Face Pose Estimation with Automatic 3D Model Creation in Challenging Scenarios

Pedro Jiménez[a], Luis Miguel Bergasa[a], Jesús Nuevo[b], Pablo F. Alcantarilla[c]

[a]*Department of Electronics*
University of Alcalá, Madrid, Spain
{pjimenez, bergasa}@depeca.uah.es
[b]*Information Communication Technology (ICT) Centre*
Commonwealth Scientific and Industrial Research Organisation, Brisbane, Australia
jesus.nuevo@csiro.au
[c]*ISIT-UMR 6284 CNRS*
Université d'Auvergne, Clermont-Ferrand, France
pablofdezalc@gmail.com

Abstract

This paper proposes a new method to perform real-time face pose estimation for ±90° yaw rotations and under low light conditions. The algorithm works on the basis of a completely automatic and run-time incremental 3D face modelling. The model is initially made up upon a set of 3D points derived from stereo grey-scale images. As new areas of the subject face appear to the cameras, new 3D points are automatically added to complete the model. In this way, we can estimate the pose for a wide range of rotation angles, where typically 3D frontal points are occluded.

We propose a new feature re-registering technique which combines views of both cameras of the stereo rig in a smart way in order to perform a fast and robust tracking for the full range of yaw rotations. The Levenberg-Marquardt algorithm is used to recover the pose and a RANSAC framework rejects incorrectly tracked points.

The model is continuously optimised in a Bundle Adjustment process that reduces the accumulated error on the 3D reconstruction.

The intended application of this work is estimating the focus of attention of drivers in a simulator, which imposes challenging requirements. We validate our method on sequences recorded in a naturalistic truck simulator, on driving exercises designed by a team of psychologists.

Keywords: 3D Face pose estimation, face model, yaw rotation, feature re-registering, stereo vision

1. Introduction

Face pose estimation has been a very active field of research for more than two decades. During this period, the techniques have evolved together with the increasing computational resources of modern computers. Along with this evolution, the objectives of face pose estimation systems have also become more enterprising. Earlier works only aimed to detect a few predefined poses, just enough to allow a coarse pose estimation. Those systems would enable a machine to discriminate the interlocutors of a conversation inside a room with a controlled light environment [1].

Nowadays, as the basic objective of getting fine pose estimation is being met, new requirements can be imposed, depending on specific applications. Some modern pose estimators have errors below 3° [2, 3, 4], but new applications may require the systems to work in real-time, wider rotations ranges, low and variable lighting conditions, user independence or other similar challenging requisites. These new challenges must be addressed by future intelligent face pose estimation systems. Often, the pose estimation algorithm is just a necessary previous step for a gaze estimation system. Gaze is actually what gives the real information of the point of attention of a subject. Accurate gaze estimation thus requires very precise and robust face pose estimation.

Driving inattention is a major factor to traffic crashes, which cost many lives and money every year, everywhere in the world. The latest available data (2009) report 34,500 deaths in the EU27 in traffic accidents, as well as 1.5 million injured, with associated costs representing 2% of the EU GDP. Data gathered for several decades have shown that inattention, which includes drowsiness and distractions, is behind 80% of crashes [5, 6]. Driving distraction is more diverse and implies a riskier factor than drowsiness and it is present in over half of inattention involved crashes [7]. Increasing use of In-Vehicle Information Systems (IVIS) such as cell phones, GPS navigation systems, DVDs and satellite radios and other on-board devices has exacerbated the problem by introducing additional sources of distraction [8]. Enabling drivers to benefit from IVIS without diminishing safety is an important challenge.

Most of the occurrence of distraction can be reflected through the driver's face appearance and face/gaze activity. Focalisation obtained from face pose or gaze estimation can be effective to infer parameters related to distractions. Driver's inattention monitoring systems, developed for the automotive industry, provide very challenging scenarios for face pose estimation

methods. Moreover, inattention monitoring imposes requisites such as real-time, accuracy and good integration. In addition, a consumer on-board application would require complete user independence, no matter age, gender or race, no calibration step, and fast initialisation [9]. Before these systems can be commercialized, they must be exhaustively tested in simulators. Providing a system for distraction analysis and driver's behavioural study inside a simulator adds new challenges. To date, naturalistic scenarios providing incidence data on distracting activities have been small-scale studied. An effort is needed to study distraction problem using naturalistic situations. On the other hand, simulators usually present low light conditions to increase the user immersion feeling. Systems must work under low lighting conditions and must be robust to wide head turns, partial occlusions, different users (with and without glasses) and slight illumination changes.

This paper presents face pose estimation and tracking techniques able to work properly for a driver distraction monitoring application inside a naturalistic simulator. In the state of the art there are few publications about computer vision systems under demanding real driving conditions [10]. There are only some few commercial companies offering their products for face pose estimation [11, 12]. These products are able to work in both indoors and outdoors environments, and require some degree of training for each user. However, no technical information about their relying algorithms has been published and they lack of methodological test validation. We focus our approach to a truck simulator, where ambient illumination is low. This limits the feature matching and tracking techniques to be applied. We tested our proposed method in a motorized simulator, under realistic driving conditions and with professional drivers.

The rest of the paper is structured as follows. Section 2 presents several state-of-the-art face pose estimation works related to our approach. Section 3 describes the general architecture of our approach. Then, Sections 4 and 5 describe respectively the automatic 3D face model creation and face pose estimation with model correction. Section 6 shows performance evaluation and experimental results of our face estimation proposal. Finally, we present some conclusions and future work.

2. Related Work

The huge number of works found in the literature shows that there is an intensive effort by researchers working on this topic, who have developed a wide range of approaches. Despite this effort, it is hard to find works focused in the study of drivers distractions, which is the intended application of this proposal.

Murphy-Chutorian and Trivedi [13] classified head pose estimation systems in eight different categories. Within them, the more recent publications and more promising results are provided by tracking methods, flexible models, and hybrid systems.

There are many methods that produce a coarse output. These have the advantages that do they not rely on face tracking, minimising the possibility of tracking losses. However, their lower accuracy makes those approaches unfeasible for gaze estimation. Generally, gaze is achieved by composing face pose estimation with eye direction [14]. Consequently, if the pose estimation is not precise enough, the gaze estimation will be inaccurate as well.

Focusing on fine output methods, one of the most used approaches in the last years are dimensionality reduction techniques, such as PCA. It became very popular, specially in hybrid architectures, used in conjunction with other approaches such as flexible models [15, 16] or more recently with tracking methods [17, 18]. These dimensionality reduction methods require training and often manual labelling. Moreover, PCA is a linear approach, and consequently is not well suited for the nonlinear problem of wide 3D rotation appearance variations. Some authors applied Kernel-PCA (KPCA) variations [19] to address this problem. Manifold embedding techniques have also been proposed, but their main disadvantage in the inability to separate identity and pose estimation, as the number of users in the training dataset grows. This means that the pose estimation accuracy can vary for different users [20], if the training database is big enough. On the other hand, these methods are a good option for low resolution images, where the little texture information available is well exploited by the dimensionality reduction provided by the embedding.

The non-rigid models also present some problems. The process of calculating new modes for a deformable model is slow. If many modes are allowed, there is a chance that tracking errors of rotations are interpreted as deformation. But as the number of allowed modes decreases, the system gradually loses its non-rigid capability. Paladini et al. [21], for instance, saturate the number of modes to 10 in one video experiment showing an actress talking and moving her head. Most of the deformation is captured by few modes during the first minutes of operation. This avoids increasing execution times, but actually limits the learning process in time.

Much faster algorithms may use flexible models, such Active Appearance Models (AAM) or Active Shape Models (ASM). Flexible models require extensive manual labelling of face landmarks. Using an extensive database, those methods are user independent. In [22], the authors showed that it is possible to achieve very low computational cost using a patch clustering approach. However, the main disadvantage is that they are not suitable for wide head rotations. Related works, such as the ones described in [23, 24], do not show rotations wider than 45°. In addition, models often tend to learn small rotations as deformations, not providing an accurate pose estimation.

Tracking methods, whether only tracking or as part of hybrid systems, provide better accuracy than previous approaches. This technique is user independent, and its implementation can easily meet real-time requirements. Examples are [25, 2] among others, which have errors below 3°. A recent publication [26] presented an online learning model proposal, achieving 3.8° and 4.2° error for pitch and yaw rotations. However, their results were only evaluated in a range of ±40° and ±20° respectively. In the same way, [25], while showing very good results, with an error as low as 2° for yaw rotations, only evaluates the systems for short sequences and small rotations. They

create a static model at the initialisation step, so no wider rotations are possible. It is not clear how well the system can deal with the drifting problem for longer video sequences. SIFT [27] or SIFT-like features have also been used [2]. However, the low lighting conditions in a simulator are not appropriate for SIFT-like matching techniques, as we will show in our results section.

Using a 3D face model notably improves robustness, since it makes possible to detect tracking errors due to appearance similarity of different parts of the face under some rotation. Some authors have used generic face models, such as cylindrical [28] or ellipsoidal [29] ones, and use face appearance mapping to the model shape. However, the wider rotation ranges are provided by sparse models formed from 3D points.

Despite the variety of related works, the face tracking problem is still open, and none of the detailed solutions deal with the problem of having at the same time a full-range, accurate, user independent, real-time and calibration free pose estimation system. Many of the model-based systems rely on generic models, which do not fully adapt to individual geometry. On the other hand, other methods, based on appearance and requiring training, do not generalise well enough to be classified as user independent. A dynamic 3D model can be fitted to any user and give an accurate estimation while being user independent. However, it needs being updated under different user poses, both in geometry and appearance, in order to maintain performance on the full rotation range and illumination changes.

In [30], Jimenez et al. presented a stereo camera system that automatically builds a 3D rigid model of the face. At the beginning of a video sequence, salient features of the face are detected and used to build the model. A modified SMAT [31] was used to model and track the texture around the feature points, independently on each camera. The system computed the 3D pose with POSIT [32], and used RANSAC [33] to remove outliers. The system showed good results for rotations under 45°.

In this paper, we present a series of extensions to that work in order to solve some of its weaknesses. We introduce a 3D model extension method, which dynamically adds new points to the model when the face rotates. We include Bundle Adjustment (BA) [34] to refine the model during creating and extension, and prevent drifting. Feature point tracking has been greatly improved using a new re-registering technique. Now the model considers both the texture of the features and the relative face angle to the cameras. This data is used to advance the texture of the patches as the face rotates. Finally, POSIT has been replaced with the Levenberg-Marquardt (LM) algorithm [35], and the proposal has been widely validated.

3. General Architecture

The proposed face pose estimation approach is based on tracking methods, since they obtain the best accuracy. The system is based on tracking a set of features which are automatically detected on the subject's face, by using a calibrated stereo rig. The algorithm is designed to automatically extract the interest points and to build the 3D model of the face, just requiring the driver to look straight ahead at the initialisation frame. The algorithm is designed to automatically extract the interest points and to build the 3D model of the face, just requiring the driver to look straight ahead keeping his head in vertical position at the initialisation frame.

In order to cope with feature appearance variations due to rotation, a feature template selective re-registering technique is carried out using a novel mixed-views technique using both cameras. In this way, one camera is used to anticipate what the other will see, whereas the other camera is used for tracking, yielding a more robust tracking against changes in appearance and different viewpoints. During yaw rotations, the selective re-registering chooses the frames in which pose uncertainty is minimal to avoid the template drifting problem. For roll and pitch rotations, a feature warping is performed to diminish the projection variation. Incorrectly tracked points (outliers) are detected based on their Euclidean distance to the model point projections after pose estimation, and discarded using a RANSAC [33] process. In addition, a pose uncertainty can also be estimated based on the sum of this Euclidean distance for the inliers. 3D pose is recovered from the set of 2D points assuming weak camera projection. Finally a BA optimisation is used to refine and correct the model.

Initially the model only contains features from a frontal face, which will self-occlude under wide rotations. To increase the range of rotation, it is extended with the addition of new features, when the number of nonoccluded ones falls bellow a minimum. Model extension is automatically performed when the algorithm requires it, and the conditions are appropriate for this. However, the 3D coordinates of a new added feature inherit the specific error related to the poses in which the feature is being added. To correct this, a BA background process constantly corrects the model 3D points at some key-frames. This allows for accurate point addition to the model so the algorithm works reliably for the whole yaw rotation range, ±90° degrees.

The main blocks of the system architecture are shown on Figure 1 and can be summarised as follows:

(a) Initially, a sparse 3D model is automatically built with features extracted from subject's face using a stereo rig.
(b) From frame to frame, the model pose is estimated from the features located. When conditions are met, a novel camera mixed-view re-registering technique is applied in order to improve next model pose estimations.
(c) At some key-frames, re-registering is performed. The 3D model might be extended to previously occluded parts of the face and corrected with BA.

The face rotation at initialisation represents the pose rotation reference, and it is arbitrarily assigned a rotation of 0°, with unitary vector $\vec{u} = (0, 0, 1)$. Following rotation estimations are relative to this reference.

4. Automatic 3D Face Model Creation

Although the 3D model formation takes place during the whole execution of the algorithm, there is an initial model which is automatically created during the first frames of the algorithm. After initialisation, the model will continuously be improved with corrections and extended with new 3D points. The

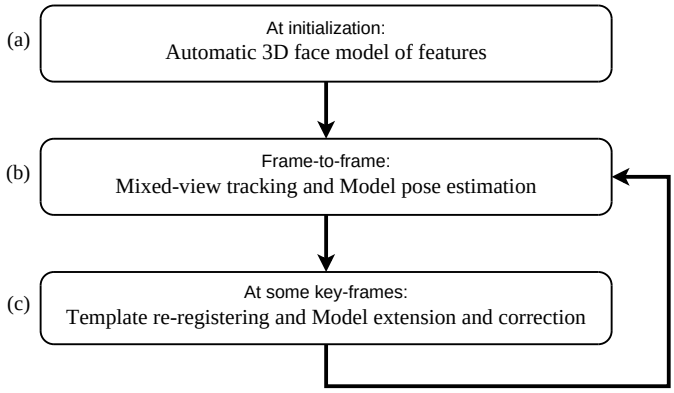


Figure 1: Main blocks of the face pose estimation algorithm.

purpose of this model is to track the user's face in a robust way and to provide a reference from which the pose can be extracted using 2D feature projections on the camera images. Figure 2(a) depicts the different steps involved in the model creation.

The model comprises the 3D coordinates of features and a cluster of its appearance descriptors associated with each one, which are used for 2D tracking and later pose estimation.

4.1. Initial Features Detection and Stereo Matching

To create the model, features must be detected within the bounds of the face. We detect a frontal face using the Viola & Jones algorithm (V&J) [36] in the right and left initial frames.

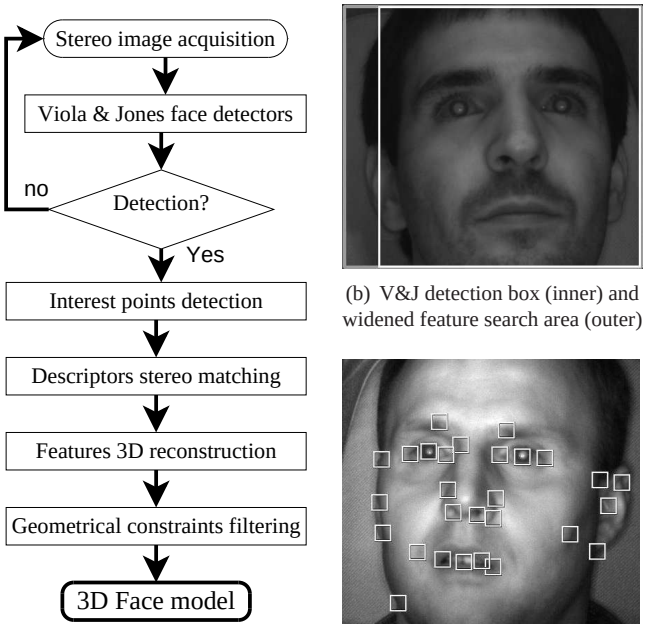
At this step, the user is asked to look forward keeping his head in vertical position, so the V&J can detect an almost frontal face in both images. This will be the only initialisation process the user will be asked to perform. V&J loops frame to frame until the face is detected in both images. These frames are set as the initialisation images, $\mathbf{I}_0^r$ and $\mathbf{I}_0^l$.

Typically, V&J detects a bounding box that can leave outside part of the face, e.g., ears, specially when the face exhibits a small yaw angle with respect to any of the cameras. Due to the base line of the stereo cameras, this is sure to happen at least for one of the cameras, if not for both. To avoid this, the detected V&J bounding box is widened 50 pixels to the left on the right camera image, and to the right for the left camera image. This value has been obtained experimentally. Figure 2(b) depicts the original V&J and widened detection box, and Figure 2(c) the extracted interest points.

The next step is the feature extraction process. A feature $\mathbf{F}_i$ is represented by its appearance template descriptor, $\mathbf{T}_i^{\{r,l\}}$, its 2D position on both camera images, $\mathbf{x}_i^{\{r,l\}}$, and its 3D coordinates, $\mathbf{X}_i$. Each $\mathbf{x}_i^{\{r,l\}}$ is obtained from a set of *interest points* in the image, extracted using the Harris corner detector [37] within the detected face box. After 3D reconstruction, the 2D feature position in the images could actually be computed as the projections of $\mathbf{X}_i$ over each camera image as follows:

$$\mathbf{x}_i^{\{r,l\}} = H^{\{r,l\}}\mathbf{X}_i, \tag{1}$$

where H^r is the projection matrix to the right camera image, and H^l to the left one. The template descriptor, $\mathbf{T}_i^{\{r,l\}}$, is extracted from the patch on each camera image located around $\mathbf{x}_i^{\{r,l\}}$.



(a) Block diagram of the model creation

(b) V&J detection box (inner) and widened feature search area (outer)

(c) Harris interest points

Figure 2: Model creation process.

To obtain a feature $\mathbf{F}_i$, it is first necessary to obtain its 2D projections on each camera, establish the correspondence of $\mathbf{x}_i^r$ to $\mathbf{x}_i^l$ and then compute $\mathbf{X}_i$ by stereo recovery. These interest points represent parts of the image which are likely to be easily matched to their counterpart interest point on the other camera image, and on subsequent frames over time. Consequently, interest points must be easily differentiable from other parts of the image. If for any reason, an interest point can not be matched with any other from the other image, it is discarded.

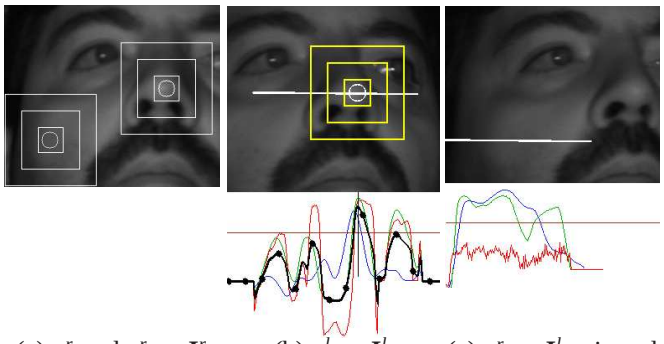
4.2. Multisize Matching Proposal

Different authors have published comparatives on detectors and image registration methods [38, 39]. Most of them cover the general case of well defined objects, full of corners and normal lighting conditions. The low light conditions in the simulator and the face smoothness force the use of correlation techniques. Classical invariant descriptor algorithms do not provide good results.

Feature matching performance is sensitive to different effects depending on the size of the patches used. If a feature changes its appearance because of projection, it works better to match small patches of the image, for which the changes will be more homogeneous than for bigger ones. On the other hand, if the image is not well focused, using bigger patches is more adequate, in order to reduce the number of incorrect matches due to repetitive texture patterns in the face.

As a convenient solution, we characterise each feature texture by three patches of the same scale and different size centred on the feature, and add the correlation of the three patches.

Let $\mathbf{x}_i^r = (u_i^r, v_i^r)$ be a feature candidate or interest point on the initial right image, $\mathbf{I}_0^r$, and $\mathbf{Q}_i^r(m) = \mathbf{Q}(\mathbf{x}_i^r, \mathbf{s}_m)$ be a patch on $\mathbf{I}_0^r$ around $\mathbf{x}_i^r$ of size $\mathbf{s}_m \in \mathbb{R}^2$. To find its corresponding



(a) $\mathbf{x}_j^r$ and $\mathbf{x}_i^r$ on $\mathbf{I}_0^r$ (b) $\mathbf{x}_i^l$ on $\mathbf{I}_0^l$ (c) $\mathbf{x}_j^r$ on $\mathbf{I}_0^l$ rejected

Figure 3: (a) Two feature candidate on image $\mathbf{I}_0^r$. (b) Correct matching (c) Incorrect matching.

$\mathbf{x}_i^l$ on image $\mathbf{I}_0^l$, three patches of different sizes are defined, $\mathbf{Q}_i^r(m)$, $m = 1 \ldots 3$. Then, the three patches are matched over a search area of size $\mathbf{s}_{search}$ on $\mathbf{I}_0^l$, producing matching results $r_{i,m}^l(u,v)$ respectively, all of them of the same size. Zero-mean normalised cross-correlation was used to compare the patches [40]. The search area $\mathbf{s}_{search}$, and consequently the size of the correlation results, is defined as a region seven pixels wide around the epipolar line on $\mathbf{I}_0^l$ corresponding to the point $\mathbf{x}_i^r$, and it is independent of the size $\mathbf{s}_m$ of the patch. The correlation result is expressed as

$$r_i^l(u,v) = \sum_{k=1}^{3} r_{i,m}^l(u,v). \tag{2}$$

The template matching problem can then be formulated as finding the location (u,v) in the image $\mathbf{I}_0^l$ that maximises the objective function

$$\mathbf{x}_i^l = (u_i^l, v_i^l) = \arg\max_{u,v}(r_i^l(u,v)). \tag{3}$$

To ensure the robustness of the feature and to minimise matching error, candidate points which do not meet the condition

$$r_{i,m}^l(u,v) > h_{tc}, \qquad m = 1 \ldots 3, \tag{4}$$

are rejected, where h_{tc} is the matching threshold, set to 0.5 at the stereo matching step. This restriction helps reducing the number of false alarms.

Figure 3 depicts the matching results for two different interest points (a). The three graphs in (b,c) represent the correlation result of the three patches along the epipolar line. In (b) the correspondence is correctly detected and matched, while (c) depicts a failed matching because the smallest patch correlation result is below the threshold, and consequently the interest point is rejected. The three concentric boxes are the three patch sizes for the texture on $\mathbf{I}_0^l$. Since we are looking for stereo correspondence, the search area is restricted to the epipolar line (horizontal line in the images). The matching result is given by the maximum of the black circle marked line. The patch sizes are $\mathbf{s}_1 = 21 \times 21$, $\mathbf{s}_2 = 41 \times 41$ and $\mathbf{s}_3 = 61 \times 61$.

4.3. 3D Face Model

The 3D coordinates of the features are recovered using stereo equations and the calibration parameters of the stereo rig, knowing its 2D projection points on the two camera images of the stereo correspondences. After the stereo reconstruction, we obtain an initial set of n' 3D points

$$\{\mathbf{X}_i\}_{i=1\ldots n'}. \tag{5}$$

From the initial set of n' points, some correspondences may be false alarms, that is, erroneous matched interest points, and must be filtered out before generating the model. The filtering process takes into account face geometrical constraints, like shape and position to ensure the rejection of points outside the face bounds.

4.3.1. Cylinder Model Fitting and Feature Self-Occlusion

One of the common problems to face pose estimation systems based on tracking methods is self-occlusion. The face model features can self-occlude when the head turns over a certain angle, so some of the model points may not be visible. To detect these features in advance, a hidden-point pattern is created during model initialisation. Each feature is associated to two limit rotation angles. Within these angles, the feature is assumed to be visible. When the face rotation angle is over the limit angles of a point, it is considered to be hidden and it is not used for tracking and pose estimation.

Figure 4(a) shows the used 3D coordinate system. To create the hidden-point pattern, a vertically oriented cylinder is adjusted to the $\{\mathbf{X}_i\}_{i=1\ldots n'}$ feature coordinates [41], as we show in Figure 4. The minimisation is implemented inside a RANSAC loop to avoid fitting the cylinder to the most extreme points, such as those of the nose. The outliers threshold h_{Cyl_RANSAC} is chosen small enough so that nose points are outliers to the initial minimisation. On each RANSAC iteration t, a group $\mathcal{N}^{(t)}$ of seven random points is generated, and a cylinder is adjusted to minimise the error function

$$\mathcal{E}^{(t)} = \min_{\theta} \sum_{k \in \mathcal{N}^{(t)}} \mathcal{E}_k(\theta)^2, \tag{6}$$

where

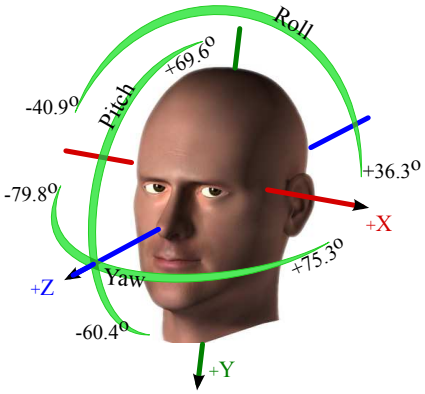
$$\mathcal{E}_k(\theta) = \sqrt{(x - x_k)^2 + (z - z_k)^2} - r \tag{7}$$

is the individual 3D point error function and $\theta = (x, z, r)$ is the parameter list in the minimisation. These parameters represent the centre in the (X, Z) plane and the radius of the fitted cylinder. After each iteration, inliers are calculated as

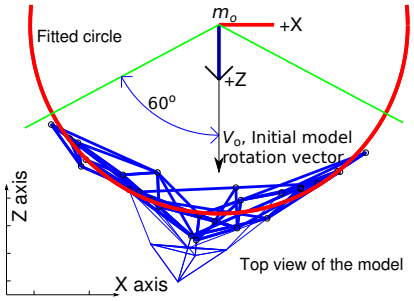
$$\mathcal{I}^{(t)} = \{\mathbf{X}_i\} : \mathcal{E}_i(\theta)^2 < h_{Cyl\_RANSAC}, \quad i = 1 \ldots n', \tag{8}$$

and the best iteration is chosen to maximise the number of inliers. After the RANSAC has found the largest set of inliers $\mathcal{I}$, a new minimisation is executed using all these inliers to find the best set of parameters $\theta_o = (x_0, z_0, r_0)$:

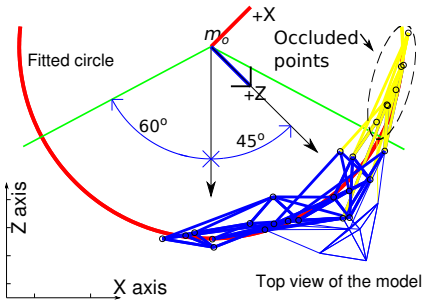
$$\theta_o = \arg\min_{\theta} \sum_{k \in \mathcal{I}} \mathcal{E}_k(\theta)^2 \tag{9}$$



(a) 3D Coordinate system



(b) Frontal face



(c) Partly occluded face

Figure 4: Circle fitted to the face to get the limit angles.

All the angles have an offset inherited from the initial model rotation offset. Each point of the model is considered hidden when its angle with respect to the initial model rotation vector $\vec{V}_0$ exceeds $\pm 60^\circ$ degrees.

The proposed self-occluding model of the face has several advantages. Although the exact occluding angle for each feature is not known, since detailed geometry of the face is not computed, the self-occluding model gives a prediction about when this is likely to happen. This prediction allows to reduce the number of erroneous feature matches at the tracking stage caused by features that are occluded. Finally, the self-occluding model also reduces the computational cost, since all the features that are occluded are not processed.

4.3.2. Model Formation

Initially, n' correspondences were extracted, of which only a set of N_0 are correct and used to form the model $\mathcal{M}$. The model centre is set to $\mathbf{m_0} = (x_o, y_o, z_o)$, where (x_o, z_o) are obtained from the cylindrical model fitting, and y_o is set to

$$y_o = \frac{1}{N_0} \sum_{i \in \mathcal{M}} y_i \tag{10}$$

After computing $\mathbf{m}_0$, those 3D points for which the distance d_i to $\mathbf{m}_0$ is outside a given range are rejected, i.e., the points are far from the 3D cylinder surface, typically 50 mm $< d_i <$ 120 mm, since these points are probably outliers.

The correct correspondences are sorted and translated from camera coordinate frame to head coordinate frame reference system to form the model.

$$\mathbf{X}_i^{(\mathcal{M})} = \mathbf{X}_i - \mathbf{m}_0 \tag{11}$$

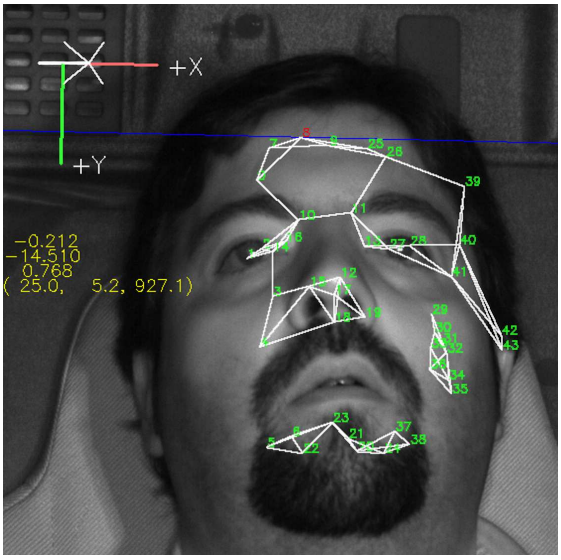
Each model feature i is formed by the 3D point coordinate $\mathbf{X}_i^{(\mathcal{M})}$ and a cluster $\mathbf{C}_i$ of appearance descriptors obtained from the 2D location of the interest points from which the feature was extracted on each camera. Since correlation is being used for feature tracking, following discussion in Section 4.2, the appearance descriptors which form the clusters are the biggest sized patches captured from the images, and will be called *textures* hereinafter, denoted as $\{\mathbf{T}_i^r, \mathbf{T}_i^l\}$. Smaller sized patches can be extracted by subsampling the biggest 2D image patch. These, however, are not the only features which form the model, since it expects new ones, up to N, to be added during tracking to reveal parts of the face initially occluded. Therefore, the model is formed as

$$\mathbf{C}_i = \{\mathbf{T}_i^r, \mathbf{T}_i^l\}, \tag{12}$$

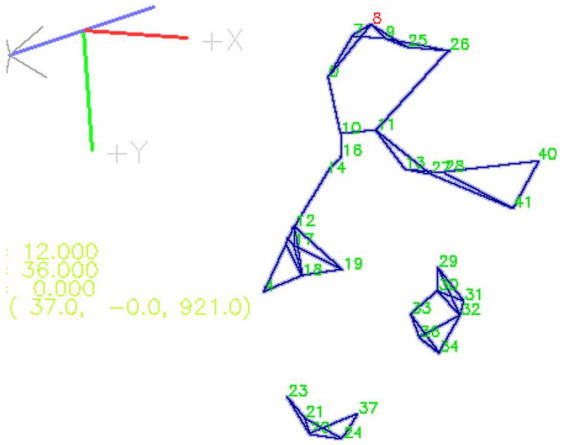
$$\mathcal{M} = \{\mathbf{X}_i^{(\mathcal{M})}, \mathbf{C}_i\}_{i=1\ldots N_0,\ldots N}, \tag{13}$$

where $\mathcal{M}$ is the 3D face model and $\mathbf{X}_i^{(\mathcal{M})}$ are the 3D points of the model in the head coordinate frame. Initial head pointing vector is defined as $\vec{V}_0 = (0, 0, 1)$, with origin in the model centre $\mathbf{m}_0$, and referenced to the right camera frame system.

The coordinates of the model $\{\mathbf{X}_i^{(\mathcal{M})}\}$ are initially set rigidly, and the distances between them are constant. However, methods to dynamically adjust $\{\mathbf{X}_i^{(\mathcal{M})}\}$ and $\{\mathbf{C}_i\}$, that is, model structure and appearance, and to extend the model will be presented in section 5.



(a) Set of projections $\{\mathbf{x}_i^l\}$ over $\mathbf{I}_0^l$. *Note: The image grey scale has been corrected for printing clearness



(b) 3D face model, $\mathcal{M} = \{\mathbf{X}_i^{(\mathcal{M})}\}_{i=1...N}$

Figure 5: Feature projections and created 3D face model.

Figure 5(a) depicts the projections $\mathbf{x}_i^l$ over the camera images $\mathbf{I}_0^l$, and Figure 5(b) shows an example of an automatically generated model.

5. Face Pose Estimation with Model Correction

This section presents the frame to frame execution of the algorithm, which involves the face tracking, pose estimation, feature templates updating and model corrections processes. Figure 6 depicts a flow chart of this process.

5.1. Feature Tracking

The matching technique that we have used for frame to frame feature tracking is the same used for stereo matching: the multisize patch correlation, described in section 4.2. The only difference is a more restrictive correlation threshold to minimise tracking error and outliers. In Equation (4), h_{tc} is now set to 0.7.

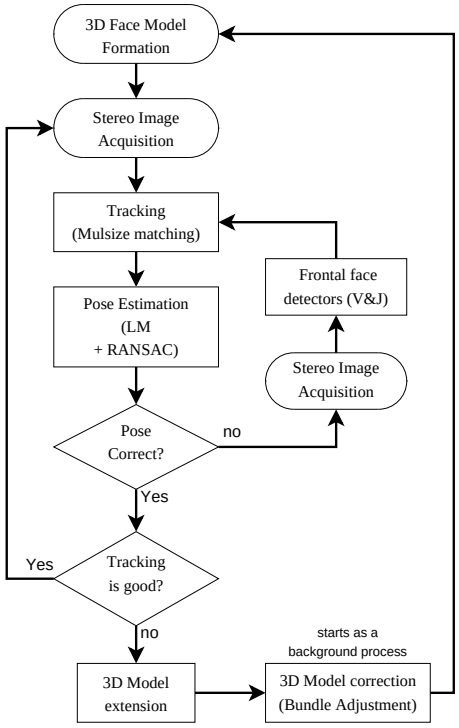


Figure 6: Schematic flow chart of the face tracking and pose estimation algorithm.

5.2. Feature Re-Registering Proposal

As the head rotates, feature appearance changes to levels at which it is not possible to establish a correspondence over the initially registered feature textures. Some algorithms proved to deal better with rotations than others, but none of them is capable of finding the correspondences under wide rotations if no extra information is provided by other means. As face features are not planar in shape, in general it is not a good solution to try a template warping. Moreover, this process is costly, and often needs some *a priori* information about the orientation of the patch in the 3D space.

To deal with feature appearance changes because of 3D rotations we have developed a new *re-registering* technique based on using different view-angles of the face from the two cameras. The idea is to capture new textures from feature patches and to store them in the model when we know that the pose estimation error is the lowest possible. At the model creation step, images patches with different view-points are captured from both cameras, and stored in the model. Instead of using disjointed appearance models for each camera, the stored textures are grouped together for each feature in a cluster. At the tracking stage, some elements of the cluster are correlated in the image, and the stored texture that gives a higher correlation value is used for feature localisation [22].

Figure 7 depicts a comparison of the mean feature localisation error using patches from one camera or from both. This error is calculated as the mean of the localisation errors for all the features which are not hidden and that have been correctly tracked, for a certain face rotation. The first curve shows the mean error using only the initial texture captured with one cam-

era in the correlation step. The second curve shows the mean error if we take the best correlation result of the initial textures captured on each of the cameras. As we can observe, the localisation error is drastically reduced and the tracking is improved by using a cluster that contains the different textures captured from both cameras. The dashed vertical line shows the angle between the two cameras from a distance of approximately 90 cm, at which the face is usually located. Localisation error has a minimum precisely at these rotation angles, because the view-point of the face from one camera is the same as the view-point from the other camera after a rotation of approximately 15°. Figure 7 shows that the optimal angle to perform re-registering is equivalent to the camera separation, because with this yaw rotation the localisation error is minimal. At these rotations, new textures from the feature patches on the image can be captured and stored in the model, since it is at this rotation when the localisation error of these patches is likely to be minimal. Repeating this process all over the yaw rotation range, tracking error can be kept very low under full range rotations.

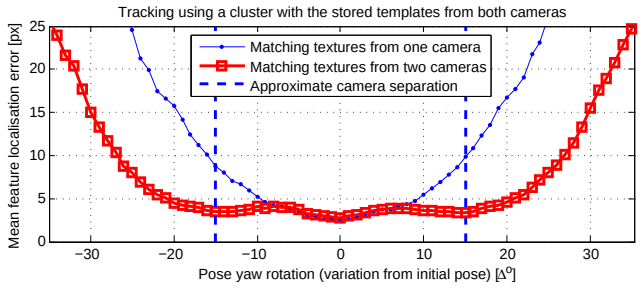


Figure 7: Comparison of localisation error using the textures from one or the two cameras in a cluster.

Following this scheme, new feature appearances are stored in clusters at certain angles α_j, from both camera frames. Let $\mathbf{T}_{ij}$ be a stored texture for feature i and view-point angle α_j, no matter from which camera it was captured. For each feature point i belonging to the model, a cluster $\mathbf{C}_i$ is stored with feature textures from different view-points, j, as

$$\mathbf{C}_i = \{(\mathbf{T}_{ij}, \theta_{ij})\}, j \geq 0. \tag{14}$$

The texture $\mathbf{T}_{ik}$ used for correlation at a certain frame t to search for the feature location in the tracking process will be

$$\mathbf{T}_{ik}: \quad k = \arg\min_j(\mathbf{T}_{ij} - \mathbf{Q}_{i,t}). \tag{15}$$

It is important to notice that the pose is estimated for all rotation angles. However, the re-registering technique can only be performed for yaw rotations because the cameras are placed horizontally. Roll and pitch angles are also estimated, although face appearance templates are not updated for these rotation angles. However, these rotations are smaller compared to yaw, and consequently they are not a big issue in the re-registering technique.

Let $\mathbf{P}_0$ be the 3D model pose at $t = 0$. From this pose, a model point $\mathbf{X}_i$ projects with view-point angle α_0 at $\mathbf{x}^r_{i,0}$ on image $\mathbf{I}^r_0$. $\mathbf{Q}^r_{i,0}$ is the patch around this point seen from right camera C_r. We assume that $\mathbf{P}_0$ is correct by definition, since it is the initial 3D model pose. The texture $\mathbf{T}_{i0} = \mathbf{Q}^r_{i,0}$ from feature i is stored to the cluster. Similarly, β_0 is the projection angle of the feature to the left camera and $\mathbf{T}_{i1} = \mathbf{Q}^l_{i,0}$ from image $\mathbf{I}^l_0$ is also stored to the cluster. This process can be followed in Figure 8 ①, which illustrates the re-registering mechanism.

Now, let the face rotate to its left for a certain time after initialisation and model creation have finished. Let $\mathbf{P}_1$ be the pose at a time t_1 for which the projection angle, β_1, of model point $\mathbf{X}_i$ into the other (left) camera image $\mathbf{I}^l_{t_1}$ is similar to α_0, or more precisely, lower than an error threshold

$$\epsilon > \beta_1 - \alpha_0. \tag{16}$$

If this is the case, the patch $\mathbf{Q}^l_{i,1}$ should be very similar to the stored $\mathbf{T}_{i0}$, previously captured from the other camera, since the projection angles to the respective cameras are the same. Thus, now $\mathbf{T}_{i0}$ can be used to track the new position of $\mathbf{x}^l_{i,1}$ on image $\mathbf{I}^l_{t_1}$ more accurately since $\mathbf{T}_{i0}$ has the same view-point than $\mathbf{Q}^l_{i,1}$, and we previously assumed that this texture is correct. (Figure 8 ②)

At this time, the localisation error is expected to be minimal, and consequently it is convenient to re-register the texture of feature i. A new $\mathbf{T}_{i3} = \mathbf{Q}^l_{i,1}$ is stored to the cluster, captured from the left camera, which should be very similar to $\mathbf{T}_{i0}$.

The angles α_j and β_j are not the same for all the features at a certain frame. However, in practice they are very similar since the size of the face compared with the distance to the camera is small. This means that the frame t_1 can be chosen so that condition in (16) is met for all the features,

$$t_1 : \sum_{i=0}^{i=N} |\beta_{t_1,i} - \alpha_{t_0,i}| < \epsilon'. \tag{17}$$

Error plot on Figure 7 shows that a pose $\mathbf{P}_1$ exists which satisfies this average minima in localisation error. A minimum in average localisation error leads to a minimum in pose estimation error, at the cost of a slight higher error in the captured $\mathbf{T}_i$, since t_1 does not minimise (16) for every single feature at a single frame, but minimises the sum of all of them. This condition implies that $\mathbf{P}_1$ error is also minimal at t_1.

Although the $\mathbf{P}_1$ error is not zero at frame t_1, it is minimum, so it is the best moment to register a texture from camera C_r. The 2D position $\mathbf{x}^l_i$ of the feature is translated to the right camera frame system, $\mathbf{x}^r_i$, knowing $\mathbf{P}_1$. From this location in $\mathbf{I}^r_0$ another new texture $\mathbf{T}_{i2} = \mathbf{Q}^r_{i,1}$ is also stored to the cluster.

Again, after some rotation, there is a time t_2 with pose $\mathbf{P}_2$ for which Equation (17) is minimal,

$$t_2 : \sum_{i=0}^{i=N} |\beta_{t_2,i} - \alpha_{t_1,i}| < \epsilon', \tag{18}$$

and the last stored texture from C_r, $\mathbf{T}_{i2}$, can be used to accurately search for $\mathbf{Q}^l_{i,2}$ in image $\mathbf{I}^l_{t_2}$ on camera C_l (Figure 8 ③).

This process repeats over the whole yaw rotation range. If the face is rotating to its left, the camera C_l is mostly used for tracking and C_r to anticipate the view-point that C_l will have after a small further yaw rotation. Similarly, the process repeats in the opposite direction, interchanging the functions of

C_r and C_l. This procedure generates a cluster as described in Equation (14) of stored textures at discrete angles

$$\alpha_j \approx j \times \theta_c, \qquad j = 0, \pm 1, \pm 2, ..., \tag{19}$$

where the angle θ_c is the average camera separation with respect to the driver's face. $\theta_c \approx 15°$ for our camera layout.

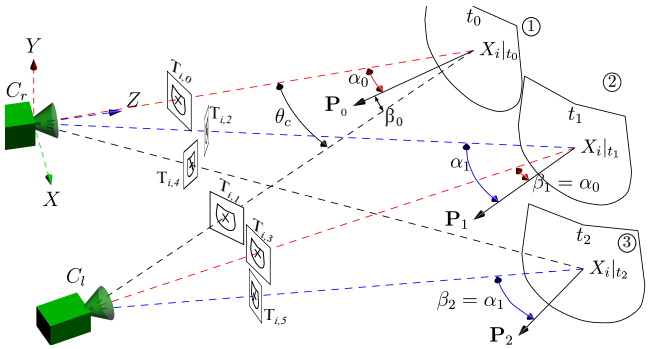


Figure 8: Re-registering process when the face is rotating to its left.

Figure 9 shows the evolution of the correlation results for the tracking of $\mathbf{x}^r_{i,t}$ and $\mathbf{x}^l_{i,t}$ of a feature $\mathbf{X}_i$ over the right and left images when the face is rotating to the left. The graph shows the correlation peaks produced for $\mathbf{x}^r_{i,t}$ when re-registering takes place, at steps of approximately 15°. As for the graph of $\mathbf{x}^l_{i,t}$, its minima represent the points at which the texture used for tracking switches from $\mathbf{T}_{ij}$ to $\mathbf{T}_{ik}$, $j \neq k$. This happens at ±7.5° from the re-registering rotations. Similarly, if the face were rotated to the right, figure 9 would be symmetric, interchanging the roles played by the cameras.

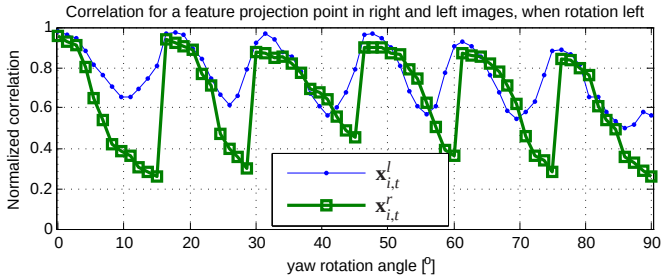


Figure 9: Correlation result of a feature patch for right and left images in a video sequence in which the face rotates to the left after initialisation. In the graph, times advance as the rotation angle increases. The peaks in the right image graph represent the moments at which re-registering happens. These peaks are not exactly one because of driver's movement right after the calibration and because not all the features meet the conditions to be re-registered.

5.3. Pose Estimation

After the position of the tracking points has been updated for the left and right frames, the 3D face pose is estimated using the set of correspondences ($\mathbf{X}^{(\mathcal{M})}_i \Leftrightarrow \mathbf{x}^{\{r,l\}}_{i,t}$) of each feature, i.e. the set of correspondences between the 3D points and their 2D projections over one or both of the camera images.

Whether $\{\mathbf{x}^r_{i,t}\}$, $\{\mathbf{x}^l_{i,t}\}$ or both will be used to extract the pose depends on the pose estimation uncertainty for each frame and tracking error derived from the previous tracking step, as shown in Figure 9. If the head is rotating left, the tracking and subsequent pose estimation is performed over the left image. When the condition in Equation (17) is met, the resulting pose is translated to the right camera, used to project the features 3D points over $\mathbf{I}^r_t$ to accurate obtain $\{\mathbf{x}^r_{i,t}\}$, and the textures from the patches around $\{\mathbf{x}^r_{i,t}\}$ in the image are re-registered. If the head is moving randomly or it is static, pose is estimated from both frames, and the results are averaged.

Pose is estimated using the Levenberg-Marquardt (LM) algorithm. The estimation is formulated as a nonlinear least squares problem, minimising the following cost function considering the right camera as reference as

$$f_{LM} = \arg\min_{\{R,T\}} \sum_i \|\mathbf{x}^r_i - proj(R\mathbf{X}_a + T)\|^2, \tag{20}$$

where $proj()$ is the camera projection function and (R, T) are the rotation and translation matrices to be estimated. The same minimization problem can be obtained considering the left camera as the reference one.

The pose $\mathbf{P} = \{R, T\}$ indicates the position of the central point of the model regarding the camera coordinate system, and its rotation from the initial model. The 3D face pose is computed individually for each camera frame in a RANSAC framework. In each RANSAC iteration, seven points are randomly selected from the model and used to calculate the pose (R and T matrices) using LM. With this R and T, all 3D visible points of the model are projected over the image plane and the Euclidean distance from the tracking point to the corresponding projected point is calculated. If this distance is less than a threshold, this point is considered to be correct, and marked as an inlier. RANSAC iterates until the mean reprojection error drops bellow 3 pixels, or until it has been iterating for approximately 15 ms, so real time performance is not compromised. The outliers error threshold is set big enough to allow certain face deformation. In our case, it is set to 30 pixels.

This process is performed over the frame used to track the points. In case both frames are used, the final pose estimation is calculated for each one, and the result given as the weighted sum, according to the next expressions:

$$R = \frac{R_r \cdot In_l}{In_l + In_r} + \frac{R^r_l \cdot In_r}{In_l + In_r}, \quad \text{if } In_r, In_l > In_{min} \tag{21}$$

$$T = \frac{\vec{T}_r \cdot In_l}{In_l + In_r} + \frac{T^r_l \cdot In_r}{In_l + In_r}, \quad \text{if } In_r, In_l > In_{min} \tag{22}$$

where In_l and In_r are the number of inliers from the left and right pose estimations, as determined with RANSAC. R and T are the resulting pose estimation. R_r and T_r are the pose estimation from the right camera, and R^r_l and T^r_l are pose estimation from the left one, translated to the right one using the corresponding stereo equations and calibration parameters. In case the number of inliers of any of the images is less the In_{min} threshold, that estimation is discarded and the estimation of the other camera is used.

5.4. Model Extension and Correction

The 3D model was created using the initial pair of stereo images of a frontal face. This model is incomplete and the points may contain noise. During the execution of the algorithm, this model is extended and corrected adding new information that can be extracted from successive pairs of stereo frames.

In the event of tracking loss, face detection is performed, using V&J algorithm to find the presence and position of the driver's frontal face. When the face is found, we assume a frontal face and reproject the model points to the area where the face has been found. We search for the exact position of each feature using wider search areas for each feature than those used in the tracking stage.

5.4.1. Model Extension with New Feature Points

Self-occlusion is a drawback of creating the 3D model from a initial single pair of frames. For yaw rotations wider than $\pm 40^0$ approximately, most of the initial points of the model are occluded. The accuracy of the pose estimation depends on the number of features, and thus a model extension procedure is needed. New features from initially concealed face areas are added to the model when all of the next conditions are met:

1. New parts of the face are exposed to the cameras.
2. Pose estimation uncertainty at the current frame is low.
3. The Bundle Adjustment (BA) process has finished correcting the 3D coordinates of previously added points.
4. The number of visible features is higher than a minimum, to ensure algorithm robustness.

The same technique explained in Section 4.3 is used to detect and obtain the 3D coordinates of the new points to be added.

The 3D coordinates of the new points are referenced to the camera coordinate system. They must first be converted to the model reference system, which is now defined as its estimated pose, $\mathbf{P}_t$.

5.4.2. Model Correction Based on Bundle Adjustment

The 3D points taken during model creation and added later are subject to error derived from stereo correspondences. In addition, the newly added points to the model also inherit the error of the pose estimation at the frame of addition. In order to get a better fitting of the model to the face, a Bundle Adjustment (BA) optimisation is used to refine the 3D model. This corrects the 3D point coordinates of the model and the poses at which any point has been added.

Bundle Adjustment is a very popular and well-known technique used in computer vision, and in particular for Structure from Motion (SfM) problems [42, 43, 44]. BA provides an iterative optimisation of the poses and 3D points involved in the reconstruction. Roughly speaking, BA is a non-linear least squares problem and consists in the minimisation of the sum of squared reprojection errors. Furthermore, if the noise in the image error is Gaussian, then BA is the *Maximum Likelihood Estimator* yielding the optimal least squares solution. A complete survey on BA methods can be found in [34].

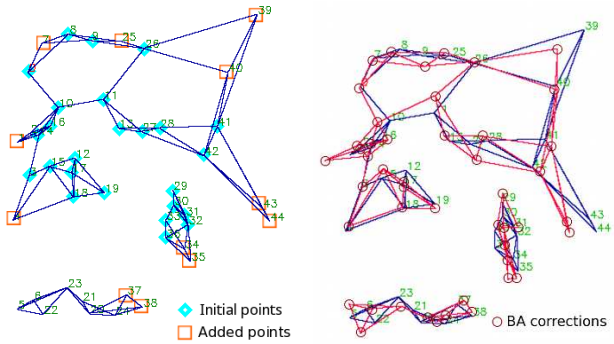
The main problem of BA-based methods is that their speed can be very low when the number of parameters is high, since for solving the full optimisation problem it is necessary to perform the inversion of several linear systems whose size is proportional to the number of estimated parameters. To save on computational load, this stage is only applied at certain *keyframes*, t_w, when a minimum movement has been detected in the pose, and only during certain time after model creation. The process is also executed after points have been added to the model. Each keyframe's pair of images are saved, along with the 2D projection $\mathbf{x}^r_{i,t_w}, \mathbf{x}^l_{i,t_w}$ of the model points and the estimated pose $\mathbf{P}_{t_w}$.

The BA process refines the values of the 3D model points $\mathbf{X}_i$ $(i = 0 \dots N)$ and the past pose estimates $\mathbf{P}_{t_j}$ $(j = 0 \dots w)$, minimising the cost function f_{t_w} which is the sum of the re-projection errors of the 3D model points $\mathbf{X}$ and estimated poses up to keyframe t_w. The error function to minimise in BA is defined as

$$\arg\min_{\mathbf{P},\mathbf{X}} \sum_{P_{t_j} \in \{P_{t_0} \dots P_{t_w}\}} \sum_{X_i \in \{X_0 \dots X_N\}} \left\| \epsilon_i^{t_j} \right\|_2 \tag{23}$$

where $\left\| \epsilon_i^{t_j} \right\|_2$ is the square of the Euclidean distance between the estimated projections of the 3D model point X_i through the pose P_{t_j} and the observed stereo measurements $\mathbf{x}^r_{i,t_j}, \mathbf{x}^l_{i,t_j}$ from the pose P_{t_j}. The process extends until the re-projection error ϵ falls below a pre-defined threshold.

Figure 10 shows the initial model and added points, and the corrections carried out to the model by BA. It can be noticed how corrections are specially needed for the new added points on the laterals of the face.



(a) Original and extended model (b) Corrected model

Figure 10: Initial 3D model, extended one, and BA optimisation. The red lines shows the corrections done by BA. It can be observed how the lateral points suffer bigger corrections. Points 39, 43 and 44 on the right lateral exhibit important corrections.

6. Experimental Tests and Results of the Driver Distraction Monitoring Application

The test environment is a naturalistic truck simulator, shown in Figures 11(a,b), which very accurately recreates day and night time driving conditions. The simulator itself is a real truck cabin, motorised with actuators to simulate driving motion. Three wide projectors outside the cabin show the scene.

The two lateral rear mirrors are also screened, so the driver can look at them to check the traffic behind.

The stereo cameras have a base line of 20 cm, and are located over the dashboard behind the driving wheel, at a distance of between 60 to 100 cm to driver's head. Camera views cannot be parallel, but have a little convergence towards the centre, to point the driver's face, making and angle of 15° between them.

6.1. Ground-Truth

The Ground-Truth (GT) data have been obtained for six different users, using video sequences more than ten minutes long each. The sequences were recorded within a very high immersion environment and simulating common driving disturbances, such as phone calls, handling the GPS and takeovers, which result in frequent head movements.

Two different methods have been used to generate the GT data. In some of the videos, we obtain the GT using a light pattern installed on a tiara, or a calibration pattern attached to the head, as shown in figures 11(c,d). The GT is calculated using MATLAB®, and its output was estimated to have an error below 0.5°. In both cases, the pattern is adjusted to the head, and treated as a disembodied rigid object.

The GT does not provide data on local face features, so this data has been extrapolated from the pose registered in the GT by model point reprojection. Variations of the face due to gestures changes are treated as tracking errors.

6.2. Hardware and Software Requirements

The capture system is formed by two synchronised Basler Scout family FireWire™ cameras and two pulsed IR illuminators synchronised with the cameras. The captured video is high resolution grey scale data at 30 frames per second. The face size is around 300×350 pixels.

The algorithm was tested in a Intel® Core™2 Quad® Processor running Kubuntu 9.10, and equipped with an ATI Radeon™ HD 4500 Series graphic unit from AMD. All code is written in C++, and parallelised using threads. Most of the specific vision operations have been programmed using the OpenCV library [45]. BA and LM algorithms are coded using the libraries provided by Lourakis and Argyros [46, 47].

In Section 6.6, we show a timing evaluation of the proposed face estimation algorithm with incremental 3D model creation. In this timing evaluation we show processing times of each of the main steps of the algorithm.

6.3. Pose Estimation Error Evaluation

The error calculation method used is the same proposed by [48] and [49]. They introduce a scaling factor in the measurements, which depends on some reference size of the object. In their work, they use the distance in pixels between the eyes of the person on the image when the face is frontal to the camera. This scaling factor compensates for the apparent variation in size when the person is closer or further away from the camera, but do not take into account the different size of the face of different subjects. We convert the localisation error to millimetres using the stereo information, so the error figure is independent of the distance to the camera and of the face size. The error measurement can be expressed in millimetres (mm) as

$$m_e = \frac{1}{n}\sum_{i=1}^{n} d_i, \qquad d_i = \sqrt{(\mathbf{X}_i^z - \bar{\mathbf{X}}_i)^T(\mathbf{X}_i^z - \bar{\mathbf{X}}_i)}, \tag{24}$$

where $\bar{\mathbf{X}}_i = (\bar{x}_i, \bar{y}_i, \bar{z}_i)^t$ is the calculated GT position of feature i and $\mathbf{X}_i^z = (x_i, y_i, z_i)^t$ is its 3D coordinate assuming $z_i = \bar{z}_i$. The condition $z_i = \bar{z}_i$ must be applied since $\mathbf{x}_i = (u_i, v_i)$ is estimated over the camera image projection, and there is no z_i information.

6.4. Performance Analysis of the Multisize Matching

We have compared the multisize matching proposal with a classical multiscale matching with Harris detector and with SURF [50]. Figure 12(a) shows the tracking technique without re-registering, while Figure 12(b) shows the same technique using the re-registering algorithm. We use squared patches of size 21, 41 and 61 pixels. Although multisize shows slightly higher error than multiscale for small rotations, the former outperforms the second for wider rotations when re-registering is applied.

As a drawback, mean matching error slightly increases for rotations below 6°. This mainly happens because of the influence of the smallest patches in the less contrasted features. A small patch contains very little texture information, while a low contrast feature also contains less information than those with better contrast. For wider rotations this effect is over passed by the sharpen correlation provided by the small patches. In

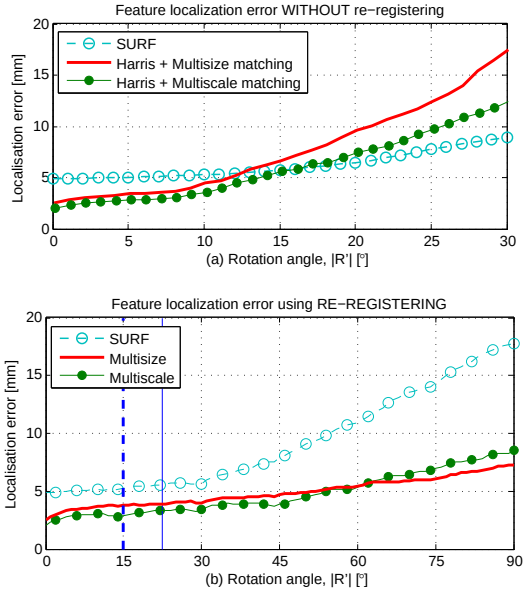


Figure 12: Comparison of tracking errors for different feature tracking methods. (a) Without re-registering (b) With re-registering.

the comparison with SURF, interest points are extracted from the face from both camera's images, and 64-dimensional descriptors are calculated. The SURF tracking error rapidly increase because it fails to extract the correct interest points as

(a) Inside the truck simulation cabin, showing camera layout.

(b) View of the road, cameras and GPS from the driver's position.

(c) GT using a light tiara.

(d) GT using a calibration pattern.

Figure 11: (a,b) Track simulator used to record the video sequences. (c,d) Ground-truth methods.

the face rotates. Figure 13 shows the stereo correspondences for the face features obtained with SURF. Another important effect

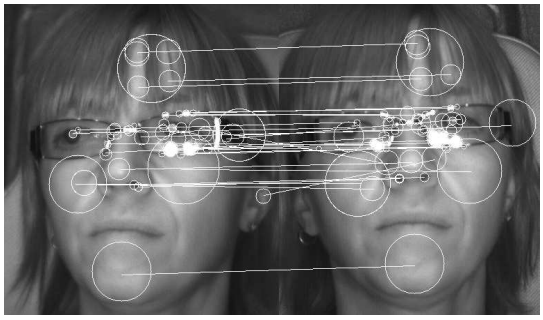


Figure 13: Stereo correspondences of face features obtained using SURF.

that should be noted is that, despite the re-registering, feature localisation error is a monotonically increasing function, due to the accumulated error. The feature tracking stage takes around 14 ms on average for a 30 points model.

6.5. *Performance of the Pose Estimation*

The last steps of the algorithm are the pose estimation and model correction process. Figure 14 depicts the pose estimation error after LM, with and without BA. POSIT [32] pose estimation is provided to compare our results with [30].

Figure 14 shows the correction effect of the underlying BA process. Corrections are especially visible for yaw rotations over 30°, as model extension takes place in that range of angles. It can be observed in both graphs that error increases quickly at that point, even when BA is used, because the addition of new points to the model also adds some error.

In Table 1, we show the pose estimation errors for the three angles of rotation, measured in degrees. Mean errors are computed for each angle of rotation for different rotation ranges. Pitch and roll angles exhibit a smaller rotation range than for yaw, since there are no pronounced head rotations for these angles in the driving experiments. In Table 2, we compare the performance of our proposal with the latest works in the state of the art.

In this case, mean errors in this table have been computed as the average error as a function of face rotation. If the mean errors were averaged over time (average of all individual error

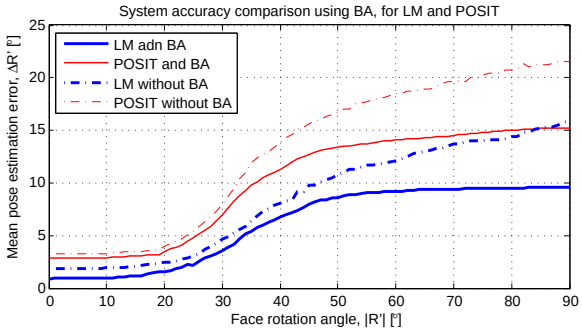


Figure 14: Pose estimation improvement applying the BA algorithm. The error results are shown using POSIT and LM.

measurements on each frame), the influence of the most common poses errors would be higher in the final mean error. Note that other authors do not specify how they calculate this error to take into account the fact that the face is most of the time looking forward.

Because the re-registering technique can not be applied under pitch variations, the system error is higher in this direction, and it can be observed how it increase for angles $\alpha_{pitch} > 30°$. Still, the BA slightly improves the results. For roll rotations a patch warping technique is applied. Consequently, the roll error is lower than pitch error. It was not possible to evaluate the error in a wider pitch and roll range because big pitch and roll rotations are not typical nor natural while driving.

The face pose estimation system has a very low error thanks to the BA corrections. The error remains low for the full range ±90° of yaw rotations. These results show equal or lower errors than other works in the literature, but with more challenging scenarios (wide rotation range, low lighting conditions and fast movements). Figures 15 and 16 depict some results for small pieces of videos. Figure 17 depicts a challenging sequence of video in which the driver moves generating bright illumination in the face, and talks through a microphone to the instructors, generating occlusions and face deformation.

6.6. *Timing Evaluation*

In Table 3 we show average computation times per each of the main steps of the face pose estimation algorithm with automatic 3D model creation. As we can observe, the algorithm can

Rotation	BA?	$\alpha < 15°$	$\alpha < 30°$	$\alpha < 45°$	$\alpha \geq 45^o$
yaw	no	1.92°	2.44°	6.72°	12.83°
yaw	**BA**	**0.98°**	**1.54°**	**3.04°**	**8.54°**
pitch	no	3.82°	7.86°	8.59°	-
pitch	**BA**	**1.81°**	**4.70°**	**6.34°**	-
roll	no	1.27°	2.06°	-	-
roll	**BA**	**1.16°**	**1.75°**	-	-

Table 1: Mean face pose estimation error. The error is divided into *yaw*, *pitch* and *roll*, and evaluated in different ranges of the absolute rotation angle in the ground truth, α.

	Rotation mean error			Rotation range		
Proposal	*yaw*	*pitch*	*roll*	*yaw*	*pitch*	*roll*
Proposed work	**3.8°**	**4.26°**	**1.71°**	**±90°**	**±45°**	**±30°**
RVM [51]	4.1°	2.3°	2.4°	±80°	±25°	±10°
3D model [10]	3.39°	4.67°	2.38°	±90°	±45°	±45°
SIFT [2]	2.44°	2.76°	2.86°	±45°	±45°	±45°
PF [25]	2.86°	2.34°	0.87°	±40°	±20°	±10°

Table 2: Face pose estimation error comparison with other approaches.

work under real-time restrictions (30 Hz approximately). Feature detection just takes place at the initialization of the system. Then, once the model is created, we need to perform per frame feature tracking and pose estimation. The model correction by means of BA, is performed only at certain keyframes throughout the sequence and runs in a parallel processing thread than the tracking and pose estimation.

Algorithm Step	**Execution**	**Time**
Feature Detection	Initialization	15 ms
Model Correction (BA)	Parallel threads	200 ms
Feature Tracking (Approximately 30 features)	Per Frame	14 ms
Pose Estimation (LM)	Per Frame	18 ms
Total	**Per Frame**	**32 ms**

Table 3: Timing evaluation of the proposed face pose estimation algorithm.

7. Conclusions and Future Works

We have implemented a real-time, fully-automatic and user-independent 3D face pose estimation algorithm based on a set of face features. The only calibration required is the stereo rig calibration, which is done offline. A sparse 3D model is automatically created at initialisation, and a technique to refine and improve the model during the whole execution time has been evaluated. This model creation extended in time generates a very accurate model of the face of the subject, providing a very precise pose estimation. In this sense, we have evaluated how to initially create a good model and how to extend it with new features of the face. A bundle adjustment algorithm has been used to correct the model after point addition. The final result is an accurate sparse 3D model.

We found that the well-known and extensively used SURF does not provide good results due to the lack of irregularities and corners in the face, and the low ambient illumination. Instead, we implemented a *multisize* matching technique, based on Harris interest points and patch correlation. This technique joins the goodness of different patch sizes for correlation. Smaller patches give better performance under rotations, while being less sensitive to illumination changes. Bigger ones, on the other hand, are more robust although less accurate. We implemented a new re-registering technique which takes advantage of the stereo cameras disposition. Using this technique we can add new features to the model in a consistent way, no matter whether the driver is in a frontal position or not. This allows for a full range and very accurate face tracking from -90° to +90° yaw rotations.

As face rotates, we use the forward camera in the direction of rotation to capture new texture patches of the features, and the backward camera to track using the patches that were previously captured. This means that a texture patch needs to be tracked only for a range of ±7.5°. For pitch and roll rotation, where the developed re-registering technique can not be applied, patch warping could be used in the future.

The system has been evaluated under challenging conditions and has shown good performance. Results for the proposed algorithm show a mean yaw rotation error below 1° for rotations in the ±15° range, and 1.54° in the ±30° range, improving the results of other works in the literature.

8. Acknowledgements

This work has been financed with funds from the Ministerio de Ciencia e Innovación through the project DRIVER-ALERT (TRA2008 - 03600), as well as from the project CABINTEC (PSE-370000-2009-12).

References

[1] F. Dornaika, B. Raducanu, Three-dimensional face pose detection and tracking using monocular videos: Tool and application, IEEE Trans. Syst., Man, Cybern. 39 (4) (2009) 935–944.

[2] G. Zhao, L. Chen, J. Song, G. Chen, Large head movement tracking using sift-based registration, in: Proceedings of the 15th Int. Conf. on Multimedia, 2007, pp. 807–810.

[3] V. Balasubramanian, S. Panchanathan, Biased manifold embedding: Supervised isomap for person-independent head pose estimation, in: Computer Vision and Computer Graphics. Theory and Applications, Vol. 21, Springer, 2009, pp. 177–188.

[4] M. Krinidis, N. Nikolaidis, I. Pitas, 3D head pose estimation in monocular video sequences using deformable surfaces and radial basis functions, IEEE Trans. Circuits Syst. Video Technol. 19 (2) (2009) 261 –272.

[5] SafetyNet, Annual Statistical Report, European Road Safety Observatory, 2010, www.erso.eu.

[6] Y. Mahieu, Highlights of the panorama of transport, Tech. rep., Eurostats (2009).

[7] J. C. Stutts, D. W. Reinfurt, L. Staplin, E. A. Rodgman, The role of driver distraction in traffic crashes, Tech. rep., AAA Foundation for Traffic Safety (2001).

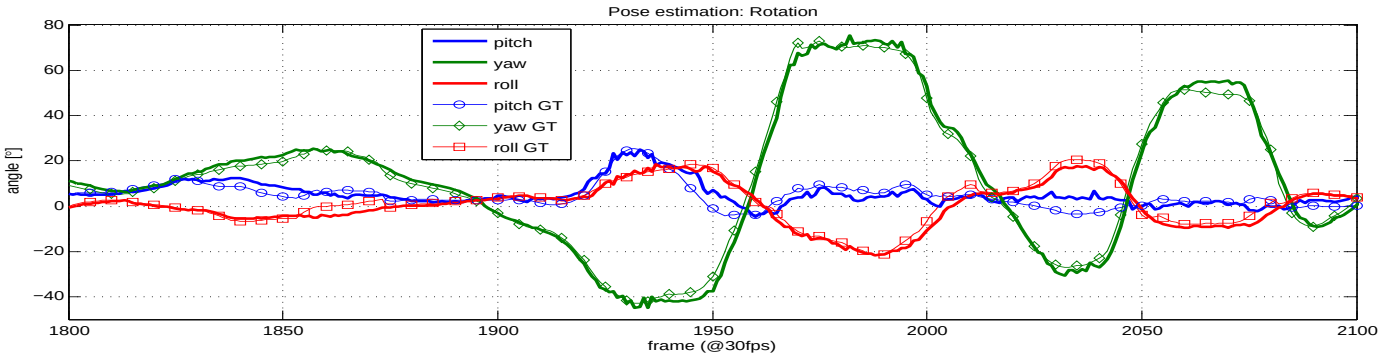


Figure 15: Example of pose estimation. Ground-truth data is shown for comparison.

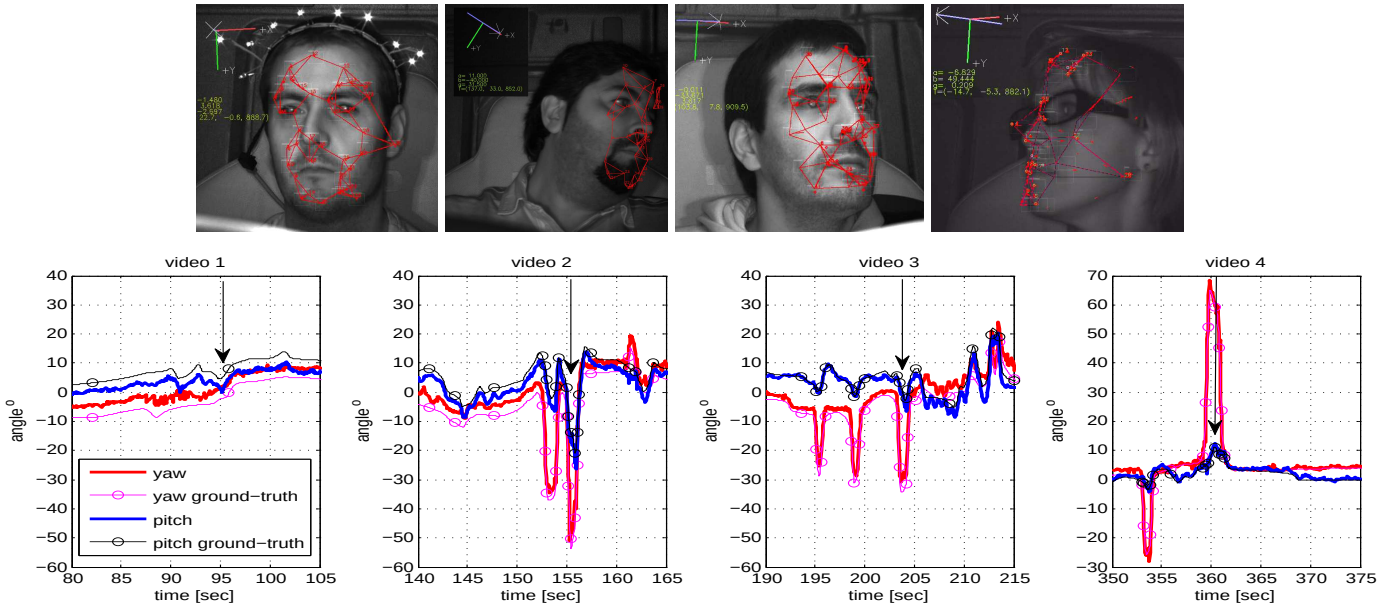


Figure 16: Face pose estimation results. Positive and negative peaks on yaw indicates when the driver is looking at the right or left mirror respectively.

[8] T. A. Ranney, Driver distraction: a review of the current state-of-knowledge, Tech. rep., U.S. Dept. of Transportation, National Highway Traffic Safety Administration, Washington, D.C. (2008).

[9] L. M. Bergasa, J. Nuevo, M. A. Sotelo, R. Barea, E. López, Real-time system for monitoring driver vigilance, IEEE Trans. on Intelligent Transportation Systems 7 (1) (2006) 1524–1538.

[10] E. Murphy-Chutorian, M. Trivedi, Hyhope: Hybrid head orientation and position estimation for vision-based driver head tracking, in: Intelligent Vehicles Symposium, 2008 IEEE, 2008, pp. 512 –517.

[11] Seeing Machines, faceLAB, http://www.seeingmachines.com (2004).

[12] Smart Eye AG, Smart Eye Pro, [web page] www.smarteye.se (2009).

[13] E. Murphy-Chutorian, M. Trivedi, Head pose estimation in computer vision: A survey, IEEE Trans. Pattern Anal. Machine Intell. 31 (4) (2009) 607–626.

[14] D. W. Hansen, Q. Ji, In the Eye of the Beholder: A Survey of Models for Eyes and Gaze, IEEE Trans. Pattern Anal. Machine Intell. 32 (3) (2010) 478 –500.

[15] T. Cootes, C. Taylor, D. Cooper, J. Graham, Active Shape Models-Their Training and Application, Computer Vision and Image Understanding 61 (1) (1995) 38–59.

[16] A. Lanitis, C. J. Taylor, T. F. Cootes, Automatic interpretation and coding of face images using flexible models, IEEE Trans. Pattern Anal. Machine Intell. 19 (1997) 742–752.

[17] K. S. Huang, M. M. Trivedi, Robust real-time detection, tracking, and pose estimation of faces in video streams, Intl. Conf. on Pattern Recognition (ICPR) 3 (2004) 965–968.

[18] J. Tu, T. Huang, H. Tao, Accurate head pose tracking in low resolution video, in: Intl. Conf. on Automatic Face and Gesture Recognition, 2006, pp. 573 –578.

[19] J. Wu, M. M. Trivedi, A two-stage head pose estimation framework and evaluation, Pattern Recogn. 41 (2008) 1138–1158.

[20] V. Balasubramanian, J. Ye, S. Panchanathan, Biased manifold embedding: A framework for person-independent head pose estimation, in: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2007, pp. 1 –7.

[21] M. Paladini, A. Bartoli, L. Agapito, Sequential non-rigid structure-from-motion with the 3d-implicit low-rank shape model, in: Eur. Conf. on Computer Vision (ECCV), Vol. 6312, 2010, pp. 15–28.

[22] J. Nuevo, L. M. Bergasa, P. Jiménez, RSMAT: Robust simultaneous modeling and tracking, Pattern Recognition Letters 31 (2010) 2455–2463.

[23] J. Xiao, S. Baker, I. Matthews, T. Kanade, Real-time combined 2D+3D active appearance models, in: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), Vol. 2, 2004, pp. 535–542.

[24] Z. Gui, C. Zhang, 3D head pose estimation using non-rigid structure-from-motion and point correspondence, in: IEEE TENCON, 2006, pp. 1 –3.

[25] K. Oka, Y. Sato, Y. Nakanishi, H. Koike, Head pose estimation system based on particle filtering with adaptive diffusion control, IEICE Trans. on Information and Systems (2005) 1601–1613.

[26] T. Sheerman-Chase, E.-J. Ong, R. Bowden, Online learning of robust fa-

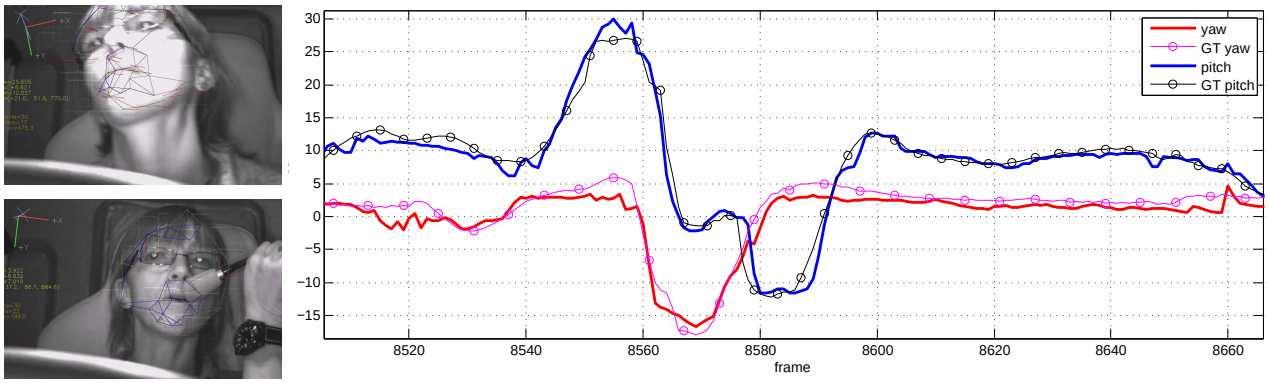


Figure 17: Sequence depicting illumination changes, occlusions and talking.

cial feature trackers, in: Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on, 2009, pp. 1386 –1392.

[27] D. Lowe, Distinctive Image Features from Scale-Invariant Keypoints, Int. J. Comput. Vision 60 (2) (2004) 91–110.

[28] Y. Lin, G. Medioni, J. Choi, Accurate 3D face reconstruction from weakly calibrated wide baseline images with profile contours, in: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2010, pp. 1490 –1497.

[29] K. An, M. Chung, 3D head tracking and pose-robust 2D texture map-based face recognition using a simple ellipsoid model, in: IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS), IEEE, 2008, pp. 307–312.

[30] P. Jiménez, J. Nuevo, L. Bergasa, M. Sotelo, Face tracking and pose estimation with automatic three-dimensional model construction, IET Computer Vision 3 (2) (2009) 93–102.

[31] N. Dowson, R. Bowden, N-tier simultaneous modelling and tracking for arbitrary warps, in: British Machine Vision Conf. (BMVC), Vol. 1, 2006, p. 6.

[32] D. F. Dementhon, L. S. Davis, Model-based object pose in 25 lines of code, Int. J. Comput. Vision 15 (1-2) (1995) 123–141.

[33] M. A. Fischler, R. C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, Comm. of the ACM 24 (6) (1981) 381–395.

[34] B. Triggs, P. McLauchlan, R. Hartley, A. Fitzgibbon, Bundle adjustment – a modern synthesis, in: W. Triggs, A. Zisserman, R. Szeliski (Eds.), Vision Algorithms: Theory and Practice, LNCS, Springer Verlag, 1999, pp. 298–375.

[35] D. W. Marquardt, An algorithm for least-squares estimation of nonlinear parameters, SIAM Journal on Applied Mathematics 11 (2) (1963) 431–441.

[36] P. Viola, M. J. Jones, Robust real-time face detection, Intl. J. of Computer Vision 57 (2) (2004) 137–154.

[37] C. Harris, M. Stephens, A combined corner and edge detector, in: Proc. Fourth Alvey Vision Conference, 1988, pp. 147–151.

[38] P. Moreels, P. Perona, Evaluation of features detectors and descriptors based on 3D objects, Intl. J. of Computer Vision 73 (3) (2007) 263–284.

[39] B. Zitova, J. Flusser, Image registration methods: a survey, Image and Vision Computing 21 (11) (2003) 977–1000.

[40] L. Di Stefano, S. Mattoccia, F. Tombari, ZNCC-based template matching using bounded partial correlation, Pattern recognition letters 26 (14) (2005) 2129–2134.

[41] D. Eberly, Fitting 3D data with a cylinder, http://www.geometrictools.com/ (2003).

[42] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, P. Sayd, Generic and real-time structure from motion using Local Bundle Adjustment, Image and Vision Computing 27 (2009) 1178–1193.

[43] S. Agarwal, N. Snavely, S. M. Seitz, R. Szeliski, Bundle adjustment in the large, in: Eur. Conf. on Computer Vision (ECCV), 2010, pp. 29–42.

[44] M. Byröd, K. Åström, Conjugate gradient bundle adjustment, in: Eur. Conf. on Computer Vision (ECCV), 2010.

[45] G. Bradski, A. Kaehler, Learning OpenCV: Computer Vision with the OpenCV Library, O'Reilly Media, 2008.

[46] M. I. A. Lourakis, A. A. Argyros, Sba: A software package for generic sparse bundle adjustment., ACM Transactions on Mathematical Software 1 (36) (2009) 1–30, [web page] http://www.ics.forth.gr/~lourakis/sba.

[47] M. Lourakis, levmar: Levenberg-Marquardt nonlinear least squares algorithms in C/C++, [web page] http://www.ics.forth.gr/~lourakis/levmar/ (2004).

[48] D. Cristinacce, T. Cootes, Feature detection and tracking with constrained local models, in: British Machine Vision Conf. (BMVC), 2006, pp. 929–938.

[49] J. Nuevo, L. M. Bergasa, D. F. Llorca, M. Ocaña, Face tracking with automatic model construction, Image and Vision Computing 29 (4) (2011) 209 – 218.

[50] H. Bay, A. Ess, T. Tuytelaars, L. V. Gool, SURF: Speeded up robust features, Computer Vision and Image Understanding 110 (3) (2008) 346–359.

[51] Y.-T. Lin, C.-M. Huang, Y.-R. Chen, L.-C. Fu, Real-time face tracking and pose estimation with partitioned sampling and relevance vector machine, in: IEEE Intl. Conf. on Robotics and Automation (ICRA), 2009, pp. 453 –458.