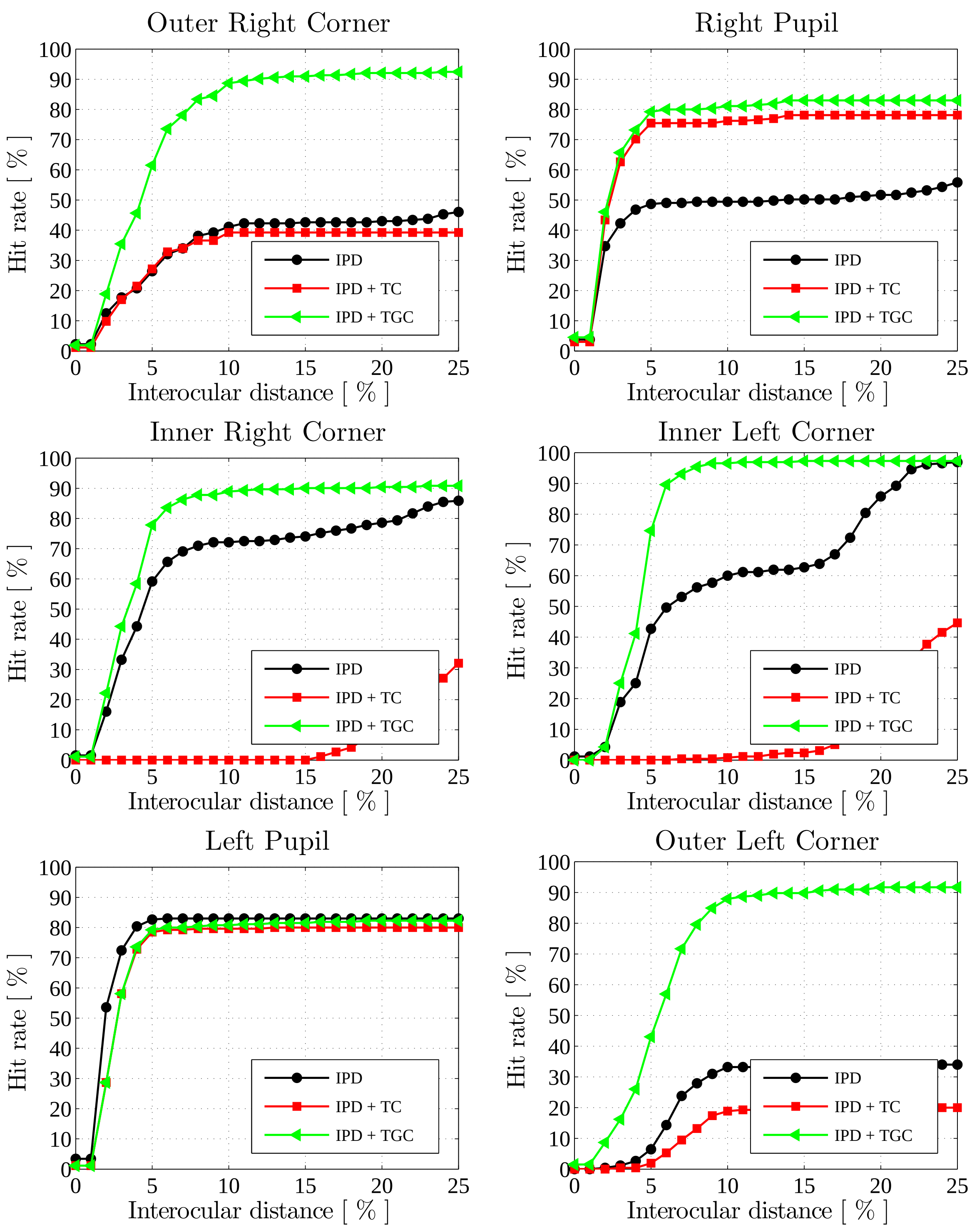


Figure B.2: Comparison between IPD (referred to as IPD), IPD feeding a Particle Filter using Temporal Constraints (referred to as IPD + TC) and the IPD with Particle Filter using the Temporal and Geometrical Constraints (referred to as IPD + TGC). The results were obtained using Sequence 14 (an intermediate/difficult sequence).

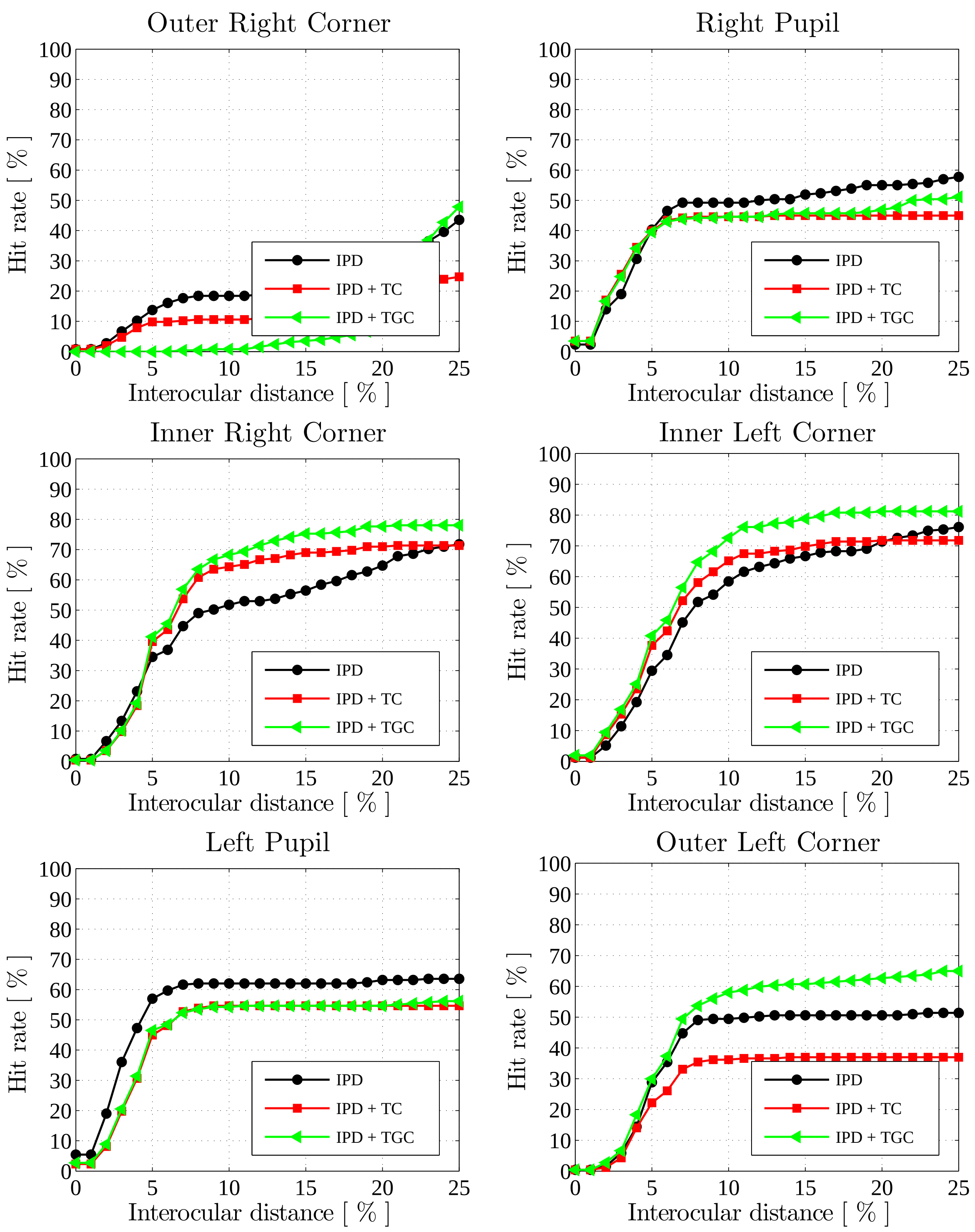


Figure B.3: Comparison between IPD (referred to as IPD), IPD feeding a Particle Filter using Temporal Constraints (referred to as IPD + TC) and the IPD with Particle Filter using the Temporal and Geometrical Constraints (referred to as IPD + TGC). The results were obtained using Sequence 08 (an intermediate/difficult sequence).

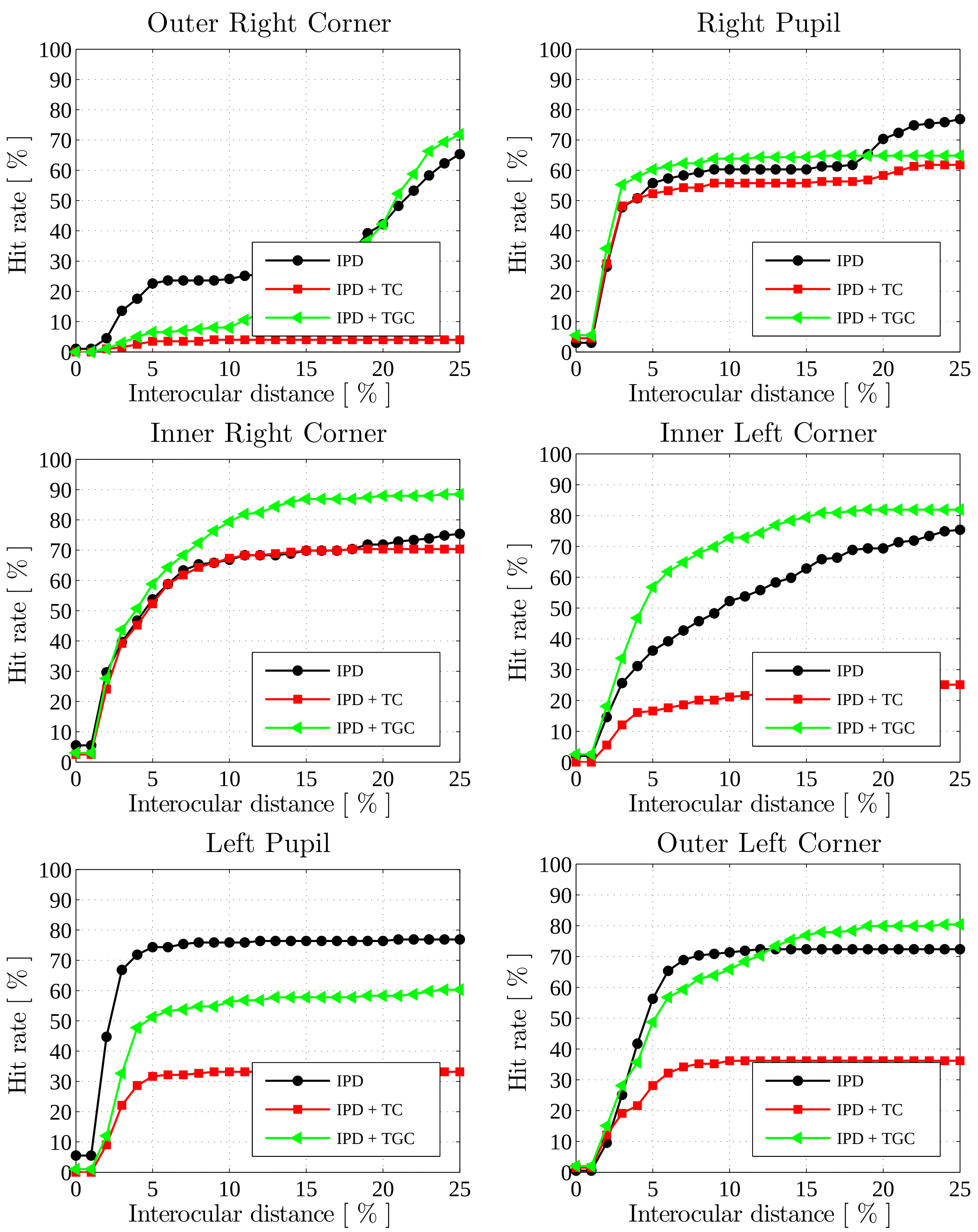


Figure B.4: Comparison between IPD (referred to as IPD), IPD feeding a Particle Filter using Temporal Constraints (referred to as IPD + TC) and the IPD with Particle Filter using the Temporal and Geometrical Constraints (referred to as IPD + TGC). The results were obtained using Sequence 02 (a hard sequence).

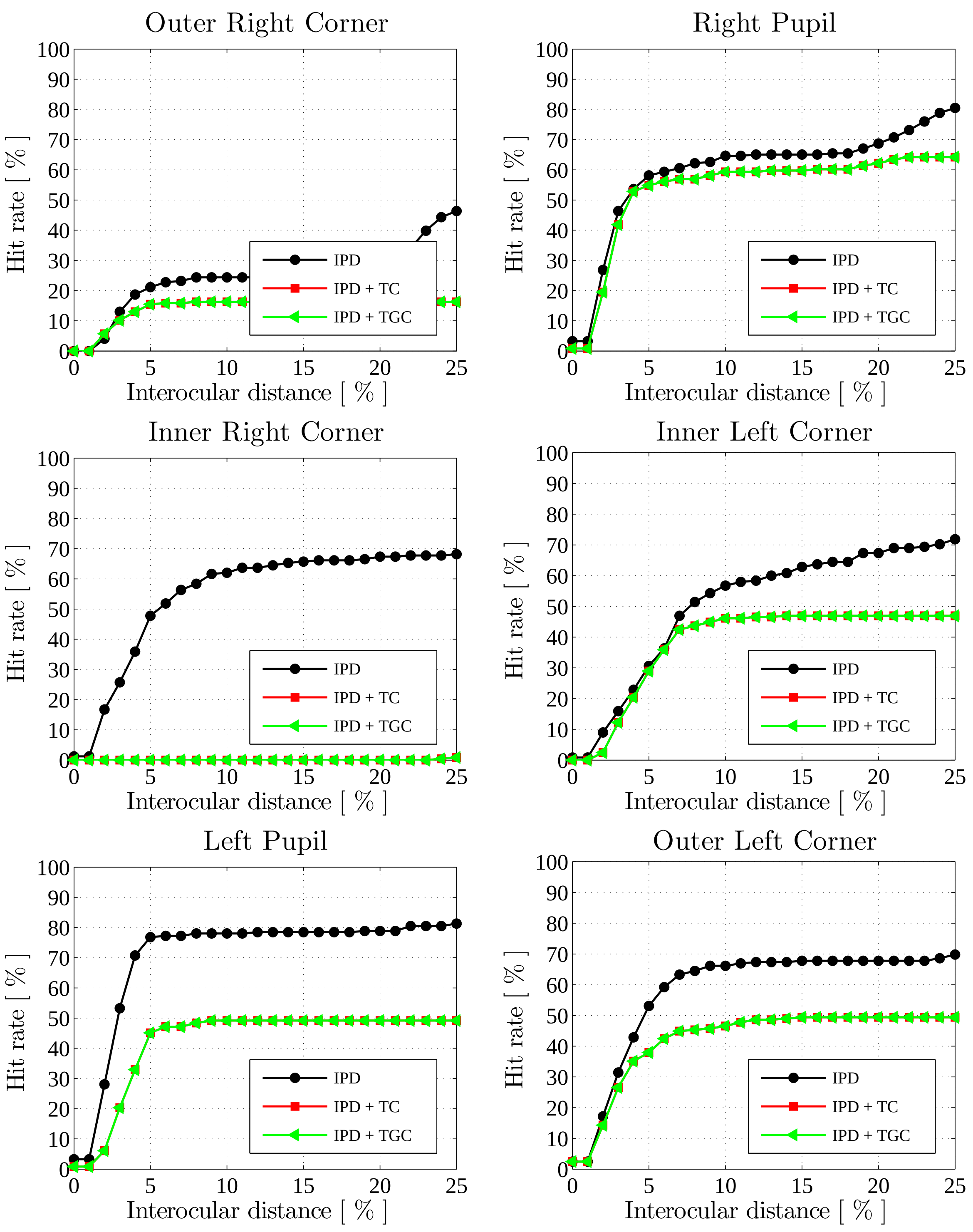


Figure B.5: Comparison between IPD (referred to as IPD), IPD feeding a Particle Filter using Temporal Constraints (referred to as IPD + TC) and the IPD with Particle Filter using the Temporal and Geometrical Constraints (referred to as IPD + TGC). The results were obtained using Sequence 01 (a hard sequence).

In Figures B.6 to B.10, we evaluate the behavior of the errors when using the Temporal Constraints (referred to as TC in our graphs) and using it with the Geometrical Constraints added (referred to as TGC). To do so, we plot the hit rate two times for each method in these figures. One considering only False Positives (referred to as FP in our graphs) as errors and other considering both, False Positives and False Negatives (referred to as FP + FN) as errors.

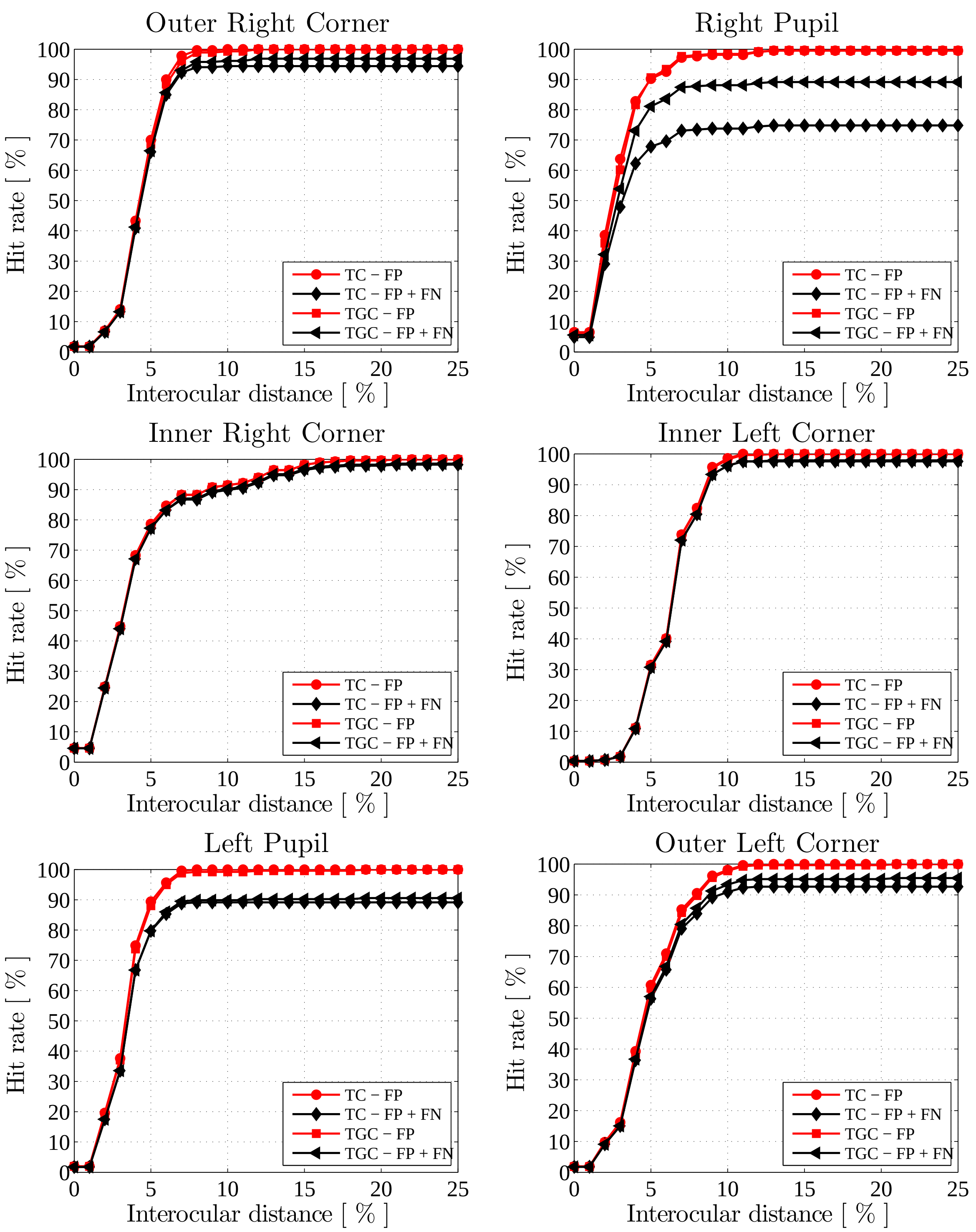


Figure B.6: Results for Sequence 18 (an easy sequence). FP stands for False Positive and FN stands for False Negative. The curves refereed to as TC were obtained by using Temporal consistency only. The curves referred to as TGC were obtained by employing both Temporal and Geometrical Constraints (together with the strategy to evaluate the obtained model)

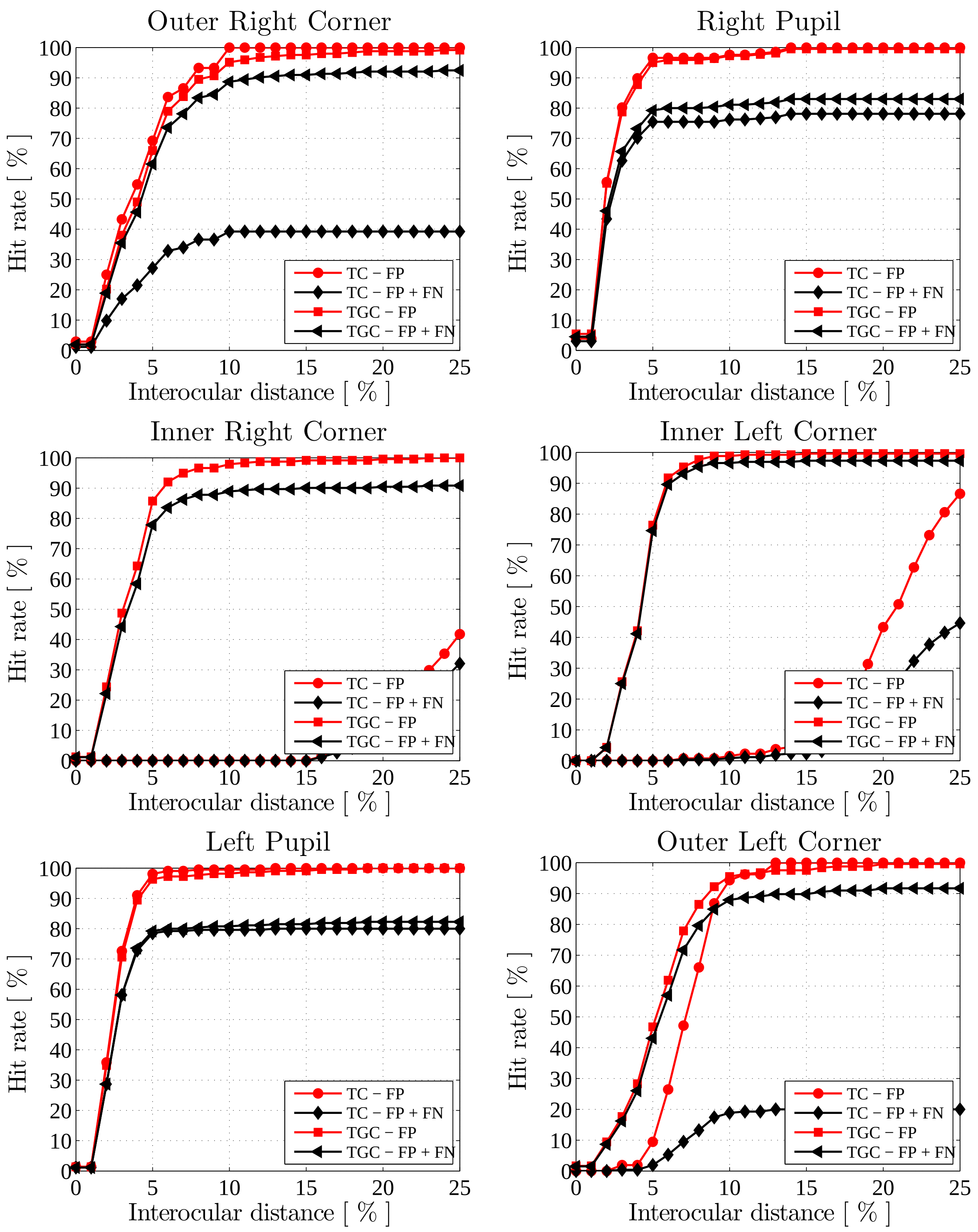


Figure B.7: Results for Sequence 14 (an intermediate/difficult sequence). FP stands for False Positive and FN stands for False Negative. The curves refereed to as TC were obtained by using Temporal consistency only. The curves referred to as TGC were obtained by employing both Temporal and Geometrical Constraints (together with the strategy to evaluate the obtained model)

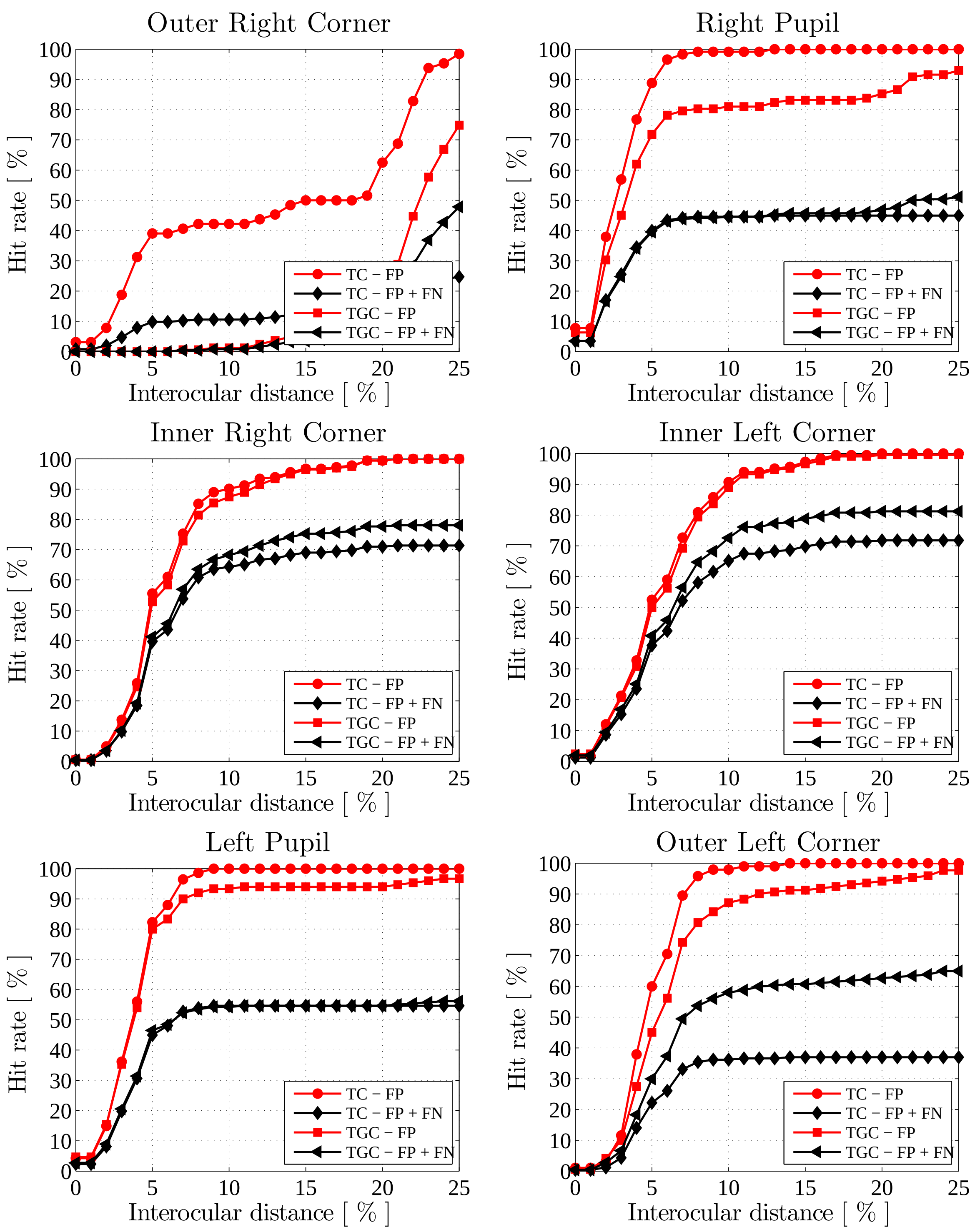


Figure B.8: Results for Sequence 08 (an intermediate/difficult sequence). FP stands for False Positive and FN stands for False Negative. The curves refereed to as TC were obtained by using Temporal consistency only. The curves referred to as TGC were obtained by employing both Temporal and Geometrical Constraints (together with the strategy to evaluate the obtained model)

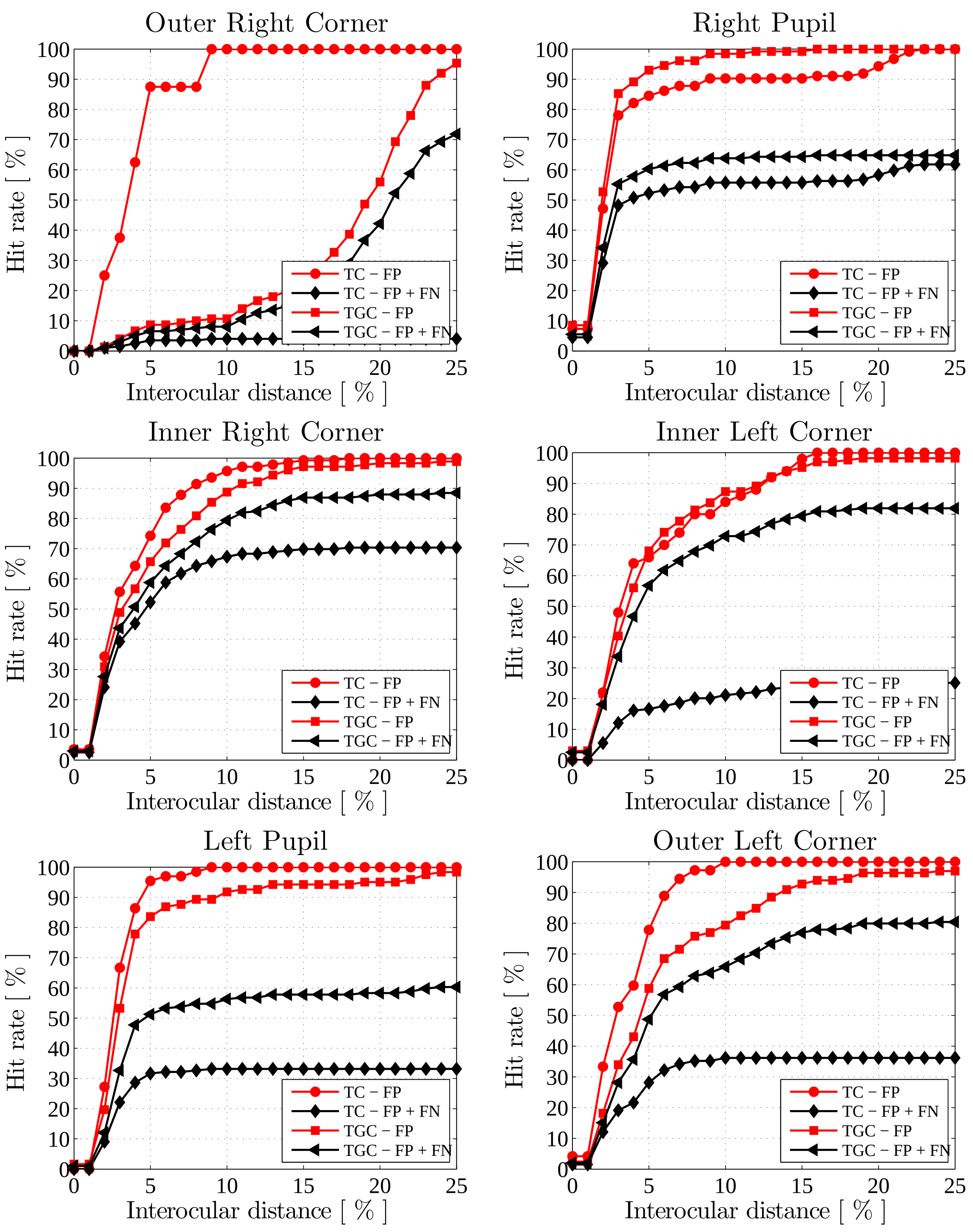


Figure B.9: Results for Sequence 02 (a hard sequence). FP stands for False Positive and FN stands for False Negative. The curves refereed to as TC were obtained by using Temporal consistency only. The curves referred to as TGC were obtained by employing both Temporal and Geometrical Constraints (together with the strategy to evaluate the obtained model)

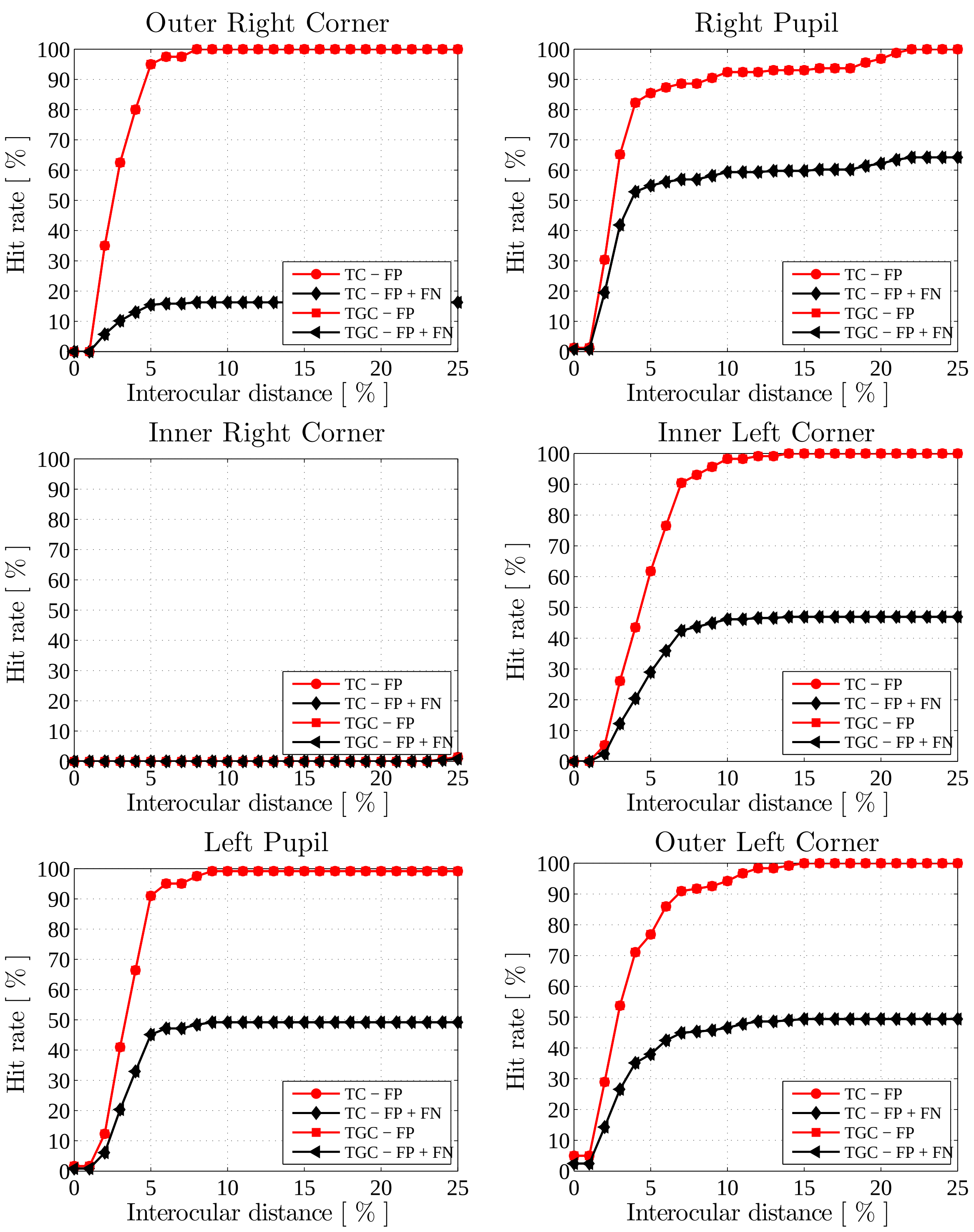


Figure B.10: Results for Sequence 01 (a hard sequence). FP stands for False Positive and FN stands for False Negative. The curves refereed to as TC were obtained by using Temporal consistency only. The curves referred to as TGC were obtained by employing both Temporal and Geometrical Constraints (together with the strategy to evaluate the obtained model)

Our method was designed to work with different types of trackers. From Figures B.11 to B.15, we evaluate the impact of changing the tracking method. In both figures, we compare the IPD (referred to as IPD in our graphics), IPD feeding a Kanade-Lucas tracker using Temporal Constraints (referred to as IPD + TC) and the IPD with kanade-Lucas using Temporal and Geometrical Constraints (referred to as IPD + TGC in our graphs). In all graphs, we considered both false positives and false negatives as errors.

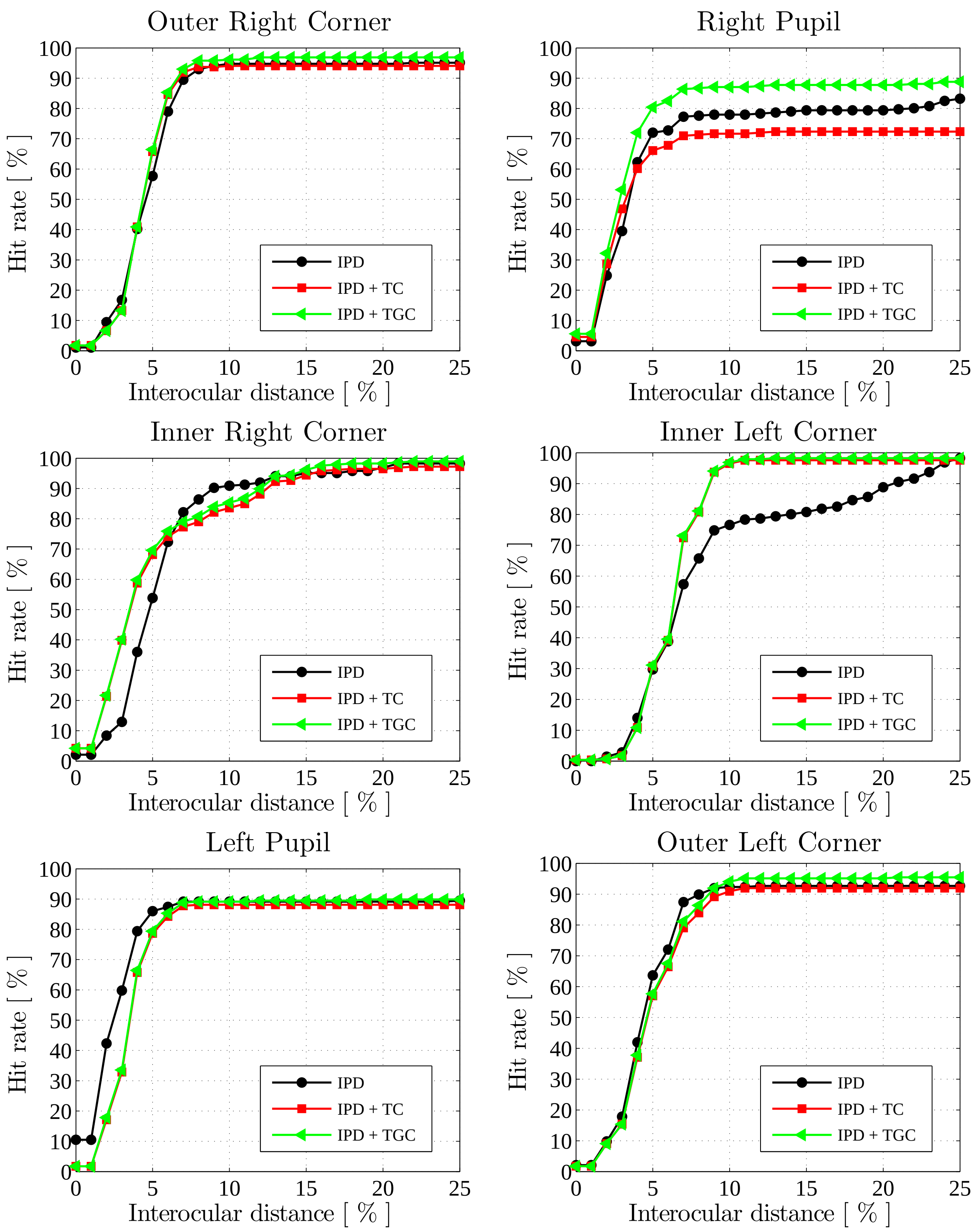


Figure B.11: Comparison between IPD (referred to as IPD), IPD feeding a Kanade-Lucas tracker using Temporal Constraints (referred to as IPD + TC) and the IPD with Kanade-Lucas using the Temporal and Geometrical Constraints (referred to as IPD + TGC). The results were obtained using Sequence 18 (an easy sequence).

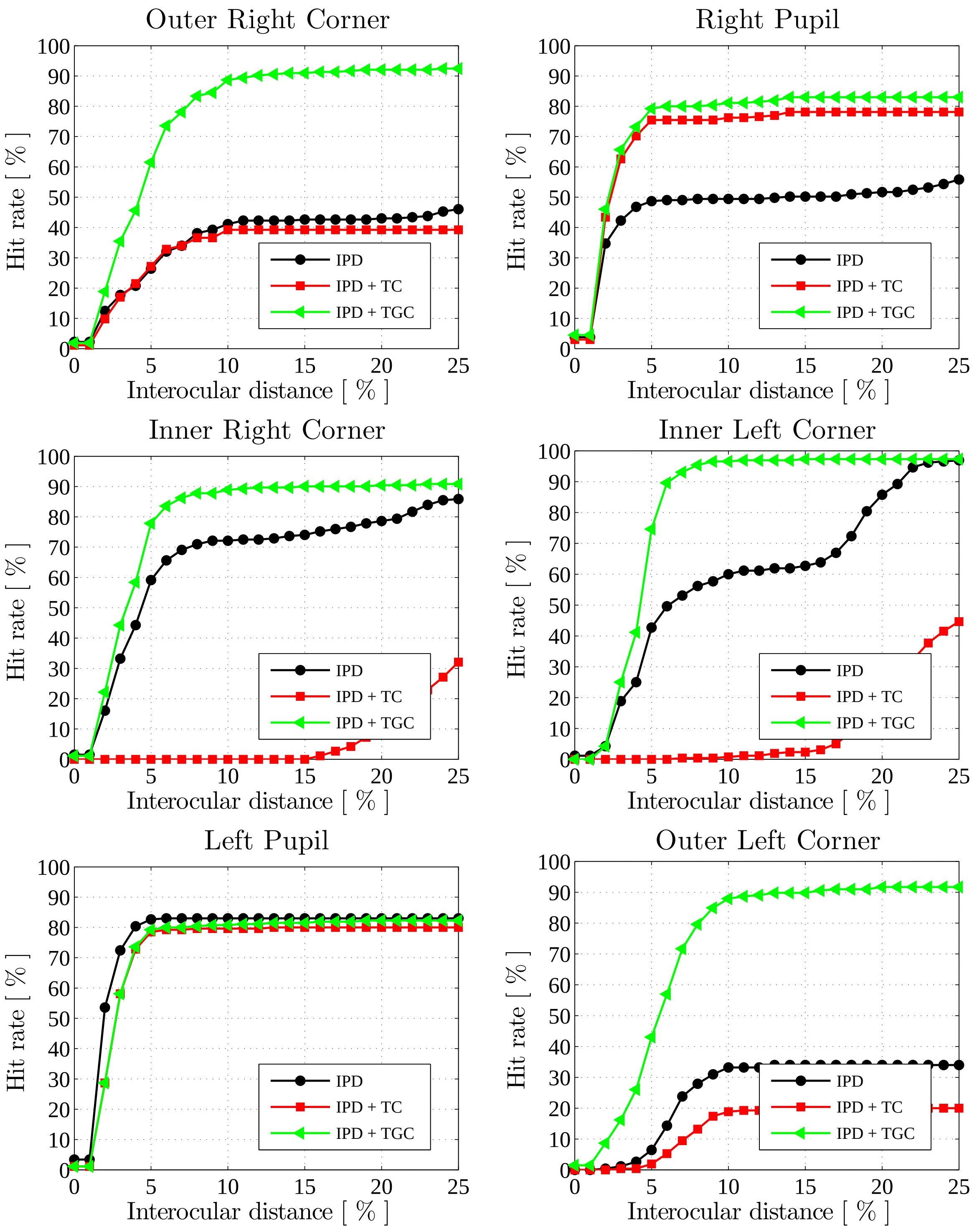


Figure B.12: Comparison between IPD (referred to as IPD), IPD feeding a Kanade-Lucas tracker using Temporal Constraints (referred to as IPD + TC) and the IPD with Kanade-Lucas using the Temporal and Geometrical Constraints (referred to as IPD + TGC). The results were obtained using Sequence 14 (an intermediate/difficult sequence).

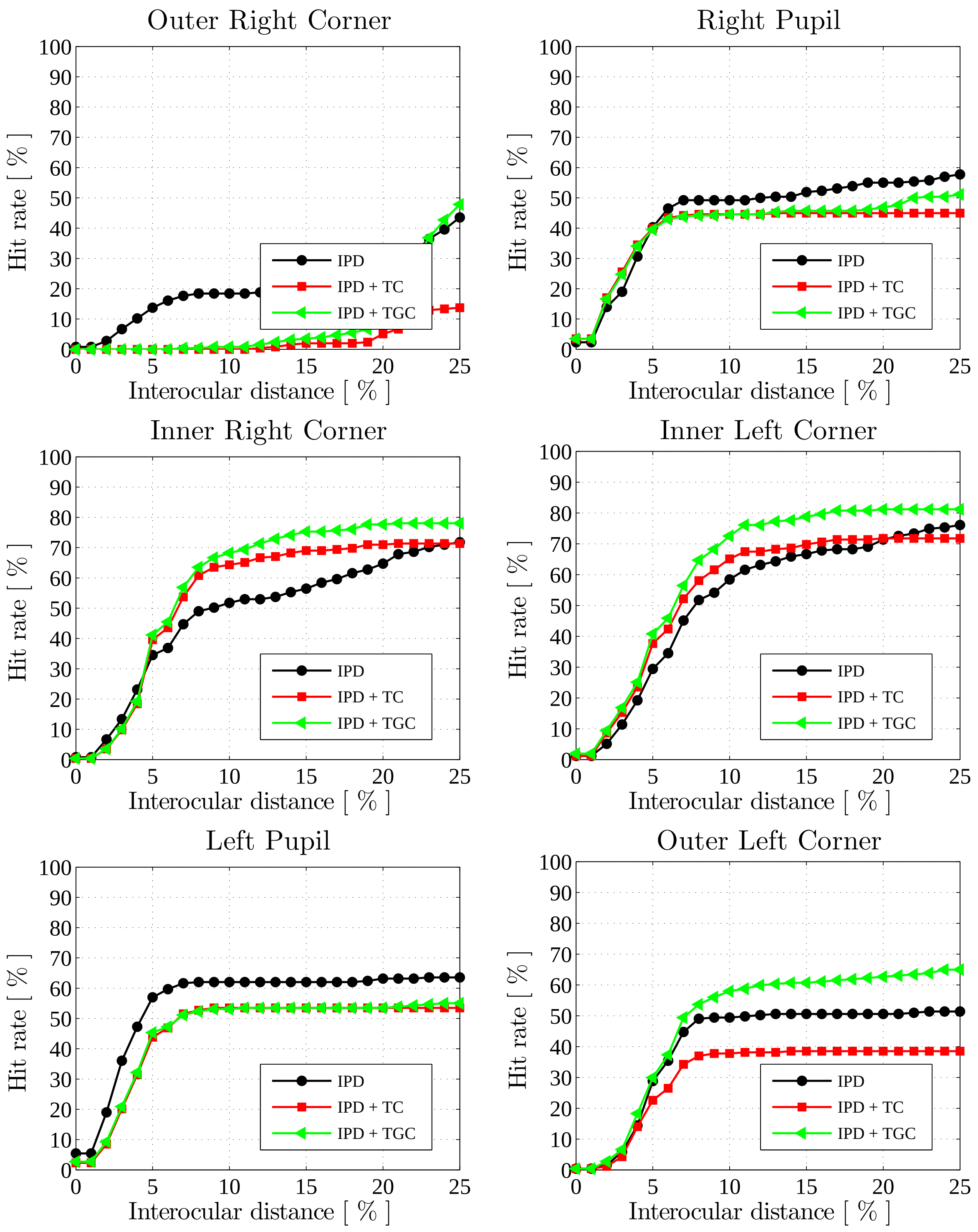


Figure B.13: Comparison between IPD (referred to as IPD), IPD feeding a Kanade-Lucas tracker using Temporal Constraints (referred to as IPD + TC) and the IPD with Kanade-Lucas using the Temporal and Geometrical Constraints (referred to as IPD + TGC). The results were obtained using Sequence 08 (an intermediate/difficult sequence).

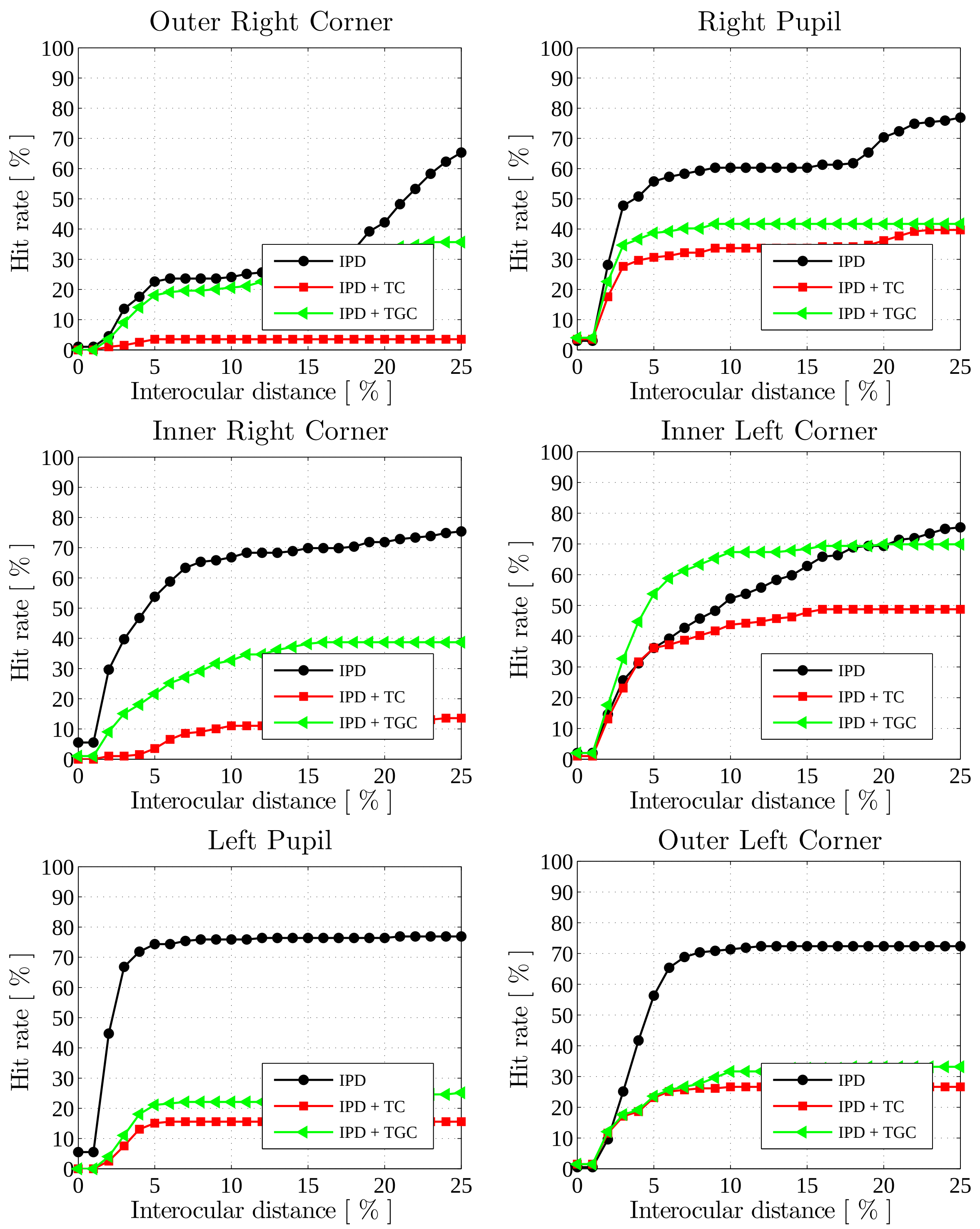


Figure B.14: Comparison between IPD (referred to as IPD), IPD feeding a Kanade-Lucas tracker using Temporal Constraints (referred to as IPD + TC) and the IPD with Kanade-Lucas using the Temporal and Geometrical Constraints (referred to as IPD + TGC). The results were obtained using Sequence 02 (a hard sequence).

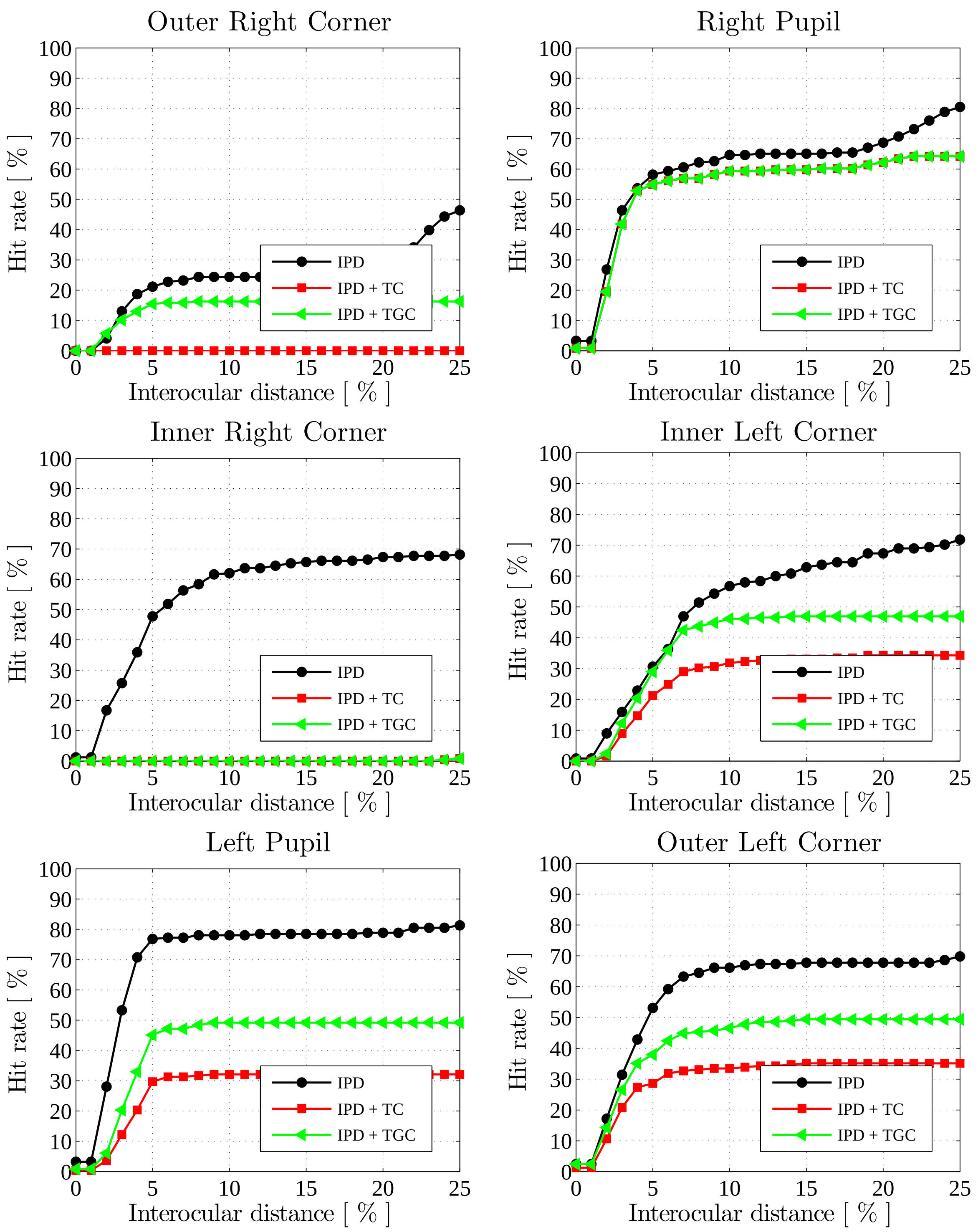


Figure B.15: Comparison between IPD (referred to as IPD), IPD feeding a Kanade-Lucas tracker using Temporal Constraints (referred to as IPD + TC) and the IPD with Kanade-Lucas using the Temporal and Geometrical Constraints (referred to as IPD + TGC). The results were obtained using Sequence 01 (a hard sequence).

List of Publications

This chapter lists the scientific production of this D.Sc. candidate. In summary, such production corresponds to:

- One submitted article in international journal;
- Four articles in international conferences;
- Three articles in national conferences;

In order to facilitate the identification of the type of publication, we use the following tags: SJ = Submitted Journal; IC = International Conference; NC = National Conference. The full list of publications follows:

[SJ1] **Araujo, G. M.**; Ribeiro, F. M. L.; Silva Júnior, W. S.; Da Silva, E. A. B.; Goldenstein, S. K., "Weak One-Class Classifier for Density Estimation in Eye Localization and Tracking," submitted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence.

[IC1] **Araujo, G. M.**; Ribeiro, F. M. L.; Da Silva, E. A. B.; Goldenstein, S. K., "Fast Eye Localization Without a Face Model Using Inner Product Detector," IEEE International Conference on Image Processing (ICIP 2014), Paris, France, pp. 1366-1370, 2014.

[IC2] **Araujo, G. M.**; Da Silva, E. A. B.; Ciancio, A. G.; Oliveira, J. F. L.; Ribeiro, F. M. L.; Said, A., "Integration of Eye Detection and Tracking in Videoconference Sequences Using Temporal Consistency and Geometrical Constraints," IEEE International Conference on Image Processing (ICIP 2012), Orlando, USA, pp. 421-424, 2012.

[IC3] Silva Júnior, W. S.; **Araujo, G. M.**; Da Silva, E. A. B.; Goldenstein, S. K., "Facial Fiducial Points Detection Using Discriminative Filtering on Principal Components," IEEE International Conference on Image Processing (ICIP 2010), Hong Kong, China, pp. 1522-4880, 2010.

[IC4] **Araujo, G. M.**; Silva Júnior, W. S.; Da Silva, E. A. B.; Goldenstein, S. K., "Facial Landmarks Detection Based on Correlation Filters," International Telecommunication Symposium (ITS 2010), Manaus, Brazil, pp. 1-5, 2010.

[NC1] **Araujo, G. M.**; Da Silva, E. A. B.; Ribeiro, F. M. L.; Oliveira, J. F. L.; Ciancio, A. G.; Said, A., "Rastreamento Robusto de Olhos Usando Consistência Temporal e Restrições Geométricas", Simpósio Brasileiro de Telecomunicações (SBrT 2013), Fortaleza, Brazil, pp. 1-5, 2013.

[NC2] Ribeiro, F. M. L.; **Araujo, G. M.**; Da Silva, E. A. B.; Oliveira, J. F. L.; Goldenstein, S. K., "Detecção de pontos fiduciais sobre a face em tempo real", Simpósio Brasileiro de Telecomunicações (SBrT 2012), Brasilia, Brazil, pp. 1-5, 2012.

[NC3] **Araujo, G. M.**; Silva Júnior, W. S.; Da Silva, E. A. B.; Goldenstein, S. K., "Detecção de Landmarks Faciais Usando SVM", Simpósio Brasileiro de Telecomunicações (SBrT 2011), Curitiba, Brazil, pp. 1-5, 2011.