UCLA
UCLA Electronic Theses and Dissertations

Title
Perspective Distortion Modeling in Face Images and Object Tracking Library

Permalink
https://escholarship.org/uc/item/90t1n8hq

Author
Valente, Joachim

Publication Date
2014

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

University of California
Los Angeles

Perspective Distortion Modeling in Face Images and Object Tracking Library

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Science in Computer Science

by

Joachim Valente

2014

© Copyright by
Joachim Valente
2014

Abstract of the Thesis

Perspective Distortion Modeling in Face Images and Object Tracking Library

by

Joachim Valente

Master of Science in Computer Science

University of California, Los Angeles, 2014

Professor Stefano Soatto, Chair

In a first chapter we describe a method to model perspective distortion as a one-parameter family of warping functions. This can be used to mitigate its effects on visual recognition, or interactively manipulate the perceived personality. The warps are learned from a novel face dataset and, by comparing orbits spanned by images instead of images themselves, we improve face recognition when small focal lengths are used. Additional applications are presented to image editing, videoconference, and multi-view validation of recognition systems.

A second chapter is devoted to a new versatile and modular open-source cross-platform online object tracking library, designed to be easily usable by the vision community. Object tracking plays a central part in a number of vision problems, and there is no, to date, a ready-to-use and extensible tracking library at the object level.

The thesis of Joachim Valente is approved.

Alexander Sherstov

Wei Wang

Stefano Soatto, Committee Chair

University of California, Los Angeles

2014

To my family...

for their ongoing support

Table of Contents

1 Perspective Distortion Modeling, Learning and Compensation with Application to Face Recognition and Videoconferencing . . . 1

1.1 Introduction . . . 1

1.1.1 Motivation . . . 1

1.1.2 Related Work . . . 4

1.1.3 Organization of this Chapter . . . 5

1.2 A New Face Dataset . . . 6

1.2.1 Focal-Distance Relation . . . 6

1.2.2 Dataset Requirements . . . 7

1.2.3 Protocol . . . 9

1.3 Impact of Perspective Distortion on Face Recognition . . . 10

1.3.1 Face Recognition Algorithms . . . 10

1.3.2 Experiments . . . 12

1.4 Learning Perspective Distortion . . . 13

1.4.1 Formalization . . . 14

1.4.2 Learning the Model . . . 16

1.4.3 When the Source Focal is Unknown . . . 18

1.4.4 Comparison of Orbits for Perspective Distortion Mitigation 18

1.5 Experimental Assessment . . . 19

1.5.1 Qualitative Results . . . 19

1.5.2 Managing Perspective Distortion in Face Recognition and Validation . . . 20

1.6 Conclusion 23

2 An Object Tracking Library 25

2.1 Tracklib, a C++ Library for Object Tracking 26

2.1.1 Design and Principles Behind Tracklib 26

2.1.2 A Few Algorithms 27

2.1.3 How to Use - Easiness of Use 31

2.1.4 How to Extend - Modularity 34

2.2 Multitrack, a Graphical Front-End for Tracklib 34

2.3 Open-Source Deployment 37

References 40

List of Figures

1.1 *Example of dolly zoom on a face. From left to right the COP (center of projection) moves closer to the subject while the focal length is decreased so as to maintain the same size of the face on the image plane. We observe two phenomena: the background seems more and more condensed and the face warps towards an unusual and less attractive appearance.* 2

1.2 *Explanation of formula 1.1. Focal length f and field-of-view θ are related with the diagonal frame length γ via $\tan\frac{\theta}{2} = \frac{\gamma}{2f}$. Field-of-view θ and real-world dimensions w, h and z are related via $\tan\frac{\theta}{2} = \frac{z}{2d}$. Thus $d = \frac{fz}{\gamma}$. For a 3:2 aspect ratio $w = \frac{3}{2}h$ so $z = \frac{h\sqrt{13}}{2}$. Lastly the crop factor is defined as $K = \frac{\gamma_{35}}{\gamma}$.* 7

1.3 *(Top) sample images from our focal-distorted face dataset. It is worth emphasizing that there is no artificial warp or optical aberration, and the perceived difference among the various samples is due solely to the distance. (Bottom) sample images used as dictionary samples.* 10

1.4 *All* 80 *subjects from our dataset.* 11

1.5 *64 landmarks fitted via active appearance models.* 16

1.6 *Face warping GUI. The handles allow to correct the source focal and to control the destination focal. The first panel is the input, the middle panel is the warped face image. When a ground truth face is available it is displayed on the third panel.* 20

1.7 *Application of face unwarping to videoconference streams. (Top) original frames 1, 115 and 403. (Bottom) unwarped versions. Since the focal is known (*$30\,\mathrm{mm}$ *in 35-mm equivalent), the image uncropped and the face frontal, the distance from the subject is estimated using the distance between the eyes, and is then converted to an estimated "focal" using formula 1.1. The face is then warped to* $f = 63\,\mathrm{mm}$ *which corresponds to a viewing distance of* $50\,\mathrm{cm}$*. Not counting the detection of the eyes and the fitting of AAMs, the application runs at an average rate of* $4.1\,\mathrm{s}$ *per* 432×270 *frame, time mostly spent for resampling in Matlab's affine transformation function and for thin-plate spline interpolation. The warp is only applied to the face area and smoothly vanishes on its edges by fixing control points on them before using thin-plate spline interpolation.* 21

1.8 *Success rate of face recognition algorithms with and without pre-processing.* 22

1.9 *Multi-view validation of an underlying single-view recognition system. In this scenario, an impostor uses a photograph pretending to be some authorized subject. The camera controls its own viewing distance and focal length and triggers the shutter from different distances. After scaling and processing, the standard deviation of each landmark trail, averaged over all landmarks, can be thresholded to unveil the supercherie. Single-view approaches would inevitably fail here.* 23

2.1 *Block diagram of a generic online object tracker. Diverging nodes figure choices. Blocks represent C++ classes (pink: background subtractor, blue: filter, green: detector) and orange ellipses are virtual methods. Offline background subtractor should be initialized with a background estimate (or a video stream to estimate it) and detector with initial image and object state.* 28

2.2 *Opening screen of Multitrack.* 36

2.3 *Creating a new project.* 36

2.4 *Playback of the video. Tracker outputs can be shown or hidden by ticking the corresponding checkboxes. Colors are randomly affected but can also be changed if needed.* 37

2.5 *Configuration of a new "tracking task."* 38

2.6 *Running a tracking task. The current frame and object state are displayed in the preview section.* 39

2.7 *Exporting the results of a tracker to a CSV file.* 39

List of Tables

1.1 *Composition and usage of the dataset.* 8

1.2 *Success rate for three face recognition algorithms for each focal length,* 70 mm *being the reference focal length. The learning set is composed of* 5 *images of each individual with different expressions. The success rate is defined as the number of correctly identified subjects over the number of subjects.* 13

Acknowledgments

First and foremost, I want to sincerely thank my advisor, Professor Stefano Soatto, who was for me an incredible source of inspiration, motivation and ideas throughout the two years I spent at UCLA.

My acknowledgements go to those who helped me with theoretical aspects of my work, notably Joshua Hernandez and Jonathan Balzer, with technical issues (Vasiliy Karasev) and to all those who spent time brainstorming on applications and different approaches, especially Nikos Karianakis and Joshua Hernandez.

I thank the professors at various universities who authorized me to use their datasets for my research, including Prof. Peter Peer from University of Ljubljana for the *FRI CVL Face Database* and Prof. Thomas Vetter from University of Basel for the *3D Basel Face Model*.

I am of course thankful to everyone who agreed to take part in the construction of the distorted-face dataset. Special thanks go to Brian Taylor who helped me gather close to one half of the resulting dataset.

Lastly, I would like to thank my elder brother Luca Valente, UCLA EE alumnus and former member of the Vision Lab, who always found the right words to support and advise me.

CHAPTER 1

Perspective Distortion Modeling, Learning and Compensation with Application to Face Recognition and Videoconferencing

Perspective distortion has been shown to have psychological implications and to affect how a person in a photograph is perceived by the viewer. We show that it also has dramatic effects on visual recognition. In this chapter we describe a method to model it and artificially control it. As a result, we are able to manipulate the perceived distance from the center of projection, and hence the perceived personality, given a single image. We quantitatively validate our method through improvement of baseline face recognition algorithms. Last, we show other applications to this method to fields like image editing, videoconference or multi-view face recognition systems.

1.1 Introduction

1.1.1 Motivation

Painters, photographers, filmmakers and computer graphics artists have long known a phenomenon called "dolly zoom" in the cinema industry: changing the viewing distance from a foreground object distorts the background and plays a significant role in visual perception of the scene. More precisely, using a large focal length makes the scene look compressed whereas using a wide-angle, while

keeping the size of the foreground object constant, makes the scene look richer and deeper [11]. This change in visual perception when manipulating the focal length is also obvious in face images (Fig. 1.1). Experiments where subjects were asked to compare faces taken from different distances - after images were scaled so that the face occupies the same area on the image plane - showed that a face viewed from closer looks nicer and more approachable while a face viewed from farther looks smarter and more impressive [31]. A common rule-of-thumb in portrait photography is to use a 50 mm focal length[1] because it places the photographer at the best viewing distance to flatter a person's face.

(a) $f = 70$ *mm* (b) $f = 24$ *mm* (c) $f = 10$ *mm*

Figure 1.1: *Example of dolly zoom on a face. From left to right the COP (center of projection) moves closer to the subject while the focal length is decreased so as to maintain the same size of the face on the image plane. We observe two phenomena: the background seems more and more condensed and the face warps towards an unusual and less attractive appearance.*

Although this phenomenon is sometimes improperly known as "focal distortion," it has nothing to do with an incorrect projection of the scene to the image plane. In fact, [11] shows that computer-generated scenes - using perfect projection - produce the exact same effect. The different emotions felt by the viewer are solely explained by a single parameter: the distance to the object.

Just as it affects perception, perspective distortion can affect the performance of any face recognition system. Our *first goal* in this chapter is to quantify such an

[1] In 35-mm equivalent. From [5]: "Normal focal length lenses most closely approximate human sight and project an image with the least distortion and compression of space from foreground to background." The *normal* focal length is defined as the diagonal length of a square frame.

effect (Table 1.2). This is done by testing different face recognition algorithms on images captured under different focal settings than those used for training. This requires a dataset of images of the same subjects taken from different distances. Given the absence of such a dataset in the public domain, our *second goal* is to construct one.

Having quantified the effect, our *third goal* is to model perspective distortion, and to learn the model parameters from the training set. It is worth emphasizing that perspective distortion is not an artificial warp or an optical aberration, but a complex deformation of the domain of the image due to the combined physical effects of distance and focal length. It depends on the shape of the underlying face, which is typically unknown, and can involve singularities and discontinuities.[2] Nevertheless, it can be represented as a one-parameter family of shape-dependent domain deformations. Neglecting the effects of occlusions, it can be interpreted as a group action generating orbits on the space of face images. Thus our *fourth goal* is to characterize those orbits so as to enable hallucination of perspective distortion, even without knowledge of the underlying shape.

We illustrate this task by interactively manipulating the perceived distance from the camera. In particular, we demonstrate "focal un-distortion" of video-conference and videochat images, that are often perceived as unattractive due to the short focal length of forward-looking cameras in consumer devices.

Finally, our *fifth goal* is to exploit the structure of the orbit space to render face recognition systems insensitive to perspective distortion. This is done by performing comparisons in orbit space, rather than in image space. We validate this method by testing the same face recognition systems studied in our first goal, where each orbit is represented by a canonical element computed via pre-processing.

[2]For instance, the ears of the subject are visible on the right in Fig. 1.3 but not on the left.

1.1.2 Related Work

An application of this work is to face recognition, a field too vast to properly review here (see [44] for a survey of the state-of-the-art as of a decade ago, and [1] for a more recent account). Since our goal is not to introduce a new face recognition algorithm, but to devise a method for any face recognition system to deal with perspective distortion, we select two representative algorithms in section 1.3.1. One is chosen for simplicity, the other because representative of the state-of-the-art.

More specifically, our work aims to reduce nuisance variability. A nuisance is a phenomenon that affects the data, but should ideally not affect the task. Most prior work on handling nuisances in face recognition focused on illumination [17, 35, 2, 19] and pose variation [19, 10], as well as partial occlusion [42], age [29, 24] and facial expressions [1]. To the best of our knowledge, variability due to optics has not been studied in a systematic way, and while its effects on recognition is not as dramatic as illumination or pose variability, it nevertheless can exceed intra-individual variability and thus lead to incorrect identification, especially at short focals.

Many face datasets for recognition are publicly available. The *FERET database* [33], the *AR-Face database* [25] or the *Extended Yale Face Database B* [23] are among the most widely used to benchmark face recognition algorithms. A more thorough review is done in [1]. Despite the number of available datasets, to the best of our knowledge, none tackles the problem of optical variability.

Additionally, our method requires the distance from the subject to be known or estimated. [14] tackles the problem of estimating this distance by solving the camera pose via Effective Perspective-n-Point. We however do not leverage 3D modeling and use a different method (section 1.4.3). Using this estimate to improve face recognition in presence of perspective distortion is also suggested

there.

The psychophysical effects of perspective distortion have been studied in [9]. It is shown to be a crucial factor affecting how a subject is perceived, notably how trustworthy, competent and attractive she looks like. The idea of using some kind of quantification of the perspective distortion, to manipulate the perceived personality, is also mentioned. Inspired by paintings from the Renaissance that use several centers of projection at once to control the viewer's perception, [32] studies how the same effect can be achieved with photographs and shows compelling experiments by combining multiple images of the same scene (a human) taken from different viewpoints, using an image editing software.

1.1.3 Organization of this Chapter

In section 1.2 we describe the dataset we have collected to test the hypothesis that warping due to perspective distortion affects the performance of face recognition. There we detail the protocol we used and further explain the reasons that motivate it.

In section 1.3 we quantify the impact of perspective distortion on face recognition by comparing the performance of several algorithms when the test image was captured from the same distance as training images and when it was captured from a different distance. We show that the effect is negligible for small differences, but significant for large ones.

In section 1.4 we begin addressing the issue of *managing* nuisance variability due to perspective distortion. The derivation we propose is generic, in the sense that it applies to any one-parameter group transformation, and in fact even higher-dimensional groups, provided that the dataset spans a sufficiently exciting sample of the variability. Other examples of applications that we have not considered in this work, but where our method could in principle be applicable, include aging

and expression, but not pose changes that induce self-occlusions.

We present our results in section 1.5, both qualitatively (*i.e.,* visually) and quantitatively (*i.e.,* showing numerical improvements on face recognition success rate). There, we also show an application to un-warping of videoconference and videochat images, to illustrate the synthesis component (as opposed to recognition) of our method. Finally in section 1.6 we discuss possible extensions and applications.

1.2 A New Face Dataset

For testing the hypothesis that perspective distortion affects face recognition, we have generated a protocol and constructed a dataset that comprises 12 images for 80 subjects. Most subjects are in their twenties, Caucasian or Asian, with about 45% females. The dataset spans 7 focal lengths and 5 different expressions for each subject, and is captured against a green screen with photographic studio quality but otherwise uncontrolled illumination.

1.2.1 Focal-Distance Relation

Throughout this work, we assume that the distance between the subject and the center of projection (COP) is varied along with focal length so that the face occupies the same area on the image plane. More precisely, under a simplistic optical model, for an aspect ratio of 3:2, this correspondance is given by:

$$d = f\frac{\sqrt{13}hK}{2\gamma_{35}} \tag{1.1}$$

where d is the distance from the subject to the COP, f is the focal length of the lens, h is the height of the face (typically around 19 cm), K is the crop factor of the image sensor and γ_{35} is the diagonal of a full-frame 35 mm (36 mm $\times$ 24 mm), *i.e.,* $\gamma_{35} = 43.3\,\text{mm}$.

See Fig. 1.2 for an explanation of this formula. Based on this relation we will use the terms "focal" and "distance" interchangeably, although the source of variability is really the distance. The term "focal" will be preferred because easier to control during the construction of the dataset.

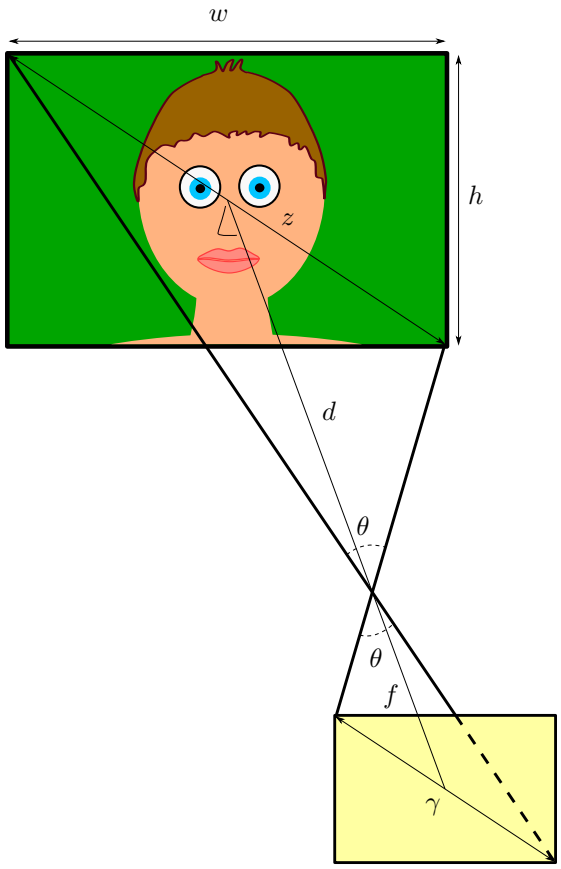


Figure 1.2: *Explanation of formula 1.1. Focal length f and field-of-view θ are related with the diagonal frame length γ via* $\tan\frac{\theta}{2} = \frac{\gamma}{2f}$. *Field-of-view θ and real-world dimensions w, h and z are related via* $\tan\frac{\theta}{2} = \frac{z}{2d}$. *Thus* $d = \frac{fz}{\gamma}$. *For a 3:2 aspect ratio* $w = \frac{3}{2}h$ *so* $z = \frac{h\sqrt{13}}{2}$. *Lastly the crop factor is defined as* $K = \frac{\gamma_{35}}{\gamma}$.

1.2.2 Dataset Requirements

As we explain in section 1.4, our method relies on averaging the dependency on the shape of the underlying face, which improves with the number of samples in the dataset. Of our set, approximately 33 % are to be used in the learning phase,

Image	10 mm / 17 mm / 22 mm / 24 mm / 34 mm / 50 mm / 70 mm	Neutral / Smiling / Angry / Looking left / Joker
33 %	Learning samples for face warping modelization	(Unused)
67 %	Test samples for face recognition	Dictionary samples in face recognition

Table 1.1: *Composition and usage of the dataset.*

and 67 % in testing.

Also, our method models the warping by learning it on face images where perspective distortion is the only nuisance. Therefore illumination is assumed to be constant within the training set, pose is frontal and expression neutral. The number of images we collect should span from wide-angle (10 mm or distance of 12.7 cm) to telephoto (70 mm or distance of 88.6 cm).

Finally, face recognition algorithms require several photos of each subject. Given that for our purpose the pictures described above will serve as test samples, we need an additional set of pictures with different poses and expressions, in order to serve as challenging training set.

Table 1.1 summarizes the composition and usage of our dataset.

1.2.3 Protocol

Each individual was asked to sit on a stool lit on both sides by a 70 W softbox RPS Studio RS-4070 to reduce the effects of cast shadows. Behind them was a green screen[3] to remove background variability.

The camera used was a Canon EOS 30D. For wide-angles we used a Canon lens EF-S 10-22 mm f/3.5-4.5 USM and for medium-range we used a Canon lens EF 25-70 mm f/2.8L II USM. The sensor's crop factor is $K = 1.6$. All photos were shot at 1/60, f5.6, ISO 400 with a white balance fixed at 5000 K. However to ensure uniformity images were further processed to adjust brightness and contrast.

In a first stage subjects were asked to remove their glasses and if needed to put their hair up so as not to hide the eyes and eyebrows. They had to look towards the camera with a frontal pose and neutral expression, but the latter was not strictly enforced, resulting in some minor expression variability. Seven photos were taken in sequence with focals 10 mm, 17 mm, 22 mm, 24 mm, 34 mm, 50 mm and 70 mm.

Then in a second stage they were asked to smile, to vary expressions, to look at a fixed object, resulting in about 30° out-of-plane rotation, to show a neutral frontal expression, and finally to make a "funny face" (akin to the "joker" expression in the IMM Face Dabase [28]).

In a post-processing step, all images were normalized and aligned with respect to the similarity group by placing the eyes in canonical position, as customary. Fig. 1.3 shows the resulting 12 samples for one of the subjects used in this work, and Fig. 1.4 shows all 80 subjects in a neutral pose.

[3]The professional setting for this exercise consists in placing the subject at around 8 feet from the green screen, lighting the screen on both sides to enforce its uniformity and lighting the subject on both sides with different sources. For our purpose however perfect uniformity of the screen was not required so placing the subject directly against it only produced acceptable shadows.

Figure 1.3: *(Top) sample images from our focal-distorted face dataset. It is worth emphasizing that there is no artificial warp or optical aberration, and the perceived difference among the various samples is due solely to the distance. (Bottom) sample images used as dictionary samples.*

1.3 Impact of Perspective Distortion on Face Recognition

In this section we examine how the particular variability due to perspective distortion influences recognition success rate. Although many algorithms are designed to be insensitive to various sources of variability, we show that in practice extreme distortions lead to incorrect identifications.

1.3.1 Face Recognition Algorithms

We will consider two families of face recognition systems. The first one (EIGENDETECT) is chosen for simplicity, based on the assumption that a linear subspace captures the within-class variability, as suggested in [39, 37]. The resulting "eigenfaces" then capture the principal components of the space spanned by the samples in the training database.

The second algorithm is considered state-of-the-art and based on sparse representation coding (SRC) [42]: given learnt faces $(I_i)_{i=1}^n$ put side by side in a dictionary-matrix A, solve the ℓ^0 minimization problem

$$\min \|x\|_0 \text{ s.t. } Ax = I \ . \tag{1.2}$$

This NP-complete problem is relaxed to an ℓ^1 minimization which naturally yields

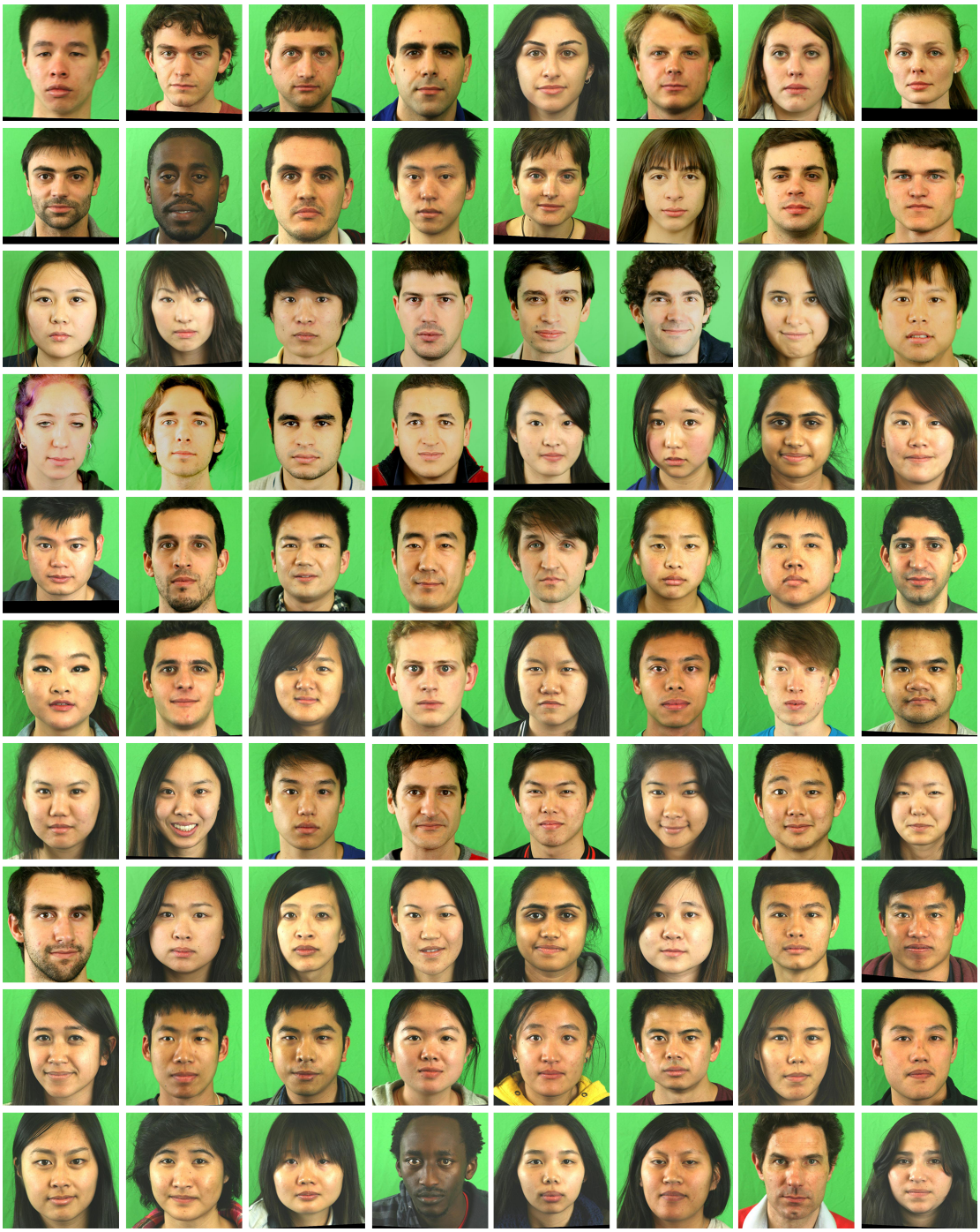


Figure 1.4: *All* 80 *subjects from our dataset.*

a sparse vector x. Lastly we compute the per-subject residuals r_k for all labels k, defined as the norm of the difference between Ax and $A\hat{x}_k$ where $\hat{x}_k$ is x for components that correspond to subject k and 0 elsewhere. The output is the

subject with lowest residual. Since no code is provided, we implemented our own version matching the same success rate on standard datasets claimed by the authors.

SRC actually projects images on a low-dimensional subspace (*e.g.,* $\mathbb{R}^{120}$) both for speed issues and efficiency reasons. This projection can be done in several ways, including downsampling (SRC+DOWNSAMPLE), masking to isolate a part of the face (SRC+MASK) or using "randomfaces" (projection using a random matrix).[4] In the mask version we isolated the right eye and the mouth in order to study the class of algorithms that only rely on local features.

Both EIGENDETECT and SRC work well on simple datasets like the *AT&T Laboratories Cambridge Face Dataset* (respectively 94.38 % and 95.62 %) but they differ on challenging ones like the *Extended Yale Face Database B* [23] (respectively 38.38 % and 90.98 %). Our goal is to show that managing perspective distortion *improves* performance, so the actual performance figure is irrelevant other than for serving as a baseline. Indeed, we will see that both are affected by perspective distortion, especially from short distances.

1.3.2 Experiments

We used the 5 "expression" images as dictionary samples (neutral, smiling, angry, looking left and "joker"). Those photos were shot with a focal of 70 mm (thereafter called the *reference focal*). Then we ran 7 recognition tasks, one for each focal length, over the last 67 % subjects of the dataset (the first 33 % being reserved for face warping modelization). We repeated the experiment for the three algorithms considered. The results are summarized in table 1.2.

Success rate is at most slightly affected for focals close to the reference focal, but dramatically drops with short focals. A wide-angle (10 mm) produces

[4] However this randomness introduces excessive variance between runs and therefore is not used in this work.

Focal length	EIGENDETECT	SRC+DOWNSAMPLE	SRC+MASK
10 mm	56.86 %	82.35 %	45.10 %
17 mm	72.55 %	88.24 %	74.51 %
22 mm	76.47 %	92.16 %	80.39 %
24 mm	94.12 %	98.04 %	86.27 %
34 mm	92.16 %	96.08 %	92.16 %
50 mm	88.24 %	96.08 %	88.24 %
70 mm	88.24 %	100 %	90.20 %

Table 1.2: *Success rate for three face recognition algorithms for each focal length,* 70 mm *being the reference focal length. The learning set is composed of* 5 *images of each individual with different expressions. The success rate is defined as the number of correctly identified subjects over the number of subjects.*

distortions that significantly decrease recognition rate, even for state-of-the-art algorithms (*e.g.,* 45.10 % instead of the nominal 90.20 % for SRC+MASK).

1.4 Learning Perspective Distortion

In this section we describe a method to hallucinate image domain deformations due to changes in frontal distance. In a first step we suppose that the initial focal is known (*e.g.,* from EXIF metadata). Then we solve the problem where the initial focal is unknown. More in general, the method allows extrapolating the orbit spanned by a single data point under a one-parameter family of group transformations. Formally, if the transformation is a function of the image lattice $\phi_{t\to t'} : D \subset \mathbb{R}^2 \to D$ parameterized by *source* $t \in \mathbb{R}$ and *destination* $t' \in \mathbb{R}$ then the orbit of an image I_{t_0} is defined as

$$[I_{t_0}] \triangleq \{I_t \mid t \in \mathbb{R}\} = \{\phi_{t_0\to t}\,(I_{t_0}) \mid t \in \mathbb{R}\} \ . \tag{1.3}$$

In all the following the parameter $f \equiv t$ belongs to the set of focals denoted $\mathbb{F}$ which is a closed bounded interval of $\mathbb{R}$.

1.4.1 Formalization

1.4.1.1 Image Formation

A face is usually represented as coupled shape S and albedo ρ where $S \subset \mathbb{R}^3$ is a simply-connected surface and $\rho : S \to \mathbb{R}^3$. A pixel x in the image is the projection of its corresponding point $p \in S$ onto the image plane and the value of the image at this pixel is given by a function (the illumination) of the albedo at this point p plus some additive noise (see equation 1.4).

$$\begin{cases} I_f(x) = h(\rho(p)) + n(x), & n \sim \mathcal{N}(0, \sigma^2) \\ x = \pi_f(g(p)) \end{cases} \tag{1.4}$$

In our simple setting, illumination $h : \mathbb{R}^3 \to \mathbb{R}^3$ is supposed diffuse and constant (*i.e.,* $h = \mathrm{Id}$). Since the face is aligned (in-plane translation, rotation and scale taken care of by fixing the position of right and left eyes) and supposed frontal (little or no out-of-plane rotation) we can canonize out the vantage point $g \in SE(3)$. The projection is a function of the surface but also of the focal length f:

$$\pi_f(p \in S) = \pi_f\left([x\ y\ z]^\top\right) = f \begin{bmatrix} \frac{x}{z} & \frac{y}{z} \end{bmatrix}^\top \tag{1.5}$$

where $z = 0$ is the image plane and z is the distance from p to the image plane. The above considerations allow us to simplify equation 1.4 as follows:

$$I_f(x) = I_{f_0}(w_f(x)), \quad x \in D \tag{1.6}$$

where w_f can be viewed as a *warp* from the image lattice D to itself and f_0 is the reference focal. Note that we have $w_f = \phi_{f_0 \to f}$.

Our goal in this section is thus: *given an image of a face $I : D \to \mathbb{R}^3$, corre-*

sponding to a known or unknown focal f_0, find the set of functions $\{w_f : D \to D \mid f \in \mathbb{F}\}$ modeling perspective distortions.

1.4.1.2 Representation of a Face

The warps w_f depend on the shape of the face but not its albedo. For this reason we can discard the albedo information in our representation of a face. We only wish to represent the shape S. Explicit reconstruction could be employed here, even though the absence of viewpoint variability makes it entirely dependent on priors [7, 18, 30]. To avoid that, and for simplicity, we consider the warp a function of the hidden variable S, represented by a few sample points within. Active appearance models (AAM) [13, 12] can then be employed to fit a model on unseen faces. The points fitted via AAM are thereafter called *landmarks*. In practice we used $N = 64$ landmarks, delineating the eyebrows, the eyes, the nose and nostrils, the mouth and the outline of the face (see Fig. 1.5). As customary, we remove the affine component (the mean) but rather than doing so across the entire dataset, we index the mean by focal length:

$$I_f \equiv X_f - \overline{X_f} = \Delta X_f \in \mathbb{R}^{2N} \tag{1.7}$$

where $X = [x_{1x}\ x_{1y}\ \dots\ x_{Nx}\ x_{Ny}]^\top$ and $\overline{X_f}$ is the average face at focal f.

1.4.1.3 Assumptions on Warps

To go further we need to make basic assumptions of regularity on the warps $\{w_f\} \equiv \{\phi_{f_0 \to f}\}$. Namely we assume that, as a function of ΔX, a warp is a diffeomorphism[5] from $\mathbb{R}^{2N}$ to itself. We can then write the linear approximation:

$$\Delta X_f = \phi_{f_0 \to f}(\Delta X_{f_0}) = \phi_{f_0 \to f}(0) + D\phi_{f_0 \to f}(0)^\top \Delta X_{f_0} + \mathcal{O}(\|\Delta X_{f_0}\|^2) \tag{1.8}$$

[5]In reality it is sufficient for the warp to be differentiable on $\mathbb{R}^{2N}$ but we naturally expect warp $\phi_{f_1 \to f_2}$ to be invertible and its inverse to be $\phi_{f_2 \to f_1}$.

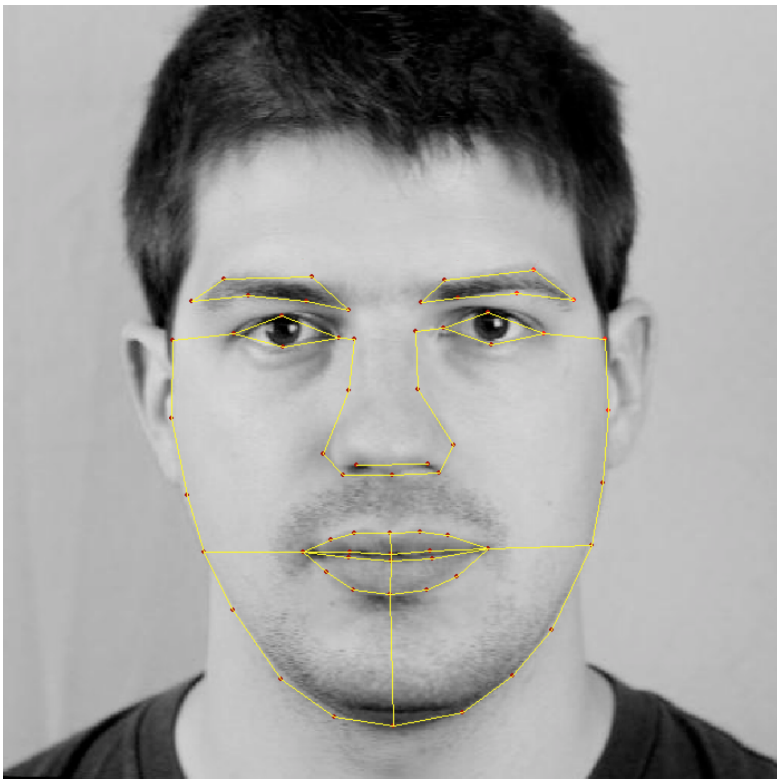


Figure 1.5: *64 landmarks fitted via active appearance models.*

where $\|\cdot\|$ is some norm on $\mathbb{R}^{2N}$. This approximation is valid so long as faces are "close" to the average face, which should be the case in practice.

By letting $b_{f_0 \to f} \triangleq \phi_{f_0 \to f}(0)$ and $A_{f_0 \to f} \triangleq D\phi_{f_0 \to f}(0)^\top$ we obtain the following affine approximation:

$$\Delta X_f \approx A_{f_0 \to f} \Delta X_{f_0} + b_{f_0 \to f} \ . \tag{1.9}$$

1.4.2 Learning the Model

1.4.2.1 Face Warping as a Quadratic Minimization Program

Eq. (1.9) gives a convenient way to warp any face taken at focal f_0 to its counterpart at focal f. Unfortunately we cannot compute $A_{f_0 \to f}$ and $b_{f_0 \to f}$ because they depend on the unknown function $\phi_{f_0 \to f}$. However, since they do not depend on the face itself, we wish to learn them using a sufficient number of samples.

To that end we want to minimize the quantity

$$\sum_{i=1}^{n_T} \|A_{f_0 \to f} \Delta X^i_{f_0} + b_{f_0 \to f} - \Delta X^i_f\|^2$$

with n_T being the number of training samples and the norm being the Euclidian norm. However this problem is typically underconstrained because there are $2N(2N+1)$ free variables and each subject contributes $2N$ constraints. To avoid overfitting it is necessary to regularize the elements of A and b. We naturally want to encourage a matrix A close to the identity and b close to zero, because this corresponds to ϕ being the identity, and even though a face undergoes important changes that motivate this work, it should stay close to itself through perspective distortion. Note that we need to learn matrix A and vector b for each pair of parameters (f_1, f_2). We thus propose to solve the following quadratic minimization program:

$$A_{f_1 \to f_2},\ b_{f_1 \to f_2} = \underset{A,\ b}{\operatorname{argmin}} \sum_{i=1}^{n_T} \|A \Delta X^i_{f_1} + b - \Delta X^i_{f_2}\|^2 + \lambda \|A - I\|^2 + \mu \|b\|^2 \ . \quad (1.10)$$

Lagrange multipliers λ and μ are selected via grid search, using 67 % of the training data for learning and 33 % for cross-validation. Once λ and μ are selected, we learn A and b again over the entire training data. In practice we used $\lambda = 10^5$ and $\mu = 10^{-2}$.

1.4.2.2 Interpolation Between Focals

The quadratic program (1.10) enables the transformation from any parameter f_1 to any other parameter f_2 for which we have data. Obviously data is only collected at sampled focal lengths $\{f_1 = \inf \mathbb{F} < f_2 < \ldots < f_p = \sup \mathbb{F}\} \subset \mathbb{F}$.

Provided that the sampling is fine enough, and that the sensitivity of $A_{f_1 \to f_2}$ and $b_{f_1 \to f_2}$ to source focal f_1 and destination focal f_2 is smooth, bilinear interpolation can be used to approximate $A_{f \to f'}$ and $b_{f \to f'}$ for any $(f, f') \in \mathbb{F}^2$. Should the sampling be too coarse, one can resort to better interpolations, such as cubic

spline interpolation.

1.4.3 When the Source Focal is Unknown

So far we have seen how to hallucinate an image $I_{f'}$ of a face at any focal f' given the image I_f, provided we know the source focal f. In typical applications we may not know this focal and therefore need to infer it. Formally, we seek a function $d : \mathbb{R}^{2N} \to \mathbb{F}$ such that $|d(X_f) - f| < \eta$ with high probability for some tolerance $\eta \in \mathbb{R}^+$. The tolerance depends on the sensitivity of A and b to the source and destination focals. Indeed mistaking f_1 for f_2 may be tolerable if $A_{f_1 \to f'} \approx A_{f_2 \to f'}$ and $b_{f_1 \to f'} \approx b_{f_2 \to f'}$.

Several approaches can be considered. Provided the focal space is sufficiently densely sampled and the data is clustered by focal length (which seems suggested by [14]), a nearest-neighbour search can be attempted. However our data did not prove clustered enough and a reliable estimate of the focal length could not be obtained. Linear SVM approaches also proved insufficient. To deal with the non-linearity of the data, we instead trained a neural network with one hidden layer containing 4 nodes.[6] This leads to an RMS error of 13.17 mm on the testing data, which is surprisingly accurate given that beyond a threshold, even a trained human cannot give such an estimate.

1.4.4 Comparison of Orbits for Perspective Distortion Mitigation

We saw in section 1.3 that both a basic and a state-of-the-art algorithms occasionally fail when shown faces taken from an unusual standpoint. To address this issue, based on the interpretation of perspective distortions generating orbits in the space of images, we propose to *compare orbits, rather than data points.* This idea is common in applications where the data is acted upon by a group, and

[6] More complex architectures, *e.g.*, three hidden layers with 32, 16 and 8 nodes, also gave good results but took longer to train.

one wishes to perform comparisons in orbit space, or equivalently in the quotient space [16].

A distance between orbits can be defined by "max-out," that is by minimizing over all possible group actions: If (I_1, I_2) are two images that we want to compare, and $[I] = \{\phi I\}$ is the orbit generated by I under the action of the group ϕ, then we can define a distance between orbits via

$$d([I_1], [I_2]) = \min_{\phi_1,\ \phi_2} d_0(\phi_1 I_1, \phi_2 I_2) \tag{1.11}$$

where d_0 is a base distance in the data space. This however requires solving an optimization problem at decision time.

Alternatively, one can exploit the fact that each orbit is an equivalence class, which can be represented by any of its elements. So long as it is possible to select a unique "canonical element," one can simply compare canonical elements (eq. 1.12). This does not entail any optimization. This is the approach we take, with the canonical element being the mapping of an image to reference focal length. This can be seen as a pre-processing step, after which the warped image can be fed to any standard face recognition system. In practice the focal estimation takes 0.3 s and the actual warp 2.0 s for a 256×256 frame on consumer hardware.

$$d([I_1], [I_2]) = d_0(\hat{I}_1, \hat{I}_2)\ . \tag{1.12}$$

1.5 Experimental Assessment

1.5.1 Qualitative Results

As suggested in [9], extrapolation of perspective distortions can be applied directly to image editing. The warp described in section 1.4 can easily be extrapolated by letting the source and destination landmarks be respectively control points and their images, and using a thin-plate spline [40] to obtain a dense warp. We implemented this solution in a Matlab GUI application (see Fig. 1.6) that allows

warping an input image into its hallucinated version at any focal in the range [10 mm ; 70 mm].

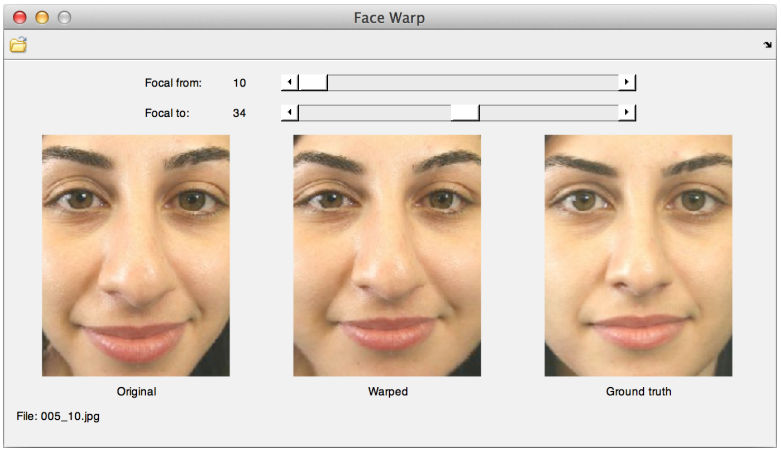


Figure 1.6: *Face warping GUI. The handles allow to correct the source focal and to control the destination focal. The first panel is the input, the middle panel is the warped face image. When a ground truth face is available it is displayed on the third panel.*

In Fig. 1.7 we show an application to un-distortion of videoconference streams. In this proof-of-concept demonstration, it is assumed that a detector/tracker yields a smooth estimate of the location of the eyes. Landmarks are fitted using AAMs. The distance to the screen (and hence the "focal") is simply estimated using the distance between the eyes, and the face is then warped to the desired viewing distance. This application enables mitigating the undesirable effects of the typical optics employed in forward-looking cameras on mobile devices and tablets.

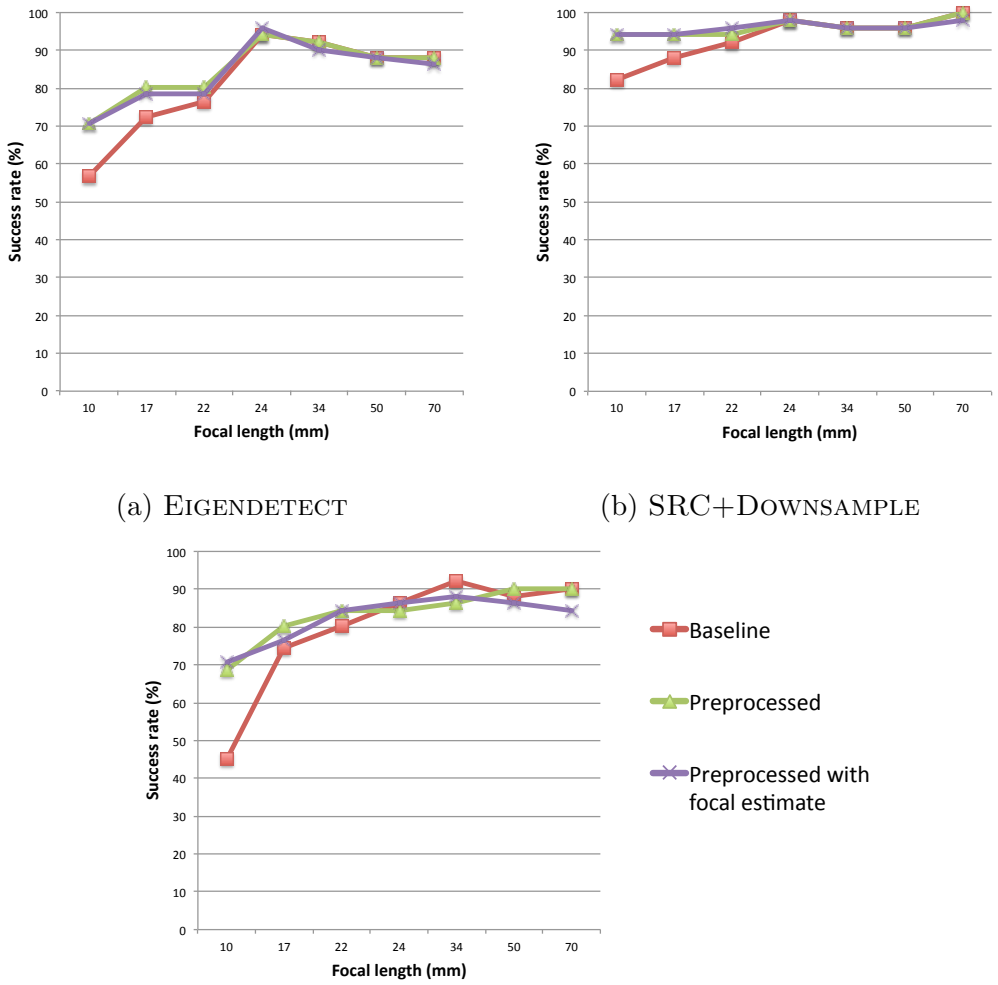
1.5.2 Managing Perspective Distortion in Face Recognition and Validation

To illustrate the mitigation of perspective distortion in face recognition, we conducted two experiments. In the first one we pre-processed the images by warping them from their true focal length (known in our dataset) to the reference focal length. In the second experiment we do not suppose the focal length known and instead estimate it as explained in section 1.4.3. Fig. 1.8 summarizes improve-

Figure 1.7: *Application of face unwarping to videoconference streams. (Top) original frames 1, 115 and 403. (Bottom) unwarped versions. Since the focal is known (*30 mm *in 35-mm equivalent), the image uncropped and the face frontal, the distance from the subject is estimated using the distance between the eyes, and is then converted to an estimated "focal" using formula 1.1. The face is then warped to* $f = 63\,\text{mm}$ *which corresponds to a viewing distance of* 50 cm*. Not counting the detection of the eyes and the fitting of AAMs, the application runs at an average rate of* 4.1 s *per* 432×270 *frame, time mostly spent for resampling in Matlab's affine transformation function and for thin-plate spline interpolation. The warp is only applied to the face area and smoothly vanishes on its edges by fixing control points on them before using thin-plate spline interpolation.*

ment of success rate by comparing the three experiments: without pre-processing, with pre-processing when focal is known and with pre-processing when focal is estimated.

The most noticeable results appear for the extreme focal length $f = 10\,\text{mm}$. Because of huge distortions happening at this distance, algorithms perform at their worst. Our method compensates for these distortions and allows to achieve higher success rates. Above a certain threshold, perspective distortion becomes negligible and, as expected, our method only produces negligible random fluctuations. Note that our focal estimate is reliable enough to give results that are almost as good as when the focal is known.



(a) EIGENDETECT

(b) SRC+DOWNSAMPLE

(c) SRC+MASK

Figure 1.8: *Success rate of face recognition algorithms with and without pre-processing.*

In a final proof-of-concept experiment (Fig. 1.9), we take the opposite approach where the focal length is known and controlled by the system. Because the warp induced by perspective distortion is shape-dependent, it is possible to capture multiple images at different focal lengths, rescale them, and then test the compatibility of the resulting deformation with the shape of the underlying scene. This would allow validation of the identity of a face in a way that a single-image based recognition system cannot do (even the best face recognition system based on a single view cannot discriminate between an image of a person and an image of

an image of a person). Practically, the application estimates the warping between images taken from different distances and validates or invalidates the output of the underlying single-view recognition system.

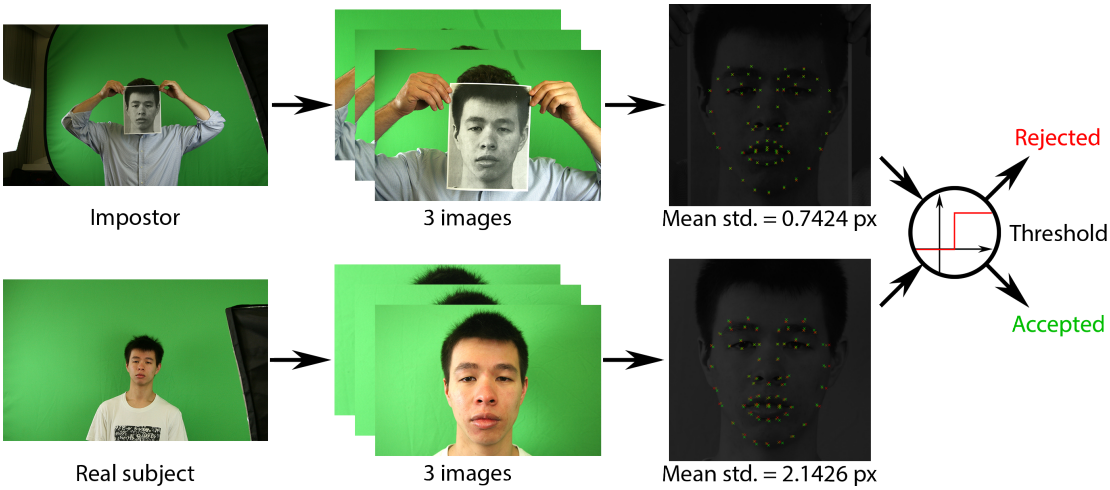


Figure 1.9: *Multi-view validation of an underlying single-view recognition system. In this scenario, an impostor uses a photograph pretending to be some authorized subject. The camera controls its own viewing distance and focal length and triggers the shutter from different distances. After scaling and processing, the standard deviation of each landmark trail, averaged over all landmarks, can be thresholded to unveil the supercherie. Single-view approaches would inevitably fail here.*

1.6 Conclusion

We study the effects of varying distance in frontal face images. While such variations have significant perceptual impact, and have been exploited by artists for centuries [32], an explicit modeling and a quantitative assessment of this phenomenon and its impact on face recognition have not been attempted before.

It is also possible to employ the system for synthesis purposes to modify the appearance of a photograph or a video as if it was taken from a different distance, thereby manipulating a person's perceived qualities.

The methodology developed could be extended to other families of one-parameter transformations, assuming that they yield differentiable and differentially-invertible warps, which is not the case in the presence, for instance, of occlusions. This includes self-occlusions from out-of-plane rotation.

CHAPTER 2

An Object Tracking Library

Object tracking is the task of locating a specified object in a frame sequence given its initial position and scale. It differs from object detection or recognition, that rather aim to decide whether an object exists in a given frame, possibly where it is and which specific object it is. Tracking can be done at three levels [43]: feature trackers model the object of interest by a single point (*e.g.,* KLT tracker [36]), kernel trackers use a simple geometric representation like a rectangle or an ellipsis around the object of interest, and silhouette trackers model the object at the pixel level, either by explicitly marking the points that belong to the object or only its contour. Object tracking plays a major part in many computer vision applications, ranging from automated videosurveillance and traffic monitoring to human-computer interaction, motion-based recognition and autonomous vehicle navigation [43]. However there is not, to date, a ready-to-use simple and robust framework for object tracking. As a building block for higher-level applications, an *ad hoc* tracker is often engineered for a given task. A reason to that is that there is no *generic tracker* that deals with any situation optimally. Some methods perform state-of-the-art in presence of a certain difficulty (*e.g.,* changes in illumination) but can be unsuitable in other circumstances, such as occlusions, large deformations or change of viewpoint. Moreover if the tracking task is easy or can leverage some knowledge, a simpler but taylored method may be preferred to an allegedly generic method that fails in that particular context.

Our purpose is thus to build a versatile and modular online object tracking

open-source cross-platform library, with the hope of coming in helpful to the computer vision community. The objectives we fix to ourselves are twofold: the library should be as easy to install and use as possible, and easily extensible. An obvious third goal is to implement a sufficient number of trackers to span a variety of applications. This chapter should be considered a basic documentation of how to use and extend our object tracking library.

In section 2.1 we describe *Tracklib*, our object tracking library written in C++, and in section 2.2 we present *Multitrack*, a graphical front-end for tracking purposes that uses and demonstrates Tracklib. Section 2.3 concludes with future directions for the project.

2.1 Tracklib, a C++ Library for Object Tracking

2.1.1 Design and Principles Behind Tracklib

Of the three levels of object tracking aforementioned, we only consider kernel trackers, *i.e.*, object trackers that use a simple geometric representation of the object like a rectangle or a ellipse. This representation is powerful enough to build a robust model of an object, while staying simpler to handle than silhouette representations.

In Tracklib, the eventual representation of an object is a rectangle, namely the bounding box of the ground-truth object. For now only straight rectangles can be used, but an upcoming release will add the option to use rotated rectangles or skewed rectangles, to allow for a tighter bounding box. However this option adds complexity and in many cases straight rectangles are a sufficient structure.

The design of Tracklib is meant to be highly modular. To this end we define an *online tracker* as a system that consists of several components:

- A *detector*, which can be more or less complex, and potentially composed

itself of simpler detectors, aims to detect the object or interest and outputs its bounding box with possibly a measure of confidence.

- An optional *filter* smoothes the preceding output and makes the general system more robust to drifts and jerks.

- An optional *background subtraction algorithm* may considerably improve the results if the background is known to be static or easy to model (*e.g.,* static camera, or panning camera with still background). The background subtraction algorithm may run online or offline, in which case a precomputed estimate of the background is given to the tracker.

- An optional *preprocessing module* takes the raw image as an input and performs low-level tasks, such as smoothing, adjusting contrast or simply cropping.

- An optional *postprocessing module* takes the candidate output of the tracker and can perform any kind of processing to it. For example it can precisely adjust the bounding box given some external knowledge.

Fig. 2.1 shows a block diagram of a generic object tracker. Here, "generic" does not mean that it applies to any situation, but that it aspires to be maximally customizable. The code, written in C++, directly implements this architecture via the base class `Tracker` and the abstract classes `Detector`, `Filter` and `BackgroundSubtractor`.

The library is object-oriented, compiles with C++11, depends on OpenCV 2.x, and uses Doxygen to generate automatic documentation.

2.1.2 A Few Algorithms

Tracklib comes with built-in algorithms for each of the components mentioned above. The number of built-in algorithms is still low at this early stage of de-

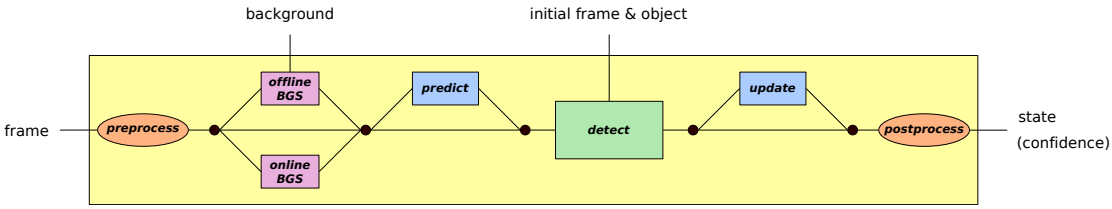


Figure 2.1: *Block diagram of a generic online object tracker. Diverging nodes figure choices. Blocks represent C++ classes (pink: background subtractor, blue: filter, green: detector) and orange ellipses are virtual methods. Offline background subtractor should be initialized with a background estimate (or a video stream to estimate it) and detector with initial image and object state.*

velopment but is meant to be constantly growing, to allow the user for more ready-to-use alternatives.

2.1.2.1 Detectors

Although object tracking is too vast a field to be reviewed here, we will go through a few examples of object tracking algorithms that are implemented or being implemented. For a thorough survey of algorithm, please refer to [43].

Template Matching Template matching consists in running a sliding window over the image and finding the location that best matches the initial template. This simple method is computationally expensive and its basic version does not allow for change of scale, and is not robust to challenging situations involving occlusions or large deformations. Changes in illumination is moderately handled by comparing colors in HSV space (dropping the vibrance component and sometimes saturation) instead of RGB space, as customary in object tracking. The basic algorithm is implemented in the class `TemplateMatchingDetector`. This algorithm is typically parallelizable and a GPU implementation is provided in `TemplateMatchingDetectorGpu`. These leverage OpenCV built-in implementations. A number of authors extended basic template matching and showed good

results to handle challenging situations: [27] uses a Kalman filter to handle not only the position as customary but also changes in illumination and object orientation, [6] uses view-based representations ("eigentracking") to track articulated objects and [20] aims to handle occlusions. Lastly, some methods extend basic template matching to allow for more complex deformations than only translations: [21] devises a template matching method invariant to rotation, scale and translation, and [22] allows any 2D affine transformation.

Meanshift Meanshift is a fast algorithm that uses a hue histogram as the object representation. This makes it very efficient when it comes to tracking an object easily detachable from the background by its color, even in presence of arbitrarily large deformations. The initial template is projected to a color histogram, which is then backprojected onto the new image. Pixels with high values are more likely to be part of the object than those with low values. The classical meanshift algorithm is then applied: given the initial estimate, find its center of gravity (weighted by the votes of each pixel inside), center the new estimate on it, and repeat until convergence. This, once again, only allows translations, but a variant called Camshift [8] also enables scale. Both methods are implemented in the class `MeanshiftDetector` (use `TL_CAMSHIFT` to enable Camshift).

MILTrack MILTrack [4], along with most methods that follow, is an example of "tracking by detection" algorithm. This class of algorithms has proven very robust over the past decade. MILTrack learns an adaptive appearance model of the object that discrimates, over the pixels inside the bounding boxes, which belong to the object and which belong to the background. An open-source implementation is provided by the authors, which has been plugged into the Tracklib framework. A related approach is described in [41]. While MILTrack uses a sliding-window detection approach, [41] employs a dynamical model to increase robustness. An implementation of this method is in progress.

Online Boosting Online boosting (*e.g.,* [3]) consists in combining weak detectors (in this case pixel-level detectors) to obtain a detector with improved accuracy. [15] also tries to discriminate foreground pixels from background pixels via online boosting. An open-source implementation has also been integrated into Tracklib.

PROST PROST ([34]) is an interesting example of modular design. Its main argument is that by combining three detectors with different qualities, one can obtain a more accurate and robust tracker. The paper explains that a clever cascade of detectors performs way better than any of those taken separately, by successively running template matching for stability, an optical flow meanshift algorithm for plasticity, and a random forest approach. A set of precedence rules determines how to give a final output from the three individual outputs.

2.1.2.2 Filters

Kalman Filter Kalman filters are incredibly efficient and popular in various fields, and tracking is no exception to the rule. It has been used in most recent approaches to enforce robustness in tracking. A generic Kalman filter (however without control-input) is implemented in the class `KalmanFilter`, along with typical specializations for use in tracking. It defaults to a Kalman filter applied to a dynamical system, where the state is given by position, scale, and their velocities, the transition model is the dynamical transition model, and the observation model is a mask for position and scale.

2.1.2.3 Background Subtraction

Online Subtraction Many algorithms exist for online background subtraction. [26] surveys background subtraction techniques from a decade ago, and [38] is an open-source background subtraction library that currently implements 34 algo-

rithms that include mean or variance-based methods, statistical methods using one gaussian or a mixture of gaussians, statistical methods using color or texture features, non-parametric methods, and various other methods.

Offline Subtraction In some scenarios the background may be known or modeled in advance and simply fed to the tracker. For example, if one has footage taken by a static surveillance camera, but is given the task to track a scene online. In this case offline subtraction often works better than online subtraction. To this end an offline background subtractor can be obtained via the class `OfflineBackgroundSubtractor`. Currently the only implemented method consists in computing the mode of each pixel over the video fed offline. At runtime those pixels that match the estimated background are removed from the image before being processed by the detector/filter.

2.1.3 How to Use - Easiness of Use

2.1.3.1 Installation

To "install" Tracklib one simply needs to add the path to Tracklib to the include paths. No dynamic libraries need to be generated. However one can also compile Tracklib as a standalone project by running `cmake` then `make` at the root directory.

2.1.3.2 Usage

To use Tracklib one just has to include the root file `tracklib.h`. The entire library is nested under the namespace `tl` to avoid conflicts.

Every detector derives from the abstract class `Detector` and can be instantiated with the initial frame (an OpenCV Mat image that has to be RGB or grayscale, and 8-bit unsigned integer or 32-bit floating point) and the initial object (figured by an OpenCV Rect structure). Detector parameters can be customized

atomically via methods `set_XXX()` where `XXX` is the name of the parameter. Many parameters accept values that are (non strongly typed) enums in the `tl` namespace, capitalized and prefixed by `TL_`, in an OpenCV fashion. A detector can be run by successively calling its `NextFrame()` method to feed him the new frame, and its `Detect()` method to trigger the computation. The computed object state can be retrieved via the `state()` accessor.

A filter derives from the abstract class `Filter` and exposes two methods: `Predict()` which outputs an estimate of the object state that is meant to be fed to the detector as a starting position, and `Update()` that updates the filter and corrects the ouput of the detector.

In simple cases one can only instantiate and run a detector. However in most cases filtering is desired. In that case it is better to implement a `Tracker` and specify its detector and filter via `set_detector()` and `set_filter()`. The main method to call is `Track()` that essentially runs the pipeline described in Fig. 2.1. The output can then be retrieved via the accessor `state()` as for a detector. The method `set_background_subtractor()` allows to use background subtraction. To add a preprocessing or postprocessing step, one must inherit the class Tracklib and redefine the virtual methods `preprocess()` or `postprocess()`.

Listing 2.1 shows a sample `main.cpp` file using Tracklib. Note that safety checks were removed to make the code more concise.

Listing 2.1: *Sample program using Tracklib.*

```
#include <cstdlib>
#include <string>
#include <opencv2/core/core.hpp>
#include "tracklib/tracklib.h"

using namespace cv;
using namespace std;
using namespace tl;

int main(int argc, char **argv) {
  // Read command-line arguments.
  string path = argv[1];
  Rect state;
  state.x = atof(argv[2]);
  state.y = atof(argv[3]);
  state.width = atof(argv[4]);
  state.height = atof(argv[5]);

  // Open video and read first frame.
  VideoCapture cap(path);
  Mat frame;
  cap >> frame;

  // Detector.
  MeanshiftDetector detector(frame, state);
  detector.set_variant(TL_CAMSHIFT);

  // Filter.
  KalmanFilter filter(state);

  // Tracker.
  Tracker tracker;
  tracker.set_detector(&detector);
  tracker.set_filter(&filter);

  while (true) {
    // Display.
    Mat image = frame.clone();
    rectangle(image, state, Scalar(0, 0, 255), 3);
    imshow("Tracking", image);

    // Read next frame.
    cap >> frame;
    if (!frame.data) break;

    // Track object.
    tracker.Track(frame);
    state = tracker.state();

    // Handle events.
    int c = waitKey(10);
    if (c == 27 /* ESC */ || c == 'q') break;
  }

  return 0;
}
```

2.1.4 How to Extend - Modularity

Extending Tracklib is very easy. One can simply add new detectors under the directory `tl_detectors` (they must redefine the `Detect()` method), new filters under `tl_filters` (they must redefine the `Predict()` and `Update()` methods), and so on. Special mention should be made to GPU implementations which should be included in `tl_gpu` and be named *`ClassName`*`Gpu`. Indeed this directory can be excluded from the compilation on request (*e.g.,* if CUDA is not installed on the machine[1]). Once implemented, the new class should be added in the main include file `tracklib.h`.

2.2 Multitrack, a Graphical Front-End for Tracklib

Experiments on object tracking are generally difficult to set up, computationally expensive, and time consuming. On top of that many algorithms are not real-time, and quantification of results is tedious. With the idea of easing research experiments, we developed a software called *Multitrack*, built on top on Tracklib, that enables easy manipulation of videos, parametrization of algorithms and analysis of results.

In a nutshell its features include:

- Support for video files (those read by OpenCV) with possibility to restrict the interest range in terms of frames or seconds,
- Support for sequence of frames (readable by OpenCV) with control over the number of frames per second,
- Playback of the source video with multimedia controls (play/pause, slider for navigation, playback speed),

[1]To disable CUDA one can simply comment the line `find_package(CUDA)` in the file `CMakeLists.txt`.

- Possibility to stack multiple algorithms, run them once and save their outputs, visualize them all or disable some of them,
- Support for all the detectors in Tracklib, parametrizable within a UI,
- Support for online or offline background subtraction,
- Support for the Kalman filter,
- Possibility to easily select the object to track *in-app*, and to choose the starting frame (which does not have to be the first one in the sequence),
- Functionality to export tracking results into various CSV formats,
- Ability to save the entire work into a "Multitrack project" that can be reopened later.

Multitrack is written in C++ and requires the framework Qt 5.x.[2] Note that Multitrack was written on top of a former version of Tracklib and needs to be adapted to its new modular design.

Fig. 2.2 shows the opening screen that gives the traditional choice between creating a new project, opening an existing one or choosing between the list of recent projects. A new project (Fig. 2.3) can link either to a video file or to a sequence of frames.

The main window is displayed in Fig. 2.4. The top section is for playback, the bottom left for customizing and running the trackers (Fig. 2.5), and the bottom right for previewing a tracker that is being run (Fig. 2.6). Lastly, Fig. 2.7 shows how to export the results to a customizable CSV formatted file.

The ergonomy of the software partly comes from the use of separate threads. Currently, there is:

[2] `http://qt-project.org`

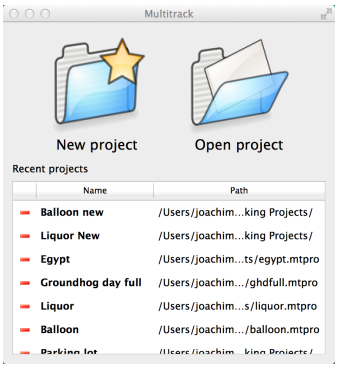


Figure 2.2: *Opening screen of Multitrack.*

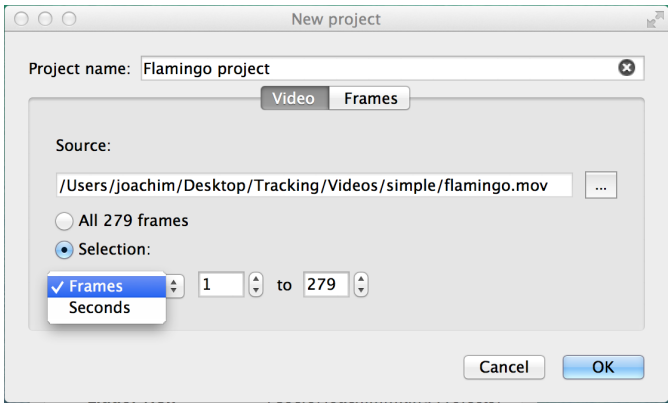


Figure 2.3: *Creating a new project.*

- One thread (main Qt thread) to handle UI events,
- One thread for playback, as customary in multimedia softwares,
- One thread to run a tracker and display its progression in the preview section.

Currently only one tracker can be run at once (but playback and other UI operations can be done concurrently). In a future version, one thread will be allocated per tracking task, which will allow to run several trackers at once. Other future features include allowing to easily annotate ground truth and evaluate algorithms when a ground truth is available.

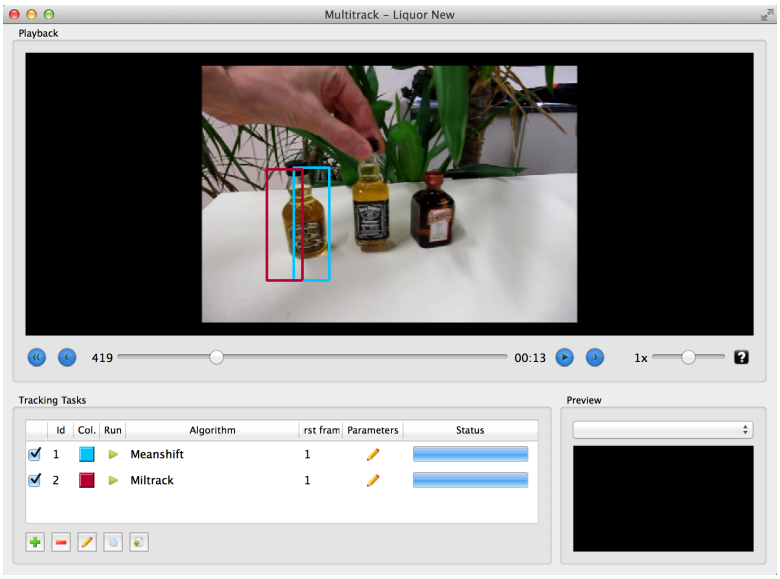


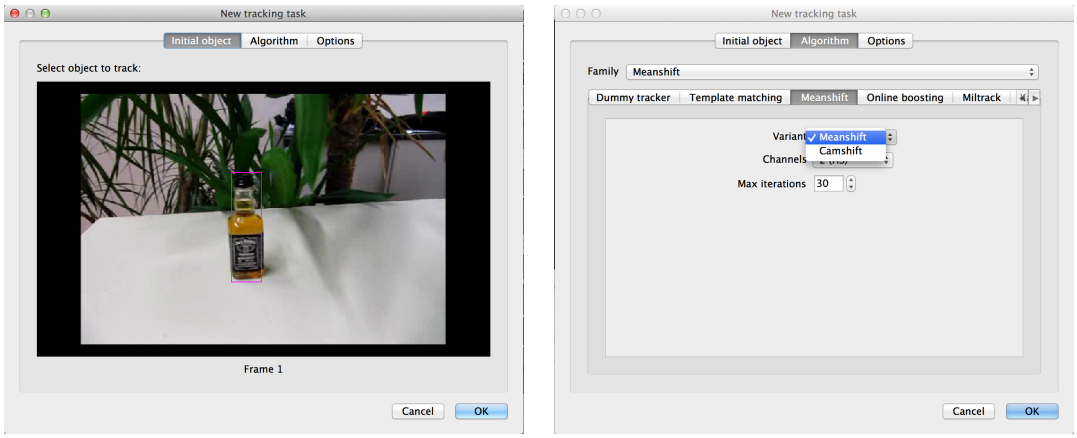
Figure 2.4: *Playback of the video. Tracker outputs can be shown or hidden by ticking the corresponding checkboxes. Colors are randomly assigned but can also be changed if needed.*

2.3 Open-Source Deployment

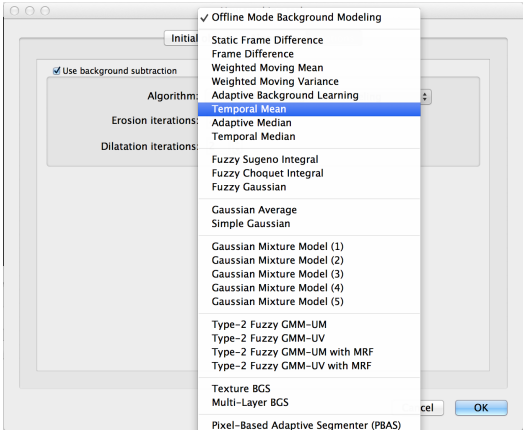
Tracklib has been designed to be potentially usable by many. The use of C++ is in general already widely spread, and in particular in the vision community with the notable preeminence of the OpenCV library. It uses CMake[3] to make the compilation easy and cross-platform. Lastly it uses Doxygen[4] to automate the documentation generation, which is crucial to help spread a library. To generate the documentation, simply run `doxygen doxyfile` at the root directory. Multitrack only adds a dependency to Qt, which is also cross-platorm.

To help spread the library and ease future development, the project has been moved to GitHub[5] under the MIT License. At the time of writing, the background subtractors, the online boosting and MILTrack detectors have not been moved yet, but this is the page where the most up-to-date material should be located.

[3] `http://www.cmake.org`
[4] `http://www.doxygen.org`
[5] `https://github.com/joachimvalente/tracklib`



(a) *Selecting the initial object.* (b) *Configuring the algorithm.*



(c) *Adding background subtraction.*

Figure 2.5: *Configuration of a new "tracking task."*

Additionally, Multitrack will also be moved to the arborescence as a demo - and utility - software.

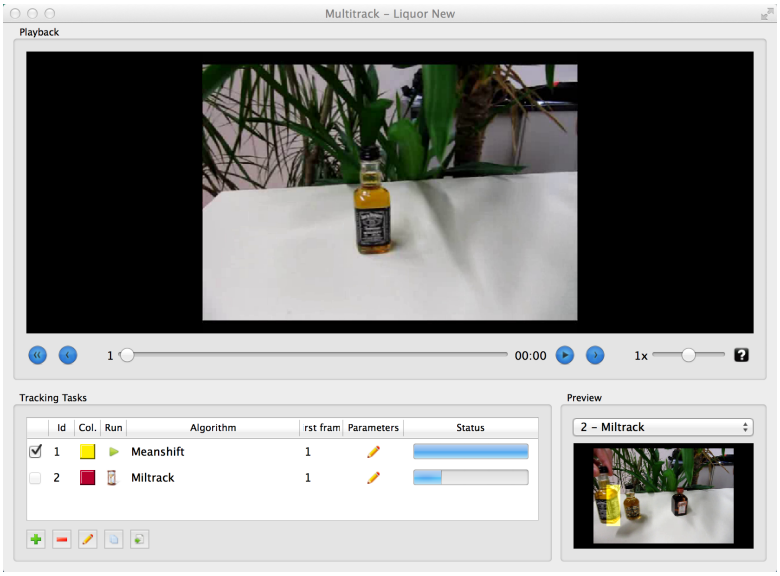


Figure 2.6: *Running a tracking task. The current frame and object state are displayed in the preview section.*

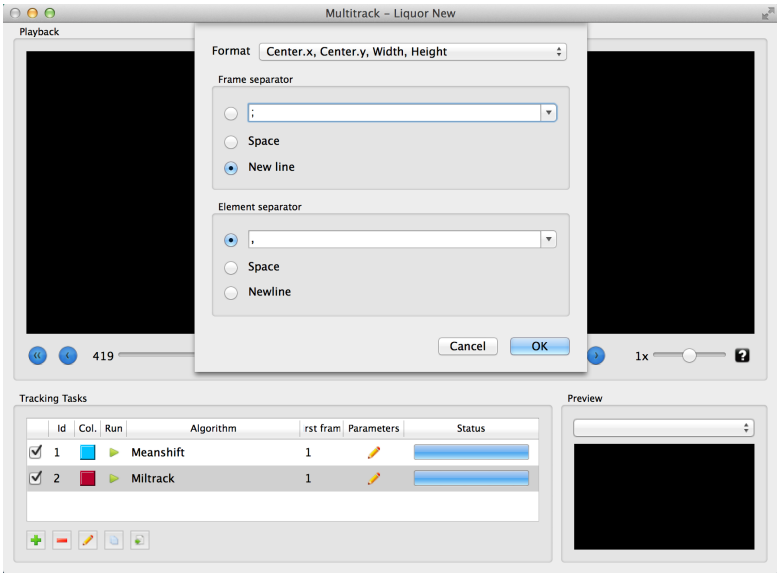


Figure 2.7: *Exporting the results of a tracker to a CSV file.*

References

[1] Andrea F Abate, Michele Nappi, Daniel Riccio, and Gabriele Sabatino. 2d and 3d face recognition: A survey. *Pattern Recognition Letters*, 28(14):1885–1906, 2007.

[2] Yael Adini, Yael Moses, and Shimon Ullman. Face recognition: The problem of compensating for changes in illumination direction. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(7):721–732, 1997.

[3] Shai Avidan. Ensemble tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(2):261–271, 2007.

[4] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. Visual tracking with online multiple instance learning. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 983–990. IEEE, 2009.

[5] Angela Faris Belt. *The Elements of Photography: Understanding and Creating Sophisticated Images*. 2011.

[6] Michael J Black and Allan D Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision*, 26(1):63–84, 1998.

[7] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 187–194. ACM Press/Addison-Wesley Publishing Co., 1999.

[8] Gary R Bradski. Computer vision face tracking for use in a perceptual user interface. 1998.

[9] Ronnie Bryan, Pietro Perona, and Ralph Adolphs. Perspective distortion from interpersonal distance is an implicit visual cue for social judgments of faces. *PloS one*, 7(9):e45301, 2012.

[10] Xiujuan Chai, Shiguang Shan, Xilin Chen, and Wen Gao. Locally linear regression for pose-invariant face recognition. *Image Processing, IEEE Transactions on*, 16(7):1716–1725, 2007.

[11] Emily A Cooper, Elise A Piazza, and Martin S Banks. The perceptual basis of common photographic practice. *Journal of vision*, 12(5), 2012.

[12] Timothy F Cootes, Gareth J Edwards, Christopher J Taylor, et al. Active appearance models. *IEEE Transactions on pattern analysis and machine intelligence*, 23(6):681–685, 2001.

[13] Gareth J Edwards, Timothy F Cootes, and Christopher J Taylor. Face recognition using active appearance models. In *Computer VisionECCV98*, pages 581–595. Springer, 1998.

[14] Arturo Flores, Eric Christiansen, David Kriegman, and Serge Belongie. Camera distance from face images. In *Advances in Visual Computing*, pages 513–522. Springer, 2013.

[15] Helmut Grabner, Michael Grabner, and Horst Bischof. Real-time tracking via on-line boosting. In *BMVC*, volume 1, page 6, 2006.

[16] Ulf Grenander. *Elements of pattern theory*. JHU Press, 1996.

[17] Ralph Gross and Vladimir Brajovic. An image preprocessing algorithm for illumination invariant face recognition. In *Audio-and Video-Based Biometric Person Authentication*, pages 10–18. Springer, 2003.

[18] Yuxiao Hu, Dalong Jiang, Shuicheng Yan, Lei Zhang, and Hongjiang Zhang. Automatic 3d reconstruction for face recognition. In *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*, pages 843–848. IEEE, 2004.

[19] Fu Jie Huang, Zhihua Zhou, Hong-Jiang Zhang, and Tsuhan Chen. Pose invariant face recognition. In *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on*, pages 245–250. IEEE, 2000.

[20] Frédéric Jurie, Michel Dhome, et al. Real time robust template matching. In *BMVC*, pages 1–10, 2002.

[21] Hae Yong Kim and Sidnei Alves De Araújo. Grayscale template-matching invariant to rotation, scale, translation, brightness and contrast. In *Advances in Image and Video Technology*, pages 100–113. Springer, 2007.

[22] Simon Korman, Daniel Reichman, Gilad Tsur, and Shai Avidan. Fast-match: Fast affine template matching. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2331–2338. IEEE, 2013.

[23] Kuang-Chih Lee, Jeffrey Ho, and David Kriegman. Acquiring linear subspaces for face recognition under variable lighting. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(5):684–698, 2005.

[24] Haibin Ling, Stefano Soatto, Narayanan Ramanathan, and David W Jacobs. A study of face recognition as people age. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[25] Aleix M Martinez. The ar face database. *CVC Technical Report*, 24, 1998.

[26] Alan M McIvor. Background subtraction techniques. *Proc. of Image and Vision Computing*, 1(3):155–163, 2000.

[27] Hieu Tat Nguyen, Marcel Worring, and Rein Van Den Boomgaard. Occlusion robust adaptive template tracking. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 678–683. IEEE, 2001.

[28] M. M. Nordstrøm, M. Larsen, J. Sierakowski, and M. B. Stegmann. The IMM face database - an annotated dataset of 240 face images. Technical report, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, may 2004.

[29] Unsang Park, Yiying Tong, and Anil K Jain. Age-invariant face recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(5):947–954, 2010.

[30] Pascal Paysan, Reinhard Knothe, Brian Amberg, Sami Romdhani, and Thomas Vetter. A 3d face model for pose and illumination invariant face recognition. In *Advanced Video and Signal Based Surveillance, 2009. AVSS'09. Sixth IEEE International Conference on*, pages 296–301. IEEE, 2009.

[31] Pietro Perona. A new perspective on portraiture. *Journal of Vision*, 7(9):992–992, 2007.

[32] Pietro Perona. Far and yet close: Multiple viewpoints for the perfect portrait. *Art & Perception*, 1(1-2):105–120, 2013.

[33] P Jonathon Phillips, Hyeonjoon Moon, Syed A Rizvi, and Patrick J Rauss. The feret evaluation methodology for face-recognition algorithms. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(10):1090–1104, 2000.

[34] Jakob Santner, Christian Leistner, Amir Saffari, Thomas Pock, and Horst Bischof. Prost: Parallel robust online simple tracking. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 723–730. IEEE, 2010.

[35] Shiguang Shan, Wen Gao, Bo Cao, and Debin Zhao. Illumination normalization for robust face recognition against varying lighting conditions. In *Analysis and Modeling of Faces and Gestures, 2003. AMFG 2003. IEEE International Workshop on*, pages 157–164. IEEE, 2003.

[36] Jianbo Shi and Carlo Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600. IEEE, 1994.

[37] Lawrence Sirovich and Michael Kirby. Low-dimensional procedure for the characterization of human faces. *JOSA A*, 4(3):519–524, 1987.

[38] Andrews Sobral. BGSLibrary: An opencv c++ background subtraction library. In *IX Workshop de Viso Computacional (WVC'2013)*, Rio de Janeiro, Brazil, Jun 2013.

[39] Matthew A Turk and Alex P Pentland. Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on*, pages 586–591. IEEE, 1991.

[40] Grace Wahba. *Spline models for observational data*, volume 59. Siam, 1990.

[41] Kamil Wnuk and Stefano Soatto. Multiple instance filtering. In *NIPS*, pages 370–378, 2011.

[42] John Wright, Allen Y Yang, Arvind Ganesh, Shankar S Sastry, and Yi Ma. Robust face recognition via sparse representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(2):210–227, 2009.

[43] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *Acm computing surveys (CSUR)*, 38(4):13, 2006.

[44] Wenyi Zhao, Rama Chellappa, P Jonathon Phillips, and Azriel Rosenfeld. Face recognition: A literature survey. *Acm Computing Surveys (CSUR)*, 35(4):399–458, 2003.