Skin feature tracking using deep feature encodings

Jose Ramon Chang, MSc ¹ and Torbjörn E. M. Nordling, MSc, PhD ¹,²∗

¹Department of Mechanical Engineering
National Cheng Kung University
Tainan 701, Taiwan

²Department of Applied Physics and Electronics
Umeå University
90187 Umeå, Sweden

{jose.chang, torbjorn.nordling}@nordlinglab.org

Abstract

Facial feature tracking is a key component of imaging bal-
listocardiography (BCG) where accurate quantification of the
displacement of facial keypoints is needed for good heart rate
estimation. Skin feature tracking enables video-based quanti-
fication of motor degradation in Parkinson’s disease. Trad-
tional computer vision algorithms include Scale Invariant Fea-
ture Transform (SIFT), Speeded-Up Robust Features (SURF),
and Lucas-Kanade method (LK). These have long represented
the state-of-the-art in efficiency and accuracy but fail when
common deformations, like affine local transformations or il-
luminance changes, are present.

Over the past five years, deep convolutional neural networks
have outperformed traditional methods for most computer vi-
sion tasks. We propose a pipeline for feature tracking, that ap-
plies a convolutional stacked autoencoder to identify the most
similar crop in an image to a reference crop containing the
feature of interest. The autoencoder learns to represent image
crops into deep feature encodings specific to the object category
it is trained on.

We train the autoencoder on facial images and validate its
ability to track skin features in general using manually labeled
face and hand videos. The tracking errors of distinctive skin
features (moles) are so small that we cannot exclude that they
stem from the manual labeling based on a $x^2$-test. With a
mean error of 0.6-4.2 pixels, our method outperformed the
other methods in all but one scenario. More importantly, our
method was the only one to not diverge.

We conclude that our method creates better feature descrip-
tors for feature tracking, feature matching, and image regis-
tration than the traditional algorithms.

Keywords: feature tracking, feature matching, image regis-
tration, autoencoder, Lucas-Kanade method, SIFT

1 Introduction

Facial feature points are distinctive points typically located
around the eyes, nose, chin, and mouth that carry the most
relevant information for both discriminatory and generative
purposes (Wang et al., 2018). With increasingly larger and
broadly available facial imagery datasets, researchers have used
deep learning to teach computers to perform interesting tasks,
such as using these feature points to determine the level of
drowsiness/sleepiness of a driver (Garcia et al., 2012), recog-
nize emotions (Zhao et al., 2016), quantify fatigue (Uchida
et al., 2018), estimate heart rate (Hassan et al., 2017), and
lip-reading (Chung et al., 2017).

Feature tracking can be achieved through iterative feature
matching across sequential frames in a video. Current feature
matching and tracking methods are not robust enough, failing
sometimes even from small photometric/geometric changes,
motion deformations, or occlusions present in the data as
we demonstrate. We define feature matching as the task of,
based on information about the feature in one image, find-
ing the same feature in another deformed image. A feature,
in this case, is any characteristic that is a part of the main
entity/object portrayed in the image. While an object consti-
tutes a whole on its own with a boundary separating it from
other objects and distinguishing it from its environment, e.g. a
car on the street or a bear in a forest, a feature is a part of an
object and does not have a boundary. A feature is defined by a
local pattern of some sort, e.g. a wrinkle on skin, and it is typi-
cally hard to even say where it ends. This makes it significantly
closer to define, detect, and track features compared to
objects. Nonetheless, features matching is sometimes treated
as an object detection/recognition task. In general, the real world
performance of tracking methods decreases with the number of
assumptions made in the method. These assumptions include
but are not limited to: small motion, specific types of geo-
metric deformations, constant lighting, corner points, and no
occurrence of points. Tracking of the position of a skin region of
interest is particularly challenging. In an image, skin typically
appears homogenous with only some irregularly spaced locally
distinct features. The pattern of these locally distinct features
may be heavily deformed due to change in facial expression
or lightning. Some skin features are though unique and easily
recognisable by the human eye, e.g. a mole or birth mark. To
track these feature points, here we follow the established en-
ingineering practice of applying the simplest method proven
to work in other applications. Therefore, we use deep learning,
more precisely an autoencoder, to track crops containing these
distinct skin feature points, aiming to push the limits of cur-
current computer vision methods by creating a more robust, and
accurate pipeline to track skin feature points.

While SIFT [Lowe, 2004] and SURF [Bay et al., 2006] have
been invented almost 20 years ago, they are still considered by
many to be state-of-the-art feature matching techniques. One
research group found SIFT to be the second-best descriptor in
terms of feature matching and recognition accuracy of the same
object or scene (Mikolajczyk and Schmid, 2005). Gradient
Location and Orientation Histogram (GLOH), their proposed
extension of SIFT, marginally outperformed SIFT for most
tasks. This shows the robustness and distinctiveness of SIFT
descriptors. A review found that SIFT and SURF converged
similarly to the lowest errors for monocular visual-odometry
tasks (Chien et al., 2016). Another group concluded that SIFT
is still the most accurate feature descriptor for image recogni-
tion and feature matching applications (Khan et al., 2015).
They also found that general-purpose feature descriptors per-
form fairly well for feature matching but are not good for image
recognition. Some researchers compared SIFT descriptors with
descriptors obtained using a pre-trained Convolutional Neural
Network (CNN) on the Imagenet dataset (Zheng et al., 2017).
They found that while CNN-based methods have advantages in
nearly all their benchmarking datasets, SIFT is still better in
the cases where images are gray-scale, there are intense color
changes, and when there are severe occlusions. One review
concluded that SIFT is the most accurate feature-detector-
descriptor for overall scale, rotations, and affine deformations
(Kareem and Saleem, 2018). We think this is due to the CNN
not having these types of instances in its training distribution.
In the past 2 years, SIFT and SURF have been used in many
applications, like underwater image recognition for au-
tonomous underwater vehicles (Ansari, 2019). Image forgery

Corresponding author
1.1 Deep learning for skin feature tracking

Deep learning has been extensively used in medical applications involving skin. Recent reviews include deep learning for remote photoplethysmography (rPPG) (Ni et al., 2021; Cheng et al., 2021a), gaze estimation (Cheng et al., 2021b), speech recognition (Lee et al., 2021), skin lesion segmentation (Stofa et al., 2019), and motion breath recognition (Tran et al., 2018). In 2015, this method was used for facial expression recognition (Pu et al., 2015).

1.2 Autoencoders for feature matching

The most popular learning paradigm in computer vision, and all fields in general, is supervised learning (Chum et al., 2019; Khan et al., 2021). However, a common bottleneck factor when training a supervised learning model is the amount of good-quality sample-label pairs. The same complexity that allows these models to learn very complex functions is also responsible for them requiring vast amounts of data to tune this extensive amount of parameters.

An autoencoder is a type of neural network that aims to learn the identity function. As such, it can be trained by minimizing the discrepancy or distance between the original data and its reconstruction. This means that it only requires samples for its training and removes the need for labels. The identity function may seem a trivial, meaningless function to learn. However, if we constrain the latent space to have fewer dimensions than our input, then we force the network to learn the most salient, distinctive features of our data. Their use for dimensionality reduction was popularized in 2006 (Hinton and Salakhutdinov, 2006). They described the algorithm as a non-linear generalization of Principal Component Analysis (PCA).

Similarly to PCA, a key feature of autoencoders is that they are able to encode the input data into latent state representations that provide interesting insight into the samples of the data. It is described as a non-linear generalization of PCA because a neural network typically passes the output of each neuron through a non-linear function, e.g. sigmoid, tanh, Rectified Linear Units (ReLU), or all of the ReLU variants.

Autoencoders have been used for feature matching in the past. A siamese network was used by researchers to match the latent space representation of tops and bottoms from the FashionVC and MfFashion (Gao et al., 2019). They used the outfits the fashionistas published on different social networks as their ground truths. Another team used a convolutional autoencoder to match image patches between two infrared images for 3D reconstruction (Knyaz et al., 2017). Their results indicate that their autoencoder was better for matching the features of the test part of their Multi-View Stereo InfraRed (MVSIR) imagery dataset compared to SIFT (Lowe, 2004), deep convolutional feature point descriptors (Simo-Serra et al., 2015), and stereo matching convolutional neural network (Zbontar and LeCun, 2016). They mentioned that if the convolutional autoencoder is able to obtain good restoration quality then it has learned to extract the most salient information from the data. Thus the hidden representations can be used for sparse image matching. Another group used a convolutional autoencoder to learn features from images of finger veins and then used a support vector machine (SVM) to classify them (Hou and Tan, 2019). They credited the convolutional autoencoder for effective learning of finger vein features, preserving the main information of the image, reducing redundant information, and improving the recognition efficiency with the help of their SVM classifier. We train our autoencoder to match skin features in general.

1.3 Deep learning for feature tracking

1.3.1 Object detection based tracking

Most deep learning feature tracking research is built upon the most elementary problems in computer vision where one seeks to locate object instances from a set of predefined categories in images (Liu et al., 2020). An object detection network takes the image as input and outputs a set of bounding boxes. Each of these bounding boxes is typically defined by its x and y coordinates, box width and height, confidence that there is an object inside, and the probability of that object being from one of the predefined classes.

Many deep learning approaches for Multiple Object Tracking (MOT), also known as Multi-Target Tracking, have been published in the last four years (Chaparrone et al., 2020). Similar to object detection, MOT assigns bounding boxes to the detected objects but in addition, each bounding box is associated with a target ID. This target ID serves to differentiate among intra-class objects.

The standard approach of most MOT algorithms is referred to as “tracking-by-detection”, i.e. the tracking is done by iteratively detecting objects successively across the frames of a video. The bounding boxes of two successive frames are usually associated together to assign the same target ID to the bounding boxes that contain the same object. Recently, many methods that have been developed can detect objects with very good accuracy and precision, e.g. Faster-Region-based Convolutional Neural Networks (RCNN) (Ren et al., 2015), Single Shot Detector (Liu et al., 2016), You Only Look Once (YOLO) (Redmon et al., 2016), YOLO v2 (Redmon and Farhadi, 2017), YOLO v3 (Redmon and Farhadi, 2018), YOLO v4 (Bochkovskiy et al., 2020), Mask R-CNN (He et al., 2017a), Region-based Fully Connected Networks (R-FCN) (Dai et al., 2016). Since most of these methods focus on developing good detections, most of these algorithms focus on developing better associating algorithms to track objects with the same target ID. These algorithms are developed for large objects,
e.g. a pedestrian, while we aim to track small objects with low resolutions, e.g. skin features. This makes our problem inherently more difficult as we have less local information and less distinctive features (patterns/structures) to identify our target in the images; leading to a possible increase in false positive cases as there could be many parts of the image with a similar appearance.

The most common deep learning algorithms for detection are variants of Faster-RCNN (Ren et al., 2015) or Single-Shot Detection (SSD) (Liu et al., 2016). While these algorithms have shown to be very accurate at tracking objects, most of these deep learning methods require large amounts of labeled data to train the detector. Labeling data is usually strenuous labor, as accurate training labels require the use of a consensus label of many labeling attempts. This data restraint is a hallmark of supervised learning. The parameters of the model are updated based on the gradients of the error between the prediction and ground truth. In addition, they are still slow and do not meet real-time requirements (Bi et al., 2019). Here we use an autoencoder to avoid the labeling issue.

CenterNet (Zhong et al., 2019) is another object detection algorithm that represents objects as the center of their bounding box. Object size, dimension, 3D extent, orientation, and pose are then regressed directly from image features at this center location. Their best model, which is based on Hourglass-104 (Law and Deng, 2018), is trained to produce heatmaps where peaks correspond to object centers. It achieves 42.2% Average Precision (AP) in 7.8 FPS in the Microsoft (MS) Common Objects in Context (COCO) (Lin et al., 2014). While this is comparable to other state-of-the-art object detectors like Faster-RCNN (Ren et al., 2015), YoloV3 (Redmon and Farhadi, 2018), and RetinaNet (Lin et al., 2017a), it differs significantly from our applications as its more similar to object detection than feature matching. An object recognition algorithm aims to identify the main object in the image. It does so by predicting a probability distribution of all the object categories it has been trained to recognize. An object detection algorithm aims to, in addition to recognizing which are the objects in the image, locate all the main objects by estimating their bounding box locations and dimensions.

1.3.2 Other deep learning based tracking

Deepflow (Weinzaepfel et al., 2013) is a variational approach that uses a CNN to calculate the dense optical flow between two images. Variational approaches had been widely used ever since their invention in 1981 (Hora and Schunck, 1981). They use a six-layered convolutional architecture to create descriptors between images and retrieve quasi-dense correspondences based on these feature descriptors. Their claim for being able to deal with large motions comes from their performance on the Middlebury (Baker et al., 2011) and MPI-Sintel (Butler et al., 2012) datasets. These datasets are well-known in the optical flow community. However, the Middlebury dataset has less than 3% of its pixels with displacements over 20 pixels and none have displacements over 25 pixels. The MPI-Sintel dataset contains approximately 10% of its pixels with a displacement of over 40 pixels. These displacements are still small for our applications. In addition, while they use convolutional layers and max pooling, there is no learning done by the network and all of its kernel parameters are predetermined.

FlowNet (Dosovitskiy et al., 2015) was one of the first attempts to train a deep CNN to estimate optical flow in an end-to-end manner. They developed FlowNetSimple and FlowNetCorr. FlowNetSimple takes as input both images stacked together and tries to predict the optical flow map. FlowNetCorr takes each image as a separate input, processes them, and then uses a correlation layer to identify correspondences between the processed maps. Both networks aim to minimize the endpoint error, which is the Euclidean distance between the predicted flow vector and the ground truth. They mentioned that the amount of frames with ground truth be the used to train the networks. In our case, we would like to distinguish between features of the same domain, e.g. a face (imaging BCG) or a hand (tremor quantification of PD patients). This makes our problem inherently more difficult. FlowNetCorr is a matching algorithm based on these feature pyramids, which is an entirely different part of the feature matching pipeline.

SUPERGLUE (Sarlin et al., 2020) is a graph neural network that can find matches between two sets of features by jointly finding correspondences and rejecting non-matching points. It is a transformer-based network that uses intra-image (self) and inter-image (cross) attention to understand the appearance of keypoints and the spatial relationships between them; respectively. This algorithm takes as input the position and detection confidence of the keypoints in two images along with their respective descriptors and tries to find associations between them. While efficient and precise matching of keypoint pairs is essential in a feature-matching algorithm, our application focuses on the learning and formulation of feature representations.
2 Materials and Methods

Feature tracking can be simplified into doing feature matching solutions. Why for the frames in a video. We want to compare the feature description ability and matching of different algorithms: SIFT, SURF, LK, and our Deep Feature Encodings (DFE). For a full schematic of the flow of our experiments see Figure 1. We decided to track two face features, under two different motion conditions, and one hand feature of a patient with Parkinson’s disease (PD) performing a postural tremor test. The facial features we decided to track are a face mole and a nose tip and the hand feature is a mole, see Figure 2. These are visible in every frame in our validation dataset. We used the default implementation of SIFT in Open CV, i.e., the 128-feature descriptor. We expanded SURF to use rich descriptors (128-feature descriptor). For LK we used the pyramidal implementation using a 10 × 10 window and a 4-leveled pyramid. For the DFE method, we constrained the autoencoder to a compression factor of 0.0444 (128 latent variables) using 31 × 31 × 3 crops as input. Skin is rather homogeneous with distinct features in some places. We, therefore, want to be able to select an area of distinct features to track, which is why it makes sense to work with patches. The task is to match a reference point from the first frame of the video (either face mole, nose tip, or hand mole) in every subsequent video frame. For SIFT, SURF, and DFE, a match is made between the pair of descriptors $h$ that minimize the Sum of Squared Residuals (SSR),

$$SSR = \sum (h_{predicted} - h_{reference})^2,$$

where $h_{predicted}$ is the descriptors of a potential match in the second image and $h_{reference}$ is the descriptor of the feature we want to track. Both must be produced using the same method. The matching strategy of DFE is shown in Figure 3. For LK, the match is simply the point that minimises, in a least-squares sense, the optical flow equations. Only DFE uses color information, coded in the CIELAB color space. SIFT, SURF, and LK all use images in grayscale. To evaluate the matching error of the methods, the images were manually labeled with pixel-level accuracy. A Chi-square analysis was performed on the differences of the $x$ and $y$ coordinates in a series of relabeling attempts for a subset of the validation images to estimate the labelling error and assess the errors of the methods.

2.1 Training dataset

The training data for the autoencoder of our DFE method is the University of Tennessee, Knoxville Face (UTKFace) dataset. UTKFace is a large-scale face dataset with a long age span (range from 0 to 116 years old) [Zhang et al., 2017]. It consists of 24,108 face images in the wild of human of varying sex, age, and ethnicity, see Figure 1A. All the wild face images are RGB color-coded but were converted into the CIELAB color space [McLaren, 1976]. The CIELAB color space aims to be perceptually pseudo uniform. Large differences in the color representations represent large color changes as perceived by humans. The transformation from RGB to CIELAB is described in the section CIELAB color space (Appendix A.3). The images have different backgrounds, lighting, postures, and resolutions. Since the images are collected from the internet, some images seem to be black and white and some include some type of watermark. There is a lot of variety in the face crops as the dataset includes faces with makeup, facial expressions, glasses, facial hair, and other head garments, e.g., headbands, hats, bandanas, bindi, and earrings. To train an autoencoder that is invariant to image scale and specialises in skin features, we have resorted to crops from the original images of the face to obtain images of different resolutions. We use the OpenCV ResNet-10 based model to detect the faces across all the 24,108 pictures in the UTKFace wild faces dataset and select the face with the highest confidence. The results of the face cropping using the ResNet-10 based model are images of aligned cropped faces with the majority of the image corresponding to face information. We subsequently, cropped image patches of size $31 \times 31 \times 3$ from all the detected faces with a stride of 30. These cropped patches serve as input for our deep convolutional autoencoder during the training phase. With this partition, we have approximately 1.9 million training crops.

2.2 Validation dataset

The videos we used to test our algorithms are all collected in our lab for two different projects. The data can be divided into the following sets:

1. Remote ballistocardiography: This dataset was obtained with the goal of estimating the heart rate remotely by tracking the subtle head movement produced by the blood flowing through the carotid artery. The face videos are recorded in full HD resolution (50 FPS) (1920 × 1080 pixels) using a Panasonic GX85 with Olympus M.ZUKIO DIGITAL ED 14-150mm F4-5.6II lens [Wang, 2020]. These videos were recorded under two motion conditions:
   - Small motion—The subject is instructed to simply look at the camera for a couple of minutes without moving, referred to as static condition.
   - Large motion—The subject is filmed while riding an exercise bicycle. The motion is larger than in the static case and periodic. This is referred to as the bike condition.

Both videos show the same subject performing the experiments. The distance-to-pixel ratio was calculated from the physical distance between pupils (mm) and the estimated distance of pupils (pixels) from the first frame in each of the experiments. To estimate the distance of pupils in the images, the two pupils were selected manually and the square root of the L2 norm of their residuals was calculated. The subject has approximately 63 mm of the distance between his pupils. In the static conditions, there are 219.23 pixels between pupils; equating to 0.287 mm/pixel. In the bike conditions, there are 227.06 pixels between pupils; equating to 0.277 mm/pixel.

For the static conditions, the distance between pupils is Face cropping was done by first detecting all faces in the video frames using the Open CV ResNet10 model. The cropping was based on the average bounding box center with width and height scaled at 1.5 times the average width and height of all detected faces across the video frames. This cropping scheme allows us to crop images of the same size where the face dominates while allowing us to observe the motion of the head in the video.

2. Parkinson’s disease postural tremor test: This data set of hands of PD patients has been recorded as a step toward digitalising the Unified Parkinson’s Disease Rating Scale (UPDRS) motor exams. We use this video to evaluate the generalizability of our method to track skin features outside the face. For more information on how the examination was administered see our protocol [Ashyani et al., 2022]. The subject was diagnosed with a Movement Disorder Society-Unified Parkinson’s Disease Rating Scale (MDS-UPDRS) score for postural tremor on the left hand of one. From the three available views (front, right, left) we took the left view of the left hand. The hand video is recorded at 240 FPS in HD resolution (1280 × 720 pixels) using the back camera of a Samsung S7 mobile phone. The area corresponding to the hand was manually cropped using a similar static window approach as for the face dataset, this video is referred to as PD.

Both video datasets were coded in RGB color space but transformed to the CIELAB color space using the algorithm described in section CIELAB color space (Appendix A.3). The videos were also subsampled to 2 FPS to increase the displacement from frame to frame, i.e., tracking challenge, and reduce
the computation time. The final frame count for the face videos after temporal subsampling is 261 for the static condition and 170 for the bike condition. For the hand video, the final frame count was 40 frames. The images were reduced to 420 × 300 pixels to better match the images extracted from the UTKface dataset and to make the evaluation process computationally cheaper.

2.3 Hardware used

The training of the autoencoder was done in an Ubuntu server with an AMD Ryzen Threadripper 1950X 16-core processor with 32 threads and 64 GB of RAM. The system also has three ASUS 1080 Ti GPUs to speed up the parallel computations in the training of the deep neural network.

2.4 Chi-square analysis

A Chi-square analysis was used to establish a threshold for determining which errors were likely a result of human labelling. Errors that are not due to human labelling errors are, by default, attributed to the flaws of the tracking method.

For each body part and motion condition, fifteen images were selected for each skin feature and relabeled five times. These images were equispaced temporally and presented cyclically to the labeller. This was done to prevent the labeller from seeing the same image multiple times in a row. The relabelling was done roughly three months after the original labelling of the images and the labeller was not allowed to take breaks while relabelling each set of 15 images five times. The original labelling was included to obtain a set of 90 samples for each body part and motion condition.

For each relabeling attempt, we computed the error against the mean of all the labeling attempts of the image separately.
Figure 2: Reference crops of the face mole for static conditions (top-left), nose tip for static conditions (top-right), face mole for bike conditions (middle-left), nose tip for bike conditions (middle-right), and the hand mole of the PD patient marked (bottom) by the blue square of size 31 × 31 pixels and their magnifications for representative frames with resolution 420 × 300 pixels.

Figure 3: Flowchart of the algorithm for using an autoencoder for matching facial features.

Table 1: Standard deviations of the error distributions for the manual relabeling attempts.

| Condition         | $\sigma_x$ | $\sigma_y$ |
|-------------------|------------|------------|
| Static face mole  | 0.773      | 1.010      |
| Static nose tip   | 1.165      | 1.256      |
| Bike face mole    | 1.206      | 1.179      |
| Bike nose tip     | 1.337      | 1.319      |
| PD hand mole      | 1.162      | 0.915      |

for each image, body part, and motion condition. This yielded a distribution of the errors for the $x$ and $y$ coordinates that is normal, see section Normality tests for relabeling errors in $x$ and $y$ directions (Appendix A.4). Thus the sum of the standardized squared errors is Chi-square distributed. The standard deviations of $\delta x$ and $\delta y$ are different for each body part and motion condition, as shown in Table 1. The standard deviations of the errors for the nose tips are larger than for the moles. This is a result of the nose tip being a harder point to locate compared to a mole as it lacks the distinctive features necessary for a human to accurately pinpoint it from frame to frame. Thus we call the moles distinctive skin features.

To assess whether the errors are due to human mistakes in the relabeling or proper to the tracking method, we used the following metric. The test statistic, $\hat{\chi}^2$, is the sum of the
standardized squared spatial errors

\[ \chi^2 = \sum_{i=1}^{n} \frac{(x_i - \mu_i)^2}{\sigma_i^2} \]

where \( n \) is the number of frames and \( \sigma_x \) and \( \sigma_y \) are the standard deviations of the errors of the manual labelling for each condition (Table 1) for the \( x \) and \( y \) directions, respectively. We evaluated our test statistic on a Chi-square distribution with 2\( n \) degrees of freedom, \( \chi^2(2n) \), to test our null hypothesis \( H_0 \) of the errors coming from human labelling. Note that significance values smaller than 5% are the percentiles of one distribution versus the percentiles of another (Holmgren 1995). For two distributions with CDFs \( F_1 \) and \( F_2 \) the P-P plot can be represented in functional form as \( F_1(F_2^{-1})(p) \) where \( p \) ranges between 0 and 1. It is used to determine how closely the distributions of two datasets agree. If two distributions are identical, then the values plotted in the P-P plot will lie on the straight line. Deviations from this straight line mean that one distribution stochastically dominates the other.

We plot the standardized squared spatial errors

\[ e_{std} = \frac{\delta x^2}{\sigma_x^2} + \frac{\delta y^2}{\sigma_y^2} \]

d of DFE against a Chi-square distribution with 2 degrees of freedom in Figure 9. The errors were standardized by the variances in Table 1.

2.5 Probability plot for Chi-square distributed errors

A Probability–Probability plot (P-P plot) is a graph of the percentiles of one distribution versus the percentiles of another (Holmgren 1995). For two distributions with CDFs \( F_1 \) and \( F_2 \) the P-P plot can be represented in functional form as \( F_1(F_2^{-1})(p) \) where \( p \) ranges between 0 and 1. It is used to determine how closely the distributions of two datasets agree. If two distributions are identical, then the values plotted in the P-P plot will lie on the straight line. Deviations from this straight line mean that one distribution stochastically dominates the other.

We plot the standardized squared spatial errors

\[ e_{std} = \frac{\delta x^2}{\sigma_x^2} + \frac{\delta y^2}{\sigma_y^2} \]

do of DFE against a Chi-square distribution with 2 degrees of freedom in Figure 9. The errors were standardized by the variances in Table 1.

2.6 Optical flow using Lucas-Kanade method

LK (Lucas and Kanade 1981) uses spatial intensity gradients to direct the search for the best match. It is still a widely used method for sparse optical flow given that it is computationally fast and efficient, making it suitable for many real-time applications. It is assumed that the points in both images are in the same neighborhood and therefore have approximate registration, the brightness is relatively constant, and that distortions are linear.

For a small \( m \times m \) window centered at point \( p \), the local image flow velocity vector \( v \) must satisfy

\[ \nabla I(q_i) \cdot v = -I(q_i) \]

where \( I(q_i) \) and \( \nabla I(q_i) \) are the temporal and spatial gradients respectively for each pixel \( q_i \) inside the \( m \times m \) window. Assuming the window contains \( n \) pixels, denoted \( q_1, q_2, ..., q_n \), we can then write the equation on matrix form

\[ \nabla I v = -I \]

Here, the flow vector \( v \) represents the original position of the point of interest, denoted \( p = [x_p, y_p] \) plus the image velocity vector \( d = [d_x, d_y] \) at \( p \), such that

\[ v = \begin{bmatrix} x_p + d_x \\ y_p + d_y \end{bmatrix} \]

To deal with cases where the motion is large, i.e. more than one pixel, a typical approach is to apply LK on a pyramid of reduced-resolution images (Bouguet 2001). The idea is to successively double the length of a pixel by halving the resolution and reducing fine image details that act as noise for large motion. A pyramid of images has its raw image \( I^0 \) as the lowest level of the pyramid and every subsequent image \( I^1, I^2, ..., I^m \) being the image from the level below sub-sampled to be half the width and height. For the overall pyramidal tracking the algorithm first calculates the optical flow at the deepest level of the pyramid \( I^m \) with the lowest resolution and uses that as an initial guess for the next level \( I^{m-1} \), the optical flow at this level is then used at \( I^{m-2} \), and so on until level 0 (the original image). In our experiments, we used the implementation from OpenCV 3.4.2.17 with a pyramid of 5 levels, \( L_m = 4 \), with a 10 \times 10 observation window at each level.

2.7 Feature matching using SIFT

SIFT (Lowe 2004) is a method to create feature descriptors that are invariant to scale and rotation and enable robust feature matching across changes in viewpoint and illumination in the presence of noise and affine distortions. The idea is that the same keypoint in two images taken from, e.g., different angles should have the same local information, i.e. gradients. The SIFT algorithm for extracting and describing distinctive invariant features, i.e. keypoints, can be divided into four parts, excluding some algorithm details, to foster understanding:

The scale space is constructed by repeatedly halving the image size and blurring at multiple different scales. To each image Gaussian blur is added to remove fine details and noise through convolution of the image with Gaussians, \( G(x, y, \sigma) \), with different scales \( \sigma \). This is implicit in the Difference of Gaussian (DOG)

\[ D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) \ast I(x, y) \]

separated by a scaling factor \( k \), which is used to find blobs. The DOG pyramid is constructed one octave at a time by repeatedly taking every second pixel in each row and column of the previous image and calculating DOGs at different scales. Keypoint localisation is based on finding the local scale space extremum in the constructed DOG pyramid of images and removing low contrast ones. Every extremum in its neighborhood of 26 surrounding pixels in the stack of images with different scales in each octave is considered a candidate keypoint. These are rather poorly localized in spatial and scale space. To obtain the location with subpixel accuracy and filter out low contrast keypoints, a second-order Taylor expansion of \( D(x, y, \sigma) \) is computed at the extremum point, \( z_0 = (x_0, y_0, \sigma_0) \), by approximating the derivative and Hessian using differences of neighbouring sample points

\[ D(z_0 + \delta z) \approx D(z_0) + D_z(z_0)^T \delta z + 0.5 \sigma_0^2 D_{zz}(z_0) \delta z \]

The location of the extremum, \( \delta z \) is determined by taking the derivative and setting it to zero. The value of the DOG at this localized extremum is

\[ D(\hat{x}, \hat{y}, \hat{\sigma}) \approx D(z_0) + 0.5 \sigma_0^2 D_{zz}(z_0) \]

Keypoint candidates with \( D(\hat{x}, \hat{y}, \hat{\sigma}) \) less than a contrast threshold and poorly localized keypoints along an edge are discarded. Edge keypoints have a large principal curvature along the edge but a small principal curvature in the perpendicular direction, i.e. large ratio of the eigenvalues of the Hessian which in the spatial space \((x, y)\) is defined as,

\[ H = [D_{xx} \quad D_{xy} \quad D_{yy}] \]

Any keypoint with an eigenvalue ratio above the edge threshold is discarded.

An orientation is assigned to each keypoint so its descriptor can be made rotation invariant. Based on the scale of the localized keypoint, \( \sigma \), the Gaussian smoothed image \( I(x, y, \sigma) \) closest in scale is selected. Then the gradient magnitude \( |m(x, y)| \) and orientation \( \theta(x, y) \) are computed using pixel differences:

\[ m(x, y) = \sqrt{(L(x + 1, y) - L(x - 1, y))^2 + (L(x, y + 1) - L(x, y - 1))^2} \]

\[ \theta(x, y) = \tan^{-1}(\frac{L(x, y + 1) - L(x, y - 1)}{L(x + 1, y) - L(x - 1, y)}) \]
A 36-binned orientation histogram with 10 degrees bin range is created from these for sample points in a region around the keypoint. The peaks in the histogram represent the dominant direction of the local gradients. The gradient magnitudes of the histogram are weighted by a Gaussian-weighted circular window with a \( \sigma \) that is 1.5 times that of the scale of the keypoint. The peaks of the histogram represent the weighted sum of the gradient magnitudes in that area. Peaks correspond to dominant directions and the highest one determines the keypoint orientation.

Finally, the keypoint descriptor is generated. For each keypoint, after the dominant orientation is found, its coordinates and gradient orientations are rotated relative to this orientation to achieve orientation invariance. The magnitudes and orientations are then smoothed using a \( 16 \times 16 \) Gaussian window. A new orientation histogram is then calculated from the orientations of the smoothed \( 4 \times 4 \) subregions which contain the sum of the gradients and magnitudes within that space. The histogram divides the orientations into \( 8 \) bins and the height of each bin is the magnitude of that orientation. This gives us \( 4 \times 4 \) different histograms, each with \( 8 \) bins. The resulting feature descriptor consists of these histograms concatenated into a \( 4 \times 4 \times 8 = 128 \) feature space. The feature descriptor vector is then normalized to unit length to achieve illumination normalization. We used the default implementation of SIFT in OpenCV 3.4.2.17. The parameters used were \( nFeatures = 0 \), \( nOctaves = 4 \), \( nOctaveLayers = 3 \), \( contrastThreshold = 0.04 \), \( edgeThreshold = 10 \), and \( \sigma = 1.6 \).

After obtaining the keypoint descriptors, SIFT matches a set of keypoints from one image to a set of keypoints in another image by comparing the L2 norm of their residuals. In the original publication, SIFT used the Best Bin First (Beis and Lowe, 1997) algorithm to find approximate nearest neighbors and speed up the matching process. This process does not always return the nearest neighbor. Since we are interested in the ability of the methods to produce feature descriptors, we decided to use the global minimum of the L2 norm of the residuals instead.

We evaluated SIFT under two conditions. First, forcing it to select the point minimizing the SSR. Second, allowing it to filter out points with the nearest neighbor to second nearest neighbor distance ratio greater than 0.8 to remove false positive matches for the high-dimensional descriptors, as described in the original publication (Lowe, 2004). In the second case, for the frames where the distance between the nearest neighbors exceeded the threshold, SIFT failed to find a matching feature. We, therefore, assigned the largest possible error to such frames, i.e., the length of the diagonal of the image 516.14 pixels.

### 2.8 Feature matching using SURF

SURF (Bay et al., 2006) is an algorithm largely based on SIFT. The main goal was to produce a faster feature detector-descriptor by reducing the computation time through several techniques to approximate the solutions used in SIFT. In addition, it also uses by default a simpler, 64-dimensional feature descriptor as opposed to the 128-dimensional feature descriptor used in SIFT.

For the identity differences between SIFT and SURF are the following: SIFT used the Difference of Gaussians to approximate the Laplacian of Gaussians, SURF pushes the approximation one step further and approximates it with box filters. Instead of the feature vectors representing the magnitude and orientation of the gradients, the feature descriptors of SURF are the sum of the Haar responses (and the absolute value of the responses) for the parallel \( x \) and normal \( y \) directions relative to the keypoint orientation. The Haar wavelet response is obtained by comparing how similar image patches are with square filters. The keypoint orientation is defined by the longest vector (largest magnitude) of the sum of Haar wavelet responses in the gradient direction followed by the horizontal response in the abscissa and the vertical response in the ordinate.

The extended version of SURF (named SURF-128) computes the sums of the Haar responses in the parallel direction relative to the dominant keypoint orientation \( d_x \) and \( d_y \) separately for the Haar responses in the normal direction relative to the dominant keypoint orientation \( d_x \geq 0 \) and \( d_y < 0 \). Similarly, the sums of \( d_x \) and \( d_y \) are computed separately for \( d_x \geq 0 \) and \( d_y < 0 \). This yields a descriptor vector \( v \) for each of the \( 4 \times 4 \) regions that is 8 dimensional, as opposed to the original 4 dimensional vector. Consequently, the length of the concatenated descriptors is doubled to 128 dimensions. We have used the extended version of SURF as implemented in OpenCV 3.4.2.17. The parameter values are \( hessianThreshold = 100 \), \( nOctaves = 4 \), \( nOctaveLayers = 3 \), \( extended = true \), and \( upright = false \).

### 2.9 Deep feature encodings

DFE aims to use unsupervised deep learning to learn skin feature representations that are better to track. We use an autoencoder that consists of an encoder subnetwork and a decoder subnetwork. The encoding subnetwork compresses the original data into a latent space that has typically fewer dimensions than the original input while the decoding subnetwork reconstructs the input from the compressed representation created by the encoder. Formally, we can describe the network as a composition of an encoder function \( f \) that creates a representation \( h \) of our input data \( x \), and a decoder function \( g \) that reconstructs the input from the encoded data. We define the output reconstruction \( y \) in terms of the input \( x \) as

\[
y(x) = g(f(x)).
\]

Both \( f \) and \( g \) are functions constructed using 2D convolutional (LeCun et al., 2010) and transpose convolutional (Zeller et al., 2010) layers. We used these as networks that have shown superhuman level performance for vision tasks all have this technology as their foundation (Bachlovskiy et al., 2020; Ren et al., 2015; Liu et al., 2010; He et al., 2017; Siedgey et al., 2015; He et al., 2016). For our skin specific application, these encoded representations \( h \) are the skin feature descriptors of Image crops \( x \) that we compare to find matches of keypoints between face images.

For our experiments, we want to constrain the autoencoder to have a latent space of 128 features to make it a fair comparison with SIFT and SURF with rich descriptors. This architecture yields a compression factor of 0.0444 compared to the original input crop of size \( 31 \times 31 \times 3 \). The loss function used to train the autoencoder was the Mean Squared Error (MSE) between input \( x \) encoded in the CIELAB space and its reconstructed output \( y \) which is also encoded in the same color space.

To encode the \( 31 \times 31 \times 3 \) image crops into a 128-dimensional vector we use a symmetrical convolutional autoencoder architecture, see Figure 2. We downsample the spatial dimensions of the inputs by applying valid convolution operations and iteratively increase the number of channels by increasing the number of filters in our layers. To avoid overfitting and improve generalization, we apply Batch Normalization just before applying the activation function and right after the convolution operation. This allows the activation layer to receive unit Gaussian (\( \mu = 0 \) and \( \sigma = 1 \)) inputs and for the network to control the diffusion of the outputs of the activation layer. In addition, Batch Normalization can become the identity function if the network deems that no normalization is needed. The non-linearity used at each hidden layer is the Rectified Linear Unit (ReLU), which constrains all the activations to non-negative values while preserving the magnitude of all positive inputs.

The network is symmetrical, so the decoder network is similar to the encoder network with the convolutional layers replaced by transposed convolutional layers to upsample the spatial dimensions of the input. The last transposed convolutional layer only has three filters so that the shape of the output is identical to the input and the loss function can be calculated from the feature map-channel pairs. The activation function in the last layer of the network is a Sigmoid to constrain the output values between 0 and 1. The optimizer Adamax is used, i.e., a variant of Adaptive Moment estimation (Adam) (Kingma and Ba, 2014). This network architecture contains 1,160,847 parameters.
2.10 Subpixel level prediction

By default LK estimate the location with subpixel level accuracy. We select the pixel whose feature descriptor minimizes the SSR with the reference descriptor, SIFT, SURF, and our DFE method only achieve pixel-level accuracy. To do a fair comparison of the methods, a subpixel level accuracy of the prediction from SIFT, SURF, and our DFE methods is needed. We perform subpixel level prediction by fitting a surface to the SSR between the reference point and all points of the second image and interpolating locally within a neighborhood of the pixel with minimum SSR [HajijRassouliha et al. 2018].

Let the SSR be represented as a function \( z \), which is dependent on the \( x \) and \( y \) coordinates of points in the image, such that

\[
z = f(x, y).
\]

First, a pixel-level prediction is done by forcing the methods (SIFT, SURF, and DFE) to select the pixel with the minimum SSR through a global search for the minimum in the image space. Second, to obtain subpixel level accuracy a surface of order 2 is fitted to a 3x3 neighborhood of points around the global minimum by solving for \( c \) in the equation

\[
z = Ac
\]

where \( A = [1, x, y, xy, x^2, y^2] \) and \( z \) contains the SSR of the nine points in the 3x3 window. To prevent the matrix \( A \) from becoming ill-conditioned, we center the image coordinates around the 3x3 window by subtracting the coordinates of the middle pixel from all the pixels in the observation window. The surface equation is given by

\[
z = c_0x^2 + c_1y^2 + c_2xy + c_3y + c_4x + c_5.
\]

It is possible that for a single image there are multiple points of the same minimum SSR. In this case, we will select the point with the maximum curvature with a positive 2nd derivative \( z_{xx} \). This ensures that our prediction is a local minimum. The curvature \( D \) is

\[
D = z_{xx}z_{yy} - z_{xy}^2.
\]

This is identical to finding the determinant of a 2 x 2 Hessian matrix

\[
H(z(x, y)) = \begin{bmatrix} z_{xx} & z_{xy} \\ z_{xy} & z_{yy} \end{bmatrix}.
\]

If \( D > 0 \) and \( z_{xx} > 0 \), then according to Sylvester’s criterion the Hessian \( H(z(x, y)) \) is positive definite and the surface has a local minimum. Similarly, if \( D > 0 \) and \( z_{xx} < 0 \), then the surface has a local maximum. If \( D < 0 \), then the eigenvalues of \( H(z(x, y)) \) are of opposite sign, meaning the surface has a saddle point. If \( D = 0 \), then at least one of the eigenvalues of \( H(z(x, y)) \) is zero, yielding the test inconclusive [Abramowitz and Stegun 1988; Thomas and Finney 1961]. That means that there is no curvature in at least one direction, i.e. L-shaped curve or flat surface. Any point that is not a local minimum will be discarded. Lowe discard a point if it does not meet the curvature criterion [Lowe 2004].

Once the point with the lowest SSR, \( D > 0 \), and \( z_{xx} > 0 \) is found, the local minimum of \( z \) is the point where the gradients are zero in both directions, i.e. \( z_x = 0 \) and \( z_y = 0 \). We find this theoretical minimum of the surface by solving \( \nabla f(x, y) = 0 \), which yields

\[
x = \frac{z_3c_4 - 2c_2c_6}{4c_5c_6 - c_4^2} \\
y = \frac{c_2c_4 - 2c_3c_5}{4c_5c_6 - c_4^2}.
\]

2.11 Tracking

We evaluate the tracking of features using two extreme cases. The first uses only the information from the first frame, i.e. no update of the reference feature. A divergent error is less common in this tracking scheme as the method is always searching for the same reference feature in all frames. Even if the method matches the feature poorly, the error can be corrected due to the reference feature always remaining constant, but it is sensitive to change in the feature with time. This tracking scheme is just repeated feature matching frame by frame.

The second is using only the information from the previous frame, allowing the algorithm to replace the reference feature. This makes the tracking insensitive to gradual change in the feature with time. The reference feature is the representation in the 128-dimensional space of the point we want to locate from the previous frame. A divergent error is common as locating and associating features across many frames yields errors that can propagate when the method matches the feature poorly. For the case where we evaluate SIFT by its nearest neighbor to nearest neighbor ratio, we track the last prediction for which the ratio was less than the threshold even if it was several frames ago. That is to prevent SIFT from completely deviating. If SIFT can not assign a prediction, we assign the squared maximum possible distance to that image.

The errors are squared and standardized by the variances in \( x \) and \( y \) directions of each body part and motion condition. These are presented in a cumulative sum for each frame. A 99% CI line is plotted by evaluating the inverse of the \( \chi^2 \) CDF evaluated at a probability of 0.99 with \( 2n_{frame} \) degrees of freedom, where \( n_{frame} \) is the frame number.
3.2 Tracking

The results for tracking only the reference feature from the original image is shown in Figure 7 and replacing the reference feature with the prediction from the previous frame in Figure 8. The average errors using the reference feature from the original image are shown in Table 4 and replacing the reference feature with the prediction from the previous frame in Table 5. In general, DFE is clearly the best method for tracking, achieving the lowest mean error for all conditions except the nose tip under static conditions. In the former case, the cumulative sum of standardized squared errors of DFE even remain within the 99% confidence interval for the distinctive skin mole. In the latter case, it exceeds it, but all other methods have mean errors greater than 100 pixels. The large mean errors are due to the divergence problem discussed in section Materials and Methods. In this tracking scheme, LK also diverges in all conditions, while it in the former scheme did not for the static condition. This renders it not useful for tracking. In contrast, DFE remains useful as it did not diverge; even when all other methods did. The ratios of nearest to second nearest neighbors for SIFT with threshold consistently exceeded the 0.8 as time increased. This is a result of the method diverging to the point where it is no longer tracking features that are distinctive enough for it.

3.3 Probability plot for Chi-square distributed errors

For tracking the moles the empirical data falls on or below the \( x = y \) line of the P-P plot in Figure 9 and while it for the nose tip falls above the \( x = y \) line. This means that for the face mole, the errors of DFE are likely due to human labelling. On the other hand, the errors for the nose tips are likely in part due to the methods, see Table 4. This further emphasizes our assumption that the mole is a more distinctive feature that is easier to track.

The significance level at which we can reject the null hypothesis of the errors coming from human labelling for all conditions, previously described in section Chi-square analysis, can be seen in Table 6. This hypothesis cannot be rejected for DFE when tracking the moles, nor for LK when tracking the face mole under static condition. The \( \chi^2 \)-test statistic is so small that the significance level is near one for DFE. The significance level of SURF for all conditions was smaller than the minimum precision of doubles of 5.0 \( \times 10^{-124} \).

3.4 SSR as a function of spatial coordinates

We looked into the extreme cases for localization performance of SIFT, SURF, and DFE methods. Figure 10 shows the SSR landscape of the frame with the smallest localization error for tracking the face mole under bike conditions. We chose this condition because for the best cases the localization error of all methods is below one pixel of distance. We can see that the SSR landscape of SIFT and SURF is chaotic and does not have any clear structure or pattern. This is in contrast to the SSR landscape of our DFE method which has smaller values near the face. This observation holds independent of the methods localization performance (Appendix A.10). This suggests that our method is better at creating feature descriptors for skin features than the other state-of-the-art methods. A structured SSR landscape will prevent the algorithm from selecting a false positive which is spatially far from the ground truth.
Table 2: Mean errors (in pixels) for feature matching using the same reference feature. The best results are highlighted in bold. SIFT* stands for SIFT with the nearest neighbors threshold.

|                      | SIFT  | SIFT* | SURF  | LK    | DFE   |
|----------------------|-------|-------|-------|-------|-------|
| Static face mole     | 17.28 | 230.09| 93.84 | 1.08  | 1.04  |
| Static nose tip      | 2.37  | 385.69| 48.13 | 2.16  | 2.47  |
| Bike face mole       | 16.33 | 216.25| 74.93 | 4.68  | 0.90  |
| Bike nose tip        | 64.12 | 421.99| 116.63| 11.31 | 3.31  |
| PD hand mole         | 17.29 | 227.19| 85.27 | 33.67 | 0.64  |

Figure 6: Sorted errors for matching the face mole under static conditions (top-left), nose tip under static conditions (top-right), face mole under bike conditions (bottom-left), and nose tip under bike conditions (bottom-right). The black lines represent the 50%, 95%, and 99% CI from the simulated Chi-square distribution. The green dashed line for SIFT with threshold stands for the frames where the nearest neighbor distance threshold was larger than 0.8.

Table 3: Mean errors (in pixels) for tracking using the prediction of the previous image as a reference feature. The best results are highlighted in bold. SIFT* stands for SIFT with the nearest neighbors threshold.

|                      | SIFT  | SIFT* | SURF  | LK    | DFE   |
|----------------------|-------|-------|-------|-------|-------|
| Static face mole     | 191.81| 514.16| 178.66| 179.99| 2.15  |
| Static nose tip      | 70.62 | 514.16| 173.83| 135.94| 4.24  |
| Bike face mole       | 144.41| 513.09| 149.95| 293.76| 2.56  |
| Bike nose tip        | 142.41| 513.09| 150.49| 244.70| 2.30  |
| PD hand mole         | 142.22| 502.92| 150.16| 162.07| 1.30  |

3.5 Generalisation to PD data

The performance of the methods is qualitatively the same as for the face mole under bike conditions based on a comparison of Figures 1, 6, 7, 8, and 9. DFE is still the best method for tracking the mole on the hand of the PD patient. In contrast to Figure 6, the second-best method is SIFT, followed by LK. Although the images in this dataset are qualitatively similar and there is little distortion due to motion, both SIFT and LK for some images yield errors exceeding 100 pixels. This
makes the result qualitatively closer to the face mole under bike conditions than static conditions. SURF again failed to track the mole. For DFE the errors are likely due to human labelling and not the method itself since the sampled errors are lower than the \( x = y \) line in the P-P plot. Figure 12 shows that the SSR landscape of DFE has more structure than the landscapes produced by either SIFT or SURF for the image with smallest error. The tracking results are also qualitatively similar to the ones shown in Figures 7 and 8. These show that DFE is the best tracking method, no matter what tracking scheme is employed, as it was the only method that did not diverge. It also shows that usage of the reference feature from the original image resulted in a smaller cumulative error. These results are qualitatively similar to Figures 7 and 8.

4 Discussion and Conclusions

We present an algorithm that is more accurate and precise, and more importantly, reliable for skin feature matching than the traditional Computer vision methods under the conditions of interest for our ballistocardiography and Parkinson’s disease applications. It is more precise thanks to the global minima containing several nearest neighbors within the range of acceptable predictions without the inherent problem of a large number of false positives that arise from calculating the similarity of two high-dimensional feature vectors in SIFT and SURF. The distinctiveness of the feature representations can be attributed to three main factors: preservation of spatial information through the convolution operation, learning of complex features through hierarchical feature learning, and the
creation of a set of latent variables that is able to capture the variance of the samples. As demonstrated in our experiments, when the matching method makes an obviously wrong prediction of the location of the skin feature, the error is usually extremely large. This will cause tracking algorithms to diverge.

To ensure that our testing is exhaustive, yet simple, and that large changes cannot cause trouble, we currently do not search locally for features but rather do a global search for the best match. Thus, our method is not computationally optimized. It currently takes approximately four times longer than SIFT to run on the validation data with our hardware. We could assume that the skin features have approximate registration and perform a local search to speed up the computation in the future. In addition, the matching scheme could be sped up by implementing an approximate technique for matching neighbors as it has been demonstrated in SIFT. The search could also be optimized if we took advantage of the convexity of the search space, which we have locally for DFE in all cases. We have not considered the occlusion of the key point.

In conclusion, there is still much work to be done before our pipeline is useful in the vast majority of situations. However, the Deep feature encodings have already shown promising results in terms of precision and accuracy for matching features in our applications. Previous feature tracking and feature matching methods are not good enough for the applications we are interested in.

5 Statements and declarations

5.1 Acknowledgments

We thank Chien-Chih Wang for collecting the ballistocardiography data as a part of this Master’s thesis work.

5.2 Funding

This research was supported by the Ministry of Science and Technology in Taiwan (MOST 108-2218-E-006-046, 109-2224-E-006-003, and 110-2222-E-006-010).

5.3 Competing interests

The authors have no financial or non-financial interests that are directly or indirectly related to the work submitted for publication.

5.4 Ethics approval

The use of the UTKface and remote ballistocardiography datasets was reviewed and approved by the National Cheng Kung University Human Research Ethics Review Committee.
under case 108-244. The use of the Parkinson’s Disease postural tremor test dataset was reviewed and approved by the Kaohsiung Medical University Chung-Ho Memorial Hospital Institutional Review Board under the number KMUHIRB-E(I)-20190173.

5.5 Data

Our ethics approval does not allow us to make the validation data publicly available. The training of the autoencoder used in the Deep Feature Encodings can be reproduced using the publicly available UTKface dataset.

5.6 Code availability

Our method can be easily implemented based on the description in the article and pseudo code in the Appendix A.7.

5.7 Authors’ contribution

Author contribution using the CRediT taxonomy: Conceptualization: TN; Data curation: JC; Formal analysis: JC; Methodology: JC and TN; Investigation: JC; Software: JC; Verification: JC and TN; Visualization: JC and TN; Writing - original draft preparation: JC; Writing - review and editing: TN; Funding acquisition: TN; Project administration: TN; Resources: TN; Supervision: TN.

A Appendix

A.1 Autoencoders for feature construction

An autoencoder is a type of neural network that uses an unsupervised training strategy to learn the identity function, see Figure 13. As such, it can be trained by minimizing the discrepancy or distance between the original input data and its reconstruction. The identity function may seem like a trivial and meaningless function to learn. However, if we constrain the latent space to have fewer dimensions than our input, then we force the network to learn the most salient and distinctive features of our data.

Their use for dimensionality reduction was popularized in 2006 by Geoffrey Hinton [Hinton and Salakhutdinov] 2006]. Similar to Principal Component Analysis, a key feature of autoencoders is their ability to encode the input data into latent state representations that provide insight into the samples of the data. Strictly speaking, an autoencoder only requires a
Due to their ability to consider spatial information, CAEs have been especially useful when analyzing imaging data. A deep CAE was used to cluster the embedded features of imaging data of several benchmark datasets (Guo et al. 2017). The method, coined Deep Convolutional Embedded Clustering, can create features that were used to cluster the MNIST handwritten digits with an accuracy of 0.8897. Some other CAE applications include removing artifacts in images, denoising images, and coloring grayscale images. Here we use a CAE to learn skin colors, i.e. when the dimensionality of the latent space is as large as the input (Manakov et al., 2019).

A.2 Statistics of the training dataset

UTKFace dataset of faces in the wild contains 24,108 faces gathered from the internet. The face images are labeled by age, gender, and ethnicity; see Figure 14. The ground truth of face annotations includes removing artifacts in images, denoising images, and coloring grayscale images. Here we use a CAE to learn skin colors.

Figure 10: SSR as a function of the spatial image coordinates for the images with the smallest localization errors for tracking face moles under bike conditions for SIFT (left), SURF (middle), and DFE (right) and their corresponding regional enlargements of the optimum. The red dashed lines mark the point in the space with minimum SSR, which is plotted as a red circle. The distances to the ground truth are 0.04, 0.98, and 0.06 pixels, respectively.

A.3 CIELAB color space

All the images in the UTKFace dataset are coded using the RGB color model. This color model is the conventional model in which most images are created or displayed. However, this color model is device-dependent, i.e. given that the images were created using cameras with different white points (reference whites, target whites) the colors would be represented differently. In addition, the RGB color model does not accurately represent how humans see color. A large sum of squares differences in a pair of crops might not make them seem very different to our eyes, e.g. we consider light green to be similar to yellow but not to red and we consider yellow to be very different from black but more similar to white. Since we want to train a device independent autoencoder to reconstruct images, we are interested in a reconstruction that is device independent and minimizes differences in accordance with how our visual system perceives them.

In 1976 the International Commission on Illumination (Commission internationale de l’´ eclairage in French, CIE) defined the CIELAB color space (also known as CIE L∗a∗b∗). It is a device-independent color model that defines colors relative to the white point of the CIEXYZ space from which they were converted. It represents colors in three axes:

1. \(L^*\) for lightness: Ranging from 0 as black to 100 for white.
2. \(a^*\) for green-red color channels: Ranging from \(-127\) for green to \(127\) to red.
3. \(b^*\) for blue-yellow color channels: Ranging from \(-127\) for blue to \(127\) to yellow.

Here the asterisk (*) is pronounced star and is added in order to differentiate the \(L^*, a^*, b^*\) from Hunter’s \(L, a, b\). The color gamut of CIELAB includes the gamut of the RGB color model, ensuring no information is ignored at the conversion see, Figure 16.
**Figure 11:** Results of running the methods for tracking the mole on the hand of the PD patient. Sorted errors for tracking the mole (top-left), cumulative error for tracking using the reference feature from the original image (top-right), cumulative error for tracking using the reference feature from the previous image (bottom-left), and P-P plot for the standardized squared spatial errors (bottom-right).

The CIELAB color space aims to be perceptually uniform, i.e. the nonlinear relations for \( L^* \), \( a^* \), and \( b^* \) are intended to mimic the nonlinear response of the eyes when looking at different colors. As such, the Euclidean distance between two colors is proportional to the color difference perceived by the standard (colorimetric) observer \((\text{Colantoni et al., } 2016)\). In addition, unlike many color spaces that use cylindrical representations of the hues of colors (HSV, HSL, or HSB), the color representations are non-periodic. This property is especially good for training machine learning models that minimize the sum of squared residuals as the same hue of color can have drastically different values. In HSV the red hue ranges from roughly 345 to 15 with a transition to 0 at 360 in the H channel.

CIELAB is copyright and license-free. It is also mathematically fully defined. A conversion of a floating point image (with intensity range of 0 to 1) in the RGB color space, with a reference white point given by the CIE standard illuminant D65, i.e. common sRGB, to CIELAB color space is given by:

\[
\begin{bmatrix}
X \\
Y \\
Z
\end{bmatrix} =
\begin{bmatrix}
0.412453 & 0.357580 & 0.180423 \\
0.212671 & 0.715160 & 0.072169 \\
0.019334 & 0.119193 & 0.950227
\end{bmatrix}
\begin{bmatrix}
R \\
G \\
B
\end{bmatrix}
\]

\[
X = X / X_n, \quad \text{where } X_n = 0.950456
\]

\[
Z = Z / Z_n, \quad \text{where } Z_n = 1.088754
\]

\[
L = \begin{cases}
116 \cdot Y^{1/3} - 16 & \text{for } Y > 0.008856 \\
903.3 \cdot Y - 16 & \text{for } Y \leq 0.008856
\end{cases}
\]

\[
a = 500(f(X) - f(Y)) \\
b = 200(f(Y) - f(Z)),
\]

where

\[
f(t) = \begin{cases}
t^{1/3} & \text{for } t > 0.008856 \\
7.787t + 16/116 & \text{for } t \leq 0.008856
\end{cases}
\]

The aforementioned conversion is the one employed in OpenCV under the assumption that the image uses the sRGB profile. The conversion takes as inputs floating point RGB (ranging 0 to 1) and outputs CIELAB values with ranges in \( L^* \) from 0 to 100, \( a^* \) from −127 to 127, and \( b^* \) from −127 to 127. It is typical when a variable has a defined upper and lower bound, to normalize it to values between 0 and 1. This will also allow us to directly compare the outputs of the prediction layer in the convolutional autoencoder which has a Sigmoidal function to constrain the values within this range. To normalize
each of these channels to the 0 to 1 range we have implemented a channel-wise min-max normalization technique.

\[
x' = \frac{x - \min(x)}{\max(x) - \min(x)}.
\]

Here \(\min(x)\) and \(\max(x)\) are the absolute minimum and maximum of each respective channel in the CIELAB color space.

Since the images in the UTKface dataset were obtained from the internet, it is not clear if they are encoded in standard RGB using white point D65 or not. However, we assumed all of these to be in standard RGB as this is considered to be the default color space for the internet [Anderson et al., 1996]. The images collected for the remote ballistocardiography dataset were collected using a Panasonic GX85 camera with a default standard RGB color space. The images from the Parkinson’s disease postural tremor test were collected using the main camera of a Samsung S7 phone which also encodes them in standard RGB.

A.4 Normality tests for relabeling errors in x and y directions

The errors in both x and y directions were calculated by subtracting the mean of 6 relabeling attempts for each of the 15 images and concatenating them into a vector with 90 elements.

\[
x = f(h = f(x)) \text{ and then a reconstruction } y \text{ is created from } h \text{ through } g(y = g(h) = g(f(x))).
\]

Figure 13: Typical architecture of an autoencoder network. The input \(x\) is compressed into a representation \(h\) through \(f(h = f(x))\) and then a reconstruction \(y\) is created from \(h\) through \(g(y = g(h) = g(f(x))).\)

Table 5: Number of states that the relabelling errors in both x and y directions are quantized into \((n_{\text{quant}})\).

|                | \(n_{\text{quant}}\) | Range   |
|----------------|-----------------------|---------|
| Static face mole | 19                    | 4.5     | 5.5   |
| Static nose tip  | 25                    | 5.5     | 5.7   |
| Bike face mole   | 31                    | 5.5     | 5.7   |
| Bike nose tip    | 31                    | 5.5     | 5.7   |
| PD hand mole     | 26                    | 6.3     | 6.0   |

The labelling of the image was done at pixel level by a human operator repeatedly clicking on the screen to label the feature. Since the range of the relabelling errors is small, this yields data that is quantized into \(n_{\text{quant}}\) values which are shown in Table 5.

Quantization of a normal distribution makes the distribution more similar to a binomial distribution [Tarongi and Camps, 2010]. Therefore, when applying normality tests, e.g. Lilliefors tests, the data looks non-normal. To avoid this, the number of quantization levels must increase as the sample size increases. The results of a Monte Carlo simulation of sampling 1000 standard normal distributions of 90 values each and applying the Lilliefors test for normality is shown in Figures 17. Only 90 values were drawn to match the number of relabelling errors. These were binned into 1 to 150 bins and replaced by the bin center after min max normalisation to 0 to 1. For the number of quantization states seen in our data (19-31) the Lilliefors test is misleading.

However, quantization only affects the CDF of a distribution by making it look less smooth and more like a staircase. We standardized the errors in x and y directions by their respective standard deviations and compared them against a theoretical standard normal distribution \((\mu = 0 \text{ and } \sigma = 1)\) in Figures 18, 19, and 20. Using this method, we can determine that the manual labelling errors are normally distributed.

Given that the errors in x and y directions are normal, the
Figure 14: Demographic information of race (left), sex (middle), and age (right) of the UTKface wild faces dataset.

Figure 15: Areas (left) and aspect ratios (right) of the cropped faces using the ResNet-10 model applied to the UTKface wild faces dataset. The red lines mark the area and aspect ratio of the resized validation of $1.26 \times 10^5$ pixels and 1.4, respectively.

Table 6: Error thresholds (in pixels) for the three significance levels.

| Condition      | $\alpha = 0.50$ | $\alpha = 0.05$ | $\alpha = 0.01$ |
|----------------|-----------------|-----------------|-----------------|
| Static face mole | 1.049           | 2.222           | 2.789           |
| Static nose tip  | 1.425           | 2.968           | 3.683           |
| Bike face mole   | 1.404           | 2.922           | 3.623           |
| Bike nose tip    | 1.564           | 3.246           | 4.028           |
| PD hand mole     | 1.219           | 2.571           | 3.225           |

The Cumulative Density Function (CDF) of each of the five conditions, is shown in Figure 21, generated by the normal function of the random number routines of numpy version 1.19.3 using Python 3.6. We use the CDF of the Chi-square distribution to determine the likelihood that an error stems from the manual labelling and establish error thresholds at the common 0.01, 0.05, and 0.5 significance levels ($\alpha$). The human errors fall below these thresholds with probability 0.99, 0.95, and 0.5, respectively. The respective thresholds for all the three confidence levels can be seen in Table 6.

The test statistics described in Chi square analysis are described in Table 7 with the respective 99% confidence threshold to reject the null hypothesis of the errors coming from human labelling.

A.5 Calculation of SIFT descriptors

Figure 22 shows how SIFT descriptor are calculated from histograms of the magnitudes of the gradients of neighboring pixels. The magnitude of gradients around a $16 \times 16$ area around the keypoint are weighted using a Gaussian kernel. Histograms of the weighted magnitudes of each $4 \times 4$ area in the $16 \times 16$ area are created; resulting in $4 \times 4$ histograms. The bins in each 8-binned histogram represent the orientation of the gradients and the height of the bars represent the magnitude of that orientation. These histograms are concatenated into a vector of $4 \times 4 \times 8$ dimensions which represents the keypoint descriptor.

A.6 Approximations of Laplacian of Gaussians using box filters

SURF approximates the Laplacian of Gaussian using box filters. Figure 23 shows several examples of these approximations. Box filters is a cruder approximation of Gaussian second derivatives than the Difference of Gaussians used in SIFT (Lowe, 2004). However, the calculation of these box filter approximations is computationally cheap with the use of integral images, independently of their size.

A.7 Implementation of SIFT, SURF, and LK

We used SIFT, SURF, and LK as implemented in version 3.4.2.17 of OpenCV. For SIFT and SURF we used the compute function to calculate the feature descriptors of the given coordinate of the skin feature of interest in the reference frame.
Figure 16: The CIELab color space. The color space has $L^*$ value ranges of 0 to 100, $a^*$ ranges from $-127$ to 127, and $b^*$ from $-127$ to 127. The CIELab color space contains the entire RGB gamut (top). The $a^*$-$b^*$ plane is shown at $L = 20$ (middle-left), $L = 40$ (middle-right), $L = 60$ (bottom-left) and $L = 80$ (bottom-right).

For LK we used the `calcOpticalFlowPyrLK` function to predict the location of the skin feature in the current frame. Details of our feature matching between the reference point and its closest match in the current frame are provided in Algorithm 1 for SIFT and SURF and Algorithm 2 for DFE. A detailed explanation of the methods implemented in the functions `calculate_quadratic_surface_params()` and `get_subpixel_level_pred()` is given in section Subpixel level prediction in the manuscript. Location of the skin feature of interest in the current frame using LK is described in Algorithm 3. Note that we use Python notation for clarity of certain steps and assume the usage of OpenCV and Python functions.
Figure 17: Proportion of samples for which the null hypothesis of the data coming from a normal distribution is rejected using the Lilliefors test, at significance level 0.05 as a function of the number of states that the values were quantized into. All samples were drawn from a standard normal distribution.

Table 7: Test statistics for rejecting the null hypothesis of whether the errors are due to human errors. Statistics that fall below the threshold are in bold.

|                  | SIFT         | SURF         | LK           | DFE           | Threshold |
|------------------|--------------|--------------|--------------|---------------|-----------|
| Static face mole | 6.8887e+05   | 5.0428e+06   | 507.8146     | 467.2694      | 597.9500  |
| Static nose tip  | 1.2527e+03   | 7.4170e+05   | 1.0507e+03   | 1.3334e+03    | 597.9500  |
| Bike face mole   | 2.3114e+05   | 9.2964e+05   | 6.9858e+04   | 401.4090      |           |
| Bike nose tip    | 7.0651e+05   | 1.4756e+06   | 4.8984e+04   | 401.4090      |           |
| PD hand mole     | 1.0317e+05   | 3.5759e+05   | 9.2540e+04   | 109.9581      |           |

The default parameters specific to each method are described in their respective sections in the manuscript.

A.8 Weighted errors

We also weigh the errors $e$, i.e. distances between the prediction and the ground truth, of the validation set by their probability based on the CDF of the simulated Chi square distribution. The weighted error $e_{\text{weighted}}$ is defined as

\[ e_{\text{weighted}} = \frac{e}{1 - F_{\chi^2}(e)}, \]

where $F_{\chi^2}(e)$ is the CDF of the Chi square distribution evaluated at $e$. This weighting scheme penalizes errors that are inherent of the methods and not likely due to human mislabeling. The weighted errors for each condition can be seen in Table 8.

A.9 Maximum errors

The maximum errors for feature matching, i.e. tracking the original reference feature in the first frame, are found in Table 9. DFE achieves the lowest maximum error for all conditions except for tracking the face mole under static conditions. For this condition, the method with the lowest maximum error is LK followed by SIFT. Table 10 shows the maximum errors for tracking the prediction of the previous frame. For this tracking scheme, DFE was the method with the lowest maximum error for all conditions. This is because DFE was the only method that never diverged.

A.10 Tracking errors

A sudden increase in the tracking error shown in Figure 7 occurs for frames 43 to 46 for DFE for tracking the nose tip under bike conditions. This is the only condition for which we can see this sudden increase in the tracking error for DFE. A sudden increase in the error of LK, another method that performed well in most circumstances, also appears in the same frames. A slightly smaller increase is seen in Figure 8 for the same frames for DFE. This suggests these frames are of interest as they are the first frames for which we can see a clear drop in performance. Thus we decided to investigate this drop.

We plotted frames 43 to 46 (in chronological order) along side the original frame in Figure 24. We also plotted the enlarged $160 \times 160$ window around the reference crop and the difference between the reference crop and other windows of the same dimensions in the frames 43 to 46 in grayscale. These are similar to the observation window of LK with the parameters used in our experiments, i.e. window size of $10 \times 10$ and a $L_m$ of 4.

Figure 25 shows the distance between the ground truth and the original reference image and the distances of LK and DFE methods for tracking the nose tip under bike condition temporally. We believe that the large error from LK is due to the ground truth having a very large displacement from the original feature. This displacement causes the new feature to be deformed and thus making the optical flow equations hard to solve. As mentioned previously, an assumption from the LK method is that the motion is small [Lucas and Kanade 1981]. The error in LK has a correlation coefficient of 0.52 with the distance of the ground truth to the original reference feature. Determining whether a correlation is strong is subjective but
Figure 18: CDF of the standardized errors from the videos of static conditions in x direction for face mole (top-left), errors in y direction for face mole (top-right), errors in x directions for nose tip (bottom-left), and errors in y direction for nose tip (bottom-right). The quantized errors were standardized by dividing them by their respective standard deviation. The red line marks the theoretical CDF of a standard normal distribution.

Table 8: Weighted mean errors – the original error weighted by the reciprocal of the significance level of the simulated Chi square distribution. The best results are highlighted in bold.

|                | SIFT       | SIFT with threshold | SURF       | LK         | DFE        |
|----------------|------------|---------------------|------------|------------|------------|
| Static face mole | 1.63 $\times 10^7$ | 1.87 $\times 10^8$  | 1.72 $\times 10^8$ | 1.02 $\times 10^4$ | 6.14 $\times 10^1$ |
| Static nose tip   | 6.05 $\times 10^4$ | 3.14 $\times 10^8$  | 1.63 $\times 10^8$ | **3.75 $\times 10^4$** | 6.35 $\times 10^4$ |
| Bike face mole     | 1.56 $\times 10^7$ | 1.76 $\times 10^8$  | 1.36 $\times 10^8$ | 3.83 $\times 10^6$ | **2.34 $\times 10^4$** |
| Bike nose tip      | 6.29 $\times 10^7$ | 3.45 $\times 10^8$  | 2.57 $\times 10^8$ | 9.40 $\times 10^6$ | **1.11 $\times 10^5$** |

Table 9: Maximum errors for feature matching. The best results are highlighted in bold.

|                | SIFT       | SIFT with threshold | SURF       | LK         | DFE        |
|----------------|------------|---------------------|------------|------------|------------|
| Static face mole | 210.2243   | 516.1395            | 265.0907   | 2.2168     | **2.1347** |
| Static nose tip   | 5.6971     | 516.1395            | 139.7204   | **5.1310** | 5.7030     |
| Bike face mole     | 218.5698   | 516.1395            | 207.2922   | 254.7781   | **2.0818** |
| Bike nose tip      | 79.6503    | 516.1395            | 206.8976   | 76.3600    | **6.4795** |
| PD hand mole       | 219.8465   | 516.1395            | 200.1399   | 120.3210   | **1.5241** |

A correlation coefficient above 0.5 is often considered moderate to strong (Akoglu, 2018). Thus we believe this to in part explain the large errors.
Figure 19: CDF of the standardized errors from the videos of bike conditions in x direction for face mole (top-left), errors in y direction for face mole (top-right), errors in x directions for nose tip (bottom-left), and errors in y direction for nose tip (bottom-right). The quantized errors were standardized by dividing them by their respective standard deviation. The red line marks the theoretical CDF of a standard normal distribution.

Figure 20: CDF of the standardized errors from the videos of PD condition in x direction for hand mole (left) and errors in y direction for hand mole (right). The quantized errors were standardized by dividing them by their respective standard deviation. The red line marks the theoretical CDF of a standard normal distribution.
Figure 21: Simulated cumulative distribution functions for each of the body part and motion conditions.

Figure 22: Calculation of the SIFT descriptor through the histogram of the magnitudes and gradients of neighboring pixels. The neighboring pixels are smoothed using a Gaussian filter (blue circle). The magnitudes and gradients are then accumulated and represented as histograms for each of the sub-regions. The length of each arrow in the sub-sampled space corresponds to the sum of the gradient magnitudes near that direction within that region. This figure shows a $2 \times 2$ descriptor array computed from an $8 \times 8$ set of samples, whereas the original publication (Lowe 2004) use a $4 \times 4$ descriptor array computed from a $16 \times 16$ sample array.

Table 10: Maximum errors for tracking the prediction from the previous frame. The best results are highlighted in bold.
A.11 Nearest neighbors to second nearest neighbor distance ratios

In SIFT a set of keypoints from one image is matched to a set of keypoints in another image by comparing the L2 norm of their residuals (Lowe, 2004). Merely using the global minimum of the L2 norm produces a lot of false positive matches for the high-dimensional descriptors used in SIFT. Lowe resorted to matching keypoints only if the ratio of the descriptor distances, i.e. square root of the SSR, between the nearest neighbor and second nearest neighbor is lower than 0.8. The threshold of 0.8 is an experimental result after comparing the PDF of correct matches versus the PDF of incorrect matches for different closest to second closest neighbor distances on a database of 40,000 keypoints. For their object recognition implementation, the threshold eliminated 90% of false matches while discarding less than 5% of the correct matches. This strategy works well because matches need to have the closest neighbor significantly closer than the closest incorrect match to achieve a reliable matching, hence making the algorithm more robust. For this reason we decided to also investigate the nearest and second nearest neighbor distances.

Figure 24 show the spatial distance to the ground truth of the nearest and second nearest neighbors as a function of their descriptor distances in the 128-dimensional latent feature space with respect to the reference descriptor for our DFE method, i.e. the square root of the SSR. In 56% of the images, the spatial distance between the ground truth and the second nearest neighbor is larger than that of the nearest neighbor. In 90% of the images, the largest spatial distance between any of the
Algorithm 1 Feature matching using SIFT and SURF to locate the skin feature.

Require: \( \text{ref}_\text{point}, \text{img}_\text{Reference}, \text{img}_\text{current} \)

Ensure: pred ▷ Location of best match
dep ▷ Initialize parameters

1: \( \text{size}_\text{window} = (31,31) \), stride = 1
2: positions = [], curvatures = [], candidates = []
3: \( \text{descriptor} = \text{cv2.xfeatures2d.method}_\text{create}() \) ▷ Create SIFT/SURF object by replacing “method” in the function name by SIFT/SURF
4: ref\_descriptor = descriptor.compute(img\_Reference, ref\_point) ▷ Get reference descriptor
5: for \( (i < \text{int}((\text{img}\_\text{current}.\text{shape}[0]-\text{size}_\text{window}[0])/\text{stride})) \) do ▷ Generate list of possible positions
6: for \( (j < \text{int}((\text{img}\_\text{current}.\text{shape}[1]-\text{size}_\text{window}[1])/\text{stride})) \) do
7: \( x = \text{int}(j+\text{ceil}(\text{size}_\text{window}[1]/2)) \)
8: \( y = \text{int}(i+\text{ceil}(\text{size}_\text{window}[0]/2)) \)
9: positions.append((y,x)) ▷ Get all possible positions where a \( 31 \times 31 \) window can be centered
10: end for
11: end for
12: ssrs = sum((\text{descriptors-ref}_\text{descriptor})^2,\text{axis}=1) ▷ Row-wise calculation of SSRs
13: for error\_unique in unique(sort(ssrs)) do ▷ Filter out points by their curvature
14: for candidate in \text{unique}(ssrs) do ▷ Get the 6 parameters of the surface fitted to the \( 3 \times 3 \) neighborhood around the candidate point. See section Subpixel level prediction.
15: \( c = \text{calculate}_\text{quadratic}_\text{surface}_\text{params}(\text{positions}[\text{candidate}], \text{img}_\text{current}) \)
16: if \( D > 0 \) and \( c[4] > 0 \) then ▷ Check if the fitted surface has a local minimum
17: \( \text{curvatures}.\text{append}(D) \)
18: candidates.\text{append}(\text{candidate})
19: end if
20: end for
21: end if
22: end for
23: if bool(curvatures) then ▷ Stop if a point fulfills the local minimum criteria
24: break
25: end if
26: end for
27: \( \text{pred}_{\text{pixel}} = \text{positions}[\text{candidates}[\text{argmax}(\text{curvatures})]] \) ▷ Predicted position with pixel accuracy
28: \( \text{pred} = \text{get}_\text{subpixel}_\text{level}_\text{pred}(\text{pred}_{\text{pixel}}, \text{img}_\text{current}) \) ▷ Predicted position with subpixel accuracy. See section Subpixel level prediction.

first two neighbors and the ground truth is smaller than 4.4 pixels. The largest spatial distance among the first two neighbors, shown for tracking the nose tip in bike conditions, is just 8 pixels. The largest distance of the first two neighbors is larger for the nose tip than for the face mole, reflecting the fact that it is a less distinct feature. For the static conditions, our method can achieve a lower nearest neighbor ratio than for the bike conditions. For the bike conditions, a large portion of the ratios is close to one. This makes the prediction of the location of the reference feature, characterised by a small spatial change leading to a small change in the feature descriptor, the ratio of all points surrounding the center point of the feature will be large. When tracking this feature, the large ratio increases the robustness because a small spatial error results from matching with one of the neighbouring pixels in some images. At the same time, the feature cannot be precisely located due to the low curvature of the local SSR landscape. Taken together these cases highlight the fact that sometimes a high ratio and sometimes a low ratio is desired. For DFE, we have observed local convexity of the SSR landscape around the skin features we have validated it on. Thus a moderate nearest neighbor to second nearest neighbor ratio is desired because moving just one pixel from the best match should give a fair match and a similar SSR at the same time a distinction to the best match is desired. This makes the prediction of the location of the reference feature, which at pixel-level is a result of a feature matching task work better using our subpixel level method.
Algorithm 2 Feature matching using DFE to locate the skin feature.

Require: ref_point, imgReference, imgCurrent, encoder

Ensure: pred

1: size_window = (31,31), stride = 1
2: refCrop = imgReference[ref_point[0]-size_window[0]:ref_point[0]+size_window[0], ref_point[1]-size_window[1]:ref_point[1]+size_window[1], :]
3: ref_descriptor = encoder.predict(imgReference, ref_point)
4: positions = [], crops = [], curvatures = [], candidates = []
5: for i<int((imgCurrent.shape[0]-size_window[0])/stride) do
6:     for j<int((imgCurrent.shape[1]-size_window[1])/stride) do
7:         x = int(j+ceil(size_window[1]/2))
8:         y = int(i+ceil(size_window[0]/2))
9:         crops.append(imgCurrent[i:i+size_window[0], j:j+size_window[1], :])
10:        positions.append((x,y))
11:    end for
12: end for
13: descriptors = encoder.predict(crops)
14: ssrs = sum((descriptors-ref_descriptor)**2, axis=1)
15: for error_unique in unique(sort(ssrs)) do
16:     for candidate in where(ssrs == error_unique) do
17:         c = calculate_quadratic_surface_params(positions[candidate], imgCurrent)
18:     c = calculate_quadratic_surface_params(positions[candidate], imgCurrent)
19:     D = 4c[5]c[4]+c[3]**2
20:     if D>0 and c[4] > 0 then
21:         curvatures.append(D)
22:     end if
23: end for
24: break
25: end if
26: end for
27: end if
28: end for
29: pred_pixel = positions[candidates[argmax(curvatures)]]
30: pred = get_subpixel_level_pred(pred_pixel, imgCurrent)

Algorithm 3 Location of the skin feature in the current frame using LK.

Require: ref_point, imgReference, imgCurrent

Ensure: pred

1: lucas_kanade_params = dict(winSize = (10,10), maxLevel = 4, criteria = (cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10, 0.03))
2: pred = cv2.calcOpticalFlowPyrLK(imgReference, imgCurrent, ref_point, None, **lucas_kanade_params)

A.12 Nearest neighbors within an acceptable distance threshold

We also investigated the sensitivity of the methods to the selection of nearest neighbors within the distance threshold defined by the 99% CI of the simulated Chi-square distribution. We refer here to the spatial distance (in pixels) between the ground truth and the nearest neighbor. For each frame, we investigated the nearest neighbor rank, sorting by the distance of the neighbors to the ground truth point in ascending order, of the first point to fall outside of the threshold for SIFT, SURF, and DFE. Figure 28 shows the number of nearest neighbors that fall within this spatial distance threshold for all conditions. Our method is superior to SIFT and SURF in all conditions, having more nearest neighbors close to the ground truth. This implies that even when our method localizes a point incorrectly, it still typically selects a point in the neighborhood of the ground truth. This implies that DFE is more precise since the ratio of the error to ground truth is the smallest.

A.13 Spatial SSR landscape

Similarly to Figures 12 and 30, we in Figure 29 observe that the SSR landscape of our method contains an organized structure, contrary to SIFT and SURF, which appear to have a random structure. We plotted the SSR as a function of spatial conditions of the image for the 15th largest error for track-
Figure 25: Distance between the ground truth and the original reference image and the distances of LK and DFE methods for tracking the nose tip under bike conditions. The correlation coefficient for the errors of LK and DFE are 0.52 and 0.33, respectively.

The generalisations to the PD data are seen in Figures 32 and 33 for the largest errors and smallest sum of localization errors, respectively.

A.14 Algorithm errors

The proportion of images for which the hypothesis that the error is due to human labelling of the ground truth can be rejected as a function of significance level reveals when the error is likely due to the algorithm. Figure 34 shows the proportion of images in each of the four validation cases below each possible significance threshold established by our Chi square analysis. The error in a large proportion of the images is unlikely to be due to human labelling of the ground truth, especially for SIFT and SURF, while it is likely to be due to human labelling for DFE. A low line at all points in these figures is a hallmark of the good performance of a method.
Figure 26: Distance of the second nearest neighbor as a function of the distance of the nearest neighbor for matching the face mole under static conditions (top-left), nose tip under static conditions (top-right), face mole under bike conditions (bottom-left) & nose tip under bike conditions (bottom-right). The descriptor distance of the nearest neighbors refers to the distance of the high dimensional representations between the reference point and the closest matching features in the image. The size of the markers is proportional to the largest spatial distance of the two neighbors to the ground truth. The red dashed line marks the $x = y$ line.
Figure 27: Distribution of the ratios of the nearest to the second nearest neighbor distances for matching the face mole under static conditions (top-left), nose tip under static conditions (top-right), face mole under bike conditions (bottom-left) & nose tip under bike conditions (bottom-right). The mean ratios are 0.9065, 0.8504, 0.9174, and 0.9718; respectively.
Figure 28: Number of nearest neighbors within the 99% CI for tracking the face mole under static conditions (top-left), nose tip under static conditions (top-right), face mole under bike conditions (bottom-left), and nose tip under bike conditions (bottom-right). The distances of the nearest neighbors are calculated at a pixel level resolution.
Figure 29: SSR of the frames with the 15th largest SSR value for tracking the face mole under bike conditions as a function of the spatial image coordinates for SIFT (left), SURF (middle), and DFE (right) and their corresponding regional enlargements of the optimum. The red dashed lines mark the point with minimum SSR. The distances to the ground truth are 95.65, 212.41, and 1.73 pixels, respectively. Both SIFT and SURF perform poorly while DFE performs well.

Figure 30: SSR as a function of the spatial image coordinates for the images with the largest localization errors for nose tips under bike conditions for SIFT (left), SURF (middle), and DFE (right) and the regional enlargements of the optimum. The red dashed lines mark the point in the space with minimum SSR, which is plotted as a red circle. The distances to the ground truth are 179.65, 355.45, and 6.48 pixels, respectively.
Figure 31: SSR as a function of the spatial image coordinates for the single image with the smallest sum of localization errors across SIFT (left), SURF (middle), and DFE (right) for tracking the face mole under bike conditions and their corresponding regional enlargements. The red dashed lines mark the point with minimum SSR. The distances to the ground truth are 0.89, 1.00, and 0.89 pixels, respectively.

Figure 32: SSR as a function of the spatial image coordinates for the images with the largest localization errors for tracking the moles in the hand of the PD patient for SIFT (left), SURF (middle), and DFE (right) and their corresponding regional enlargement of the optimum. The red dashed lines mark the point with minimum SSR. The distances to the ground truth are 219.85, 242.34, and 1.52 pixels, respectively.
Figure 33: SSR as a function of the spatial image coordinates for the single image with the smallest sum of localization errors for tracking the mole on the hand of the PD patient for SIFT (left), SURF (middle), and DFE (right) and their corresponding regional enlargements of the optimum. The red dashed lines mark the point with minimum SSR. The distances to the ground truth are 1.55, 196.12, and 1.41 pixels, respectively.
Figure 34: Proportion of images for which the hypothesis that the error is due to human labelling of the ground truth can be rejected at the significance level for tracking the face mole under static conditions (top-left), nose tip under static conditions (top-right), face mole under bike conditions (bottom-left), and nose tip under bike conditions (bottom-right).
References

Milton Abramowitz and Irene A Stegun. Handbook of mathematical functions with formulas, graphs, and mathematical tables, 1988.

Yassine Ahmine, Guillaume Caron, El Mustapha Mouaddib, and Fatima Chouireb. Adaptive Lucas-kanade tracking. *Image and Vis. Comput.*, 88:1–8, 2019.

Haldukn Akoglu. User’s guide to correlation coefficients. *Turkish Journal of Emergency Medicine*, 18(3):91–93, 2018.

Hesham A Alberry, Abdelzafah A Hegazy, and Gouda I Salama. A fast sift based method for copy move forgery detection. *Future Comput. and Inform. J.*, 3(2):159–165, 2018.

Matthew Anderson, Ricardo Motta, Srinivasan Chandrasekar, and Michael Stokes. Proposal for a standard default color space for the internet-srgb. In *Color and imaging conference*, volume 1996, pages 238–245. Society for Imaging Science and Technology, 1996.

Sadaf Ansari. A review on sift and surf for underwater image feature detection and matching. In *2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, pages 1–4. IEEE, 2019.

Akram Ashyani, Chi-Lun Lin, Esteban Roman, Ted Yeh, Tachyon Kuo, Wei-Fang Tsai, Yushan Lin, Ric Tu, Austin Su, Chien-Chih Wang, Chun-Hsiang Tan, and Torbjørn E Nordling. Digitization of updrs upper limb motor examinations towards automated quantification of symptoms of parkinson’s disease. *Manuscript in preparation*, 2022.

Simon Baker, Daniel Scharstein, JP Lewis, Stefan Roth, Michael J Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *Int. J. of Comput. Vis.*, 92(1):1–31, 2011.

Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

Jeffrey S Beis and David G Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Proceedings of IEEE computer society conference on computer vision and pattern recognition*, pages 1000–1006. IEEE, 1997.

Fangming Bi, Xin Ma, Wei Chen, Weidong Fang, Huayi Chen, Jingru Li, and Biruk Assefa. Review on video object tracking based on deep learning. *J. of New Media*, 1(2):63, 2019.

Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint*, 2020.

Jean-Yves Bouguet. Pyramidal implementation of the affine Lucas-kanade feature tracker description of the algorithm. *Intel Corporation*, 5(1-10):4, 2001.

Daniel J Butler, Jonas Wulff, Garrett B Stanley, and Michael J Black. A naturalistic open source movie for optical flow evaluation. In *European conference on computer vision*, pages 611–625. Springer, 2012.

Che-Han Chang, Chun-Nan Chou, and Edward Y Chang. Clkn: Cascaded Lucas-kanade networks for image alignment. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2213–2221, 2017.

Chien-Chang Chen, Wei-Yu Lu, and Chung-Hsuan Chou. Rotational copy-move forgery detection using sift and region growing strategies. *Multimed. Tools and Appl.*, 78(13):18293–18308, 2019.

Chun-Hong Cheng, Kwan-Long Wong, Jing-Wei Chin, Tsai-Tai Chan, and Richard HY So. Deep learning methods for remote heart rate measurement: a review and future research agenda. *Sensors*, 21(18):6296, 2021a.

Yihua Cheng, Haofei Wang, Yiwei Bao, and Feng Lu. Appearance-based gaze estimation with deep learning: A review and benchmark. *arXiv preprint arXiv:2104.12668*, 2021b.

Hsiang-Jen Chien, Chen-Chi Chuang, Chia-Yen Chen, and Reinhard Klette. When to use what feature? sift, surf, orb, or a-kaze features for monocular visual odometry. In *2016 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, pages 1–6. IEEE, 2016.

Lovish Chum, Anbumani Subramanian, Vineeth N Balasubramanian, and CV Jawahar. Beyond supervised learning: a computer vision perspective. *J. of the Indian Inst. of Sci.*, 99(2):177–199, 2019.

Joon Son Chung, Andrew Senior, Oriol Vinyals, and Andrew Zisserman. Lip reading sentences in the wild. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3444–3453. IEEE, 2017.

Gioele Ciaparrone, Francisco Luque Sánchez, Siham Tabik, Luigi Troiano, Roberto Tagliaferri, and Francisco Herrera. Deep learning in video multi-object tracking: A survey. *Neurocomputing*, 381:61–88, 2020.

Philippe Colantonio, Jean-Baptiste Thomas, and Alain Trémeau. Sampling cieLAB color space with perceptual metrics. *International Journal of Imaging and Robotics*, 16(3):1–22, 2016.

Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016.

Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015.

Youssef Douini, Jamal Riffi, Adnane Mohamed Mahraz, and Hamid Tairi. An image registration algorithm based on phase correlation and the classical lucas–kanade technique. *Signal, Image and Video Process.*, 11(7):1321–1328, 2017a.

Youssef Douini, Jamal Riffi, Mohamed Adnane Mahraz, and Hamid Tairi. Solving sub-pixel optical flow method. In *2017 Intelligent Systems and Computer Vision (ISCV)*, pages 1–5. IEEE, 2017b.

Iyad Abu Doush and AL-Btoush Sahar. Currency recognition using a smartphone: Comparison between color sift and gray scale sift algorithms. *J. of King Saud University-Computer and Inf. Sci.*, 29(4):484–492, 2017.

Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint*, 2018.

Guangyu Gao, Liling Liu, Li Wang, and Yihang Zhang. Fashion clothes matching scheme based on siamese network and autoencoder. *Multimed. Syst.*, 25(6):593–602, 2019.

I Garcia, Sebastian Bronte, Luis Miguel Bergasa, Javier Almazín, and J Yebes. Vision-based drowsiness detector for real driving conditions. In *2012 IEEE Intelligent Vehicles Symposium*, pages 618–623. IEEE, 2012.

Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.
Abdullah Ayub Khan, Asif Ali Laghari, and Shafique Ahmed. Convolutional autoencoder.

Borui Hou and Ruqiang Yan. Detecting objects as paired keypoints.

Vladimir A Kniaz, Oleg Vygolov, Vladimir V Kniaz, Yuriy Vizilter, Vladimir Gorbatshevich, Thomas Luhmann, and Niklas Conen. Deep learning of convolutional auto-encoder for image matching and 3d object reconstruction in the infrared range. In Proceedings of the IEEE International Conference on Computer Vision Workshops, pages 2155–2164, 2017.

H Law and J Deng. Detecting objects as paired keypoints. Lect. Notes in Comput. Sci., pages 765–781, 2018.

Yann LeCun, Koray Kavukcuoglu, and Clément Farabet. Convolutional networks and applications in vision. In Proceedings of 2010 IEEE international symposium on circuits and systems, pages 253–256. IEEE, 2010.

Woolee Lee, Jessica Jiwon Soong, Busra Ozlu, Dong Sup Shim, Azizbek Marakhimov, and Suan Lee. Biosignal sensors and deep learning-based speech recognition: A review. Sensors, 21(4):1399, 2021.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In European conference on computer vision, pages 740–755. Springer, 2014.

Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Saurabh Belongie. Feature pyramid networks for object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2117–2125, 2017a.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In Proceedings of the IEEE International conference on computer vision, pages 2980–2988, 2017b.

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In European conference on computer vision, pages 21–37. Springer, 2016.

David G. Lowe. Distinctive image features from scale-invariant keypoints. Int. J. of Comp. Vis., 60(2):91–110, 2004. ISSN 09252312.

Bruce Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In IJCAI, volume 81, 04 1981.

Ilja Manakov, Markus Rohm, and Volker Tresp. Walking the tightrope: an investigation of the convolutional autoencoder bottleneck. arXiv preprint, 2019.

Francesca Manni, Fons van der Sommen, Svitalana Zinger, Caifei Shan, Ronald Holthuijsen, Marco Lai, Gustav Buström, Richelle JM Hoveiling, Erik Edström, Adrian Elimi-Terander, et al. Hyperspectral imaging for skin feature detection: Advances in markerless tracking for spine surgery. Applied Sciences, 10(12):4078, 2020.

K McLaren. Xiii–the development of the cie 1976 (l* a* b*) uniform colour space and colour-difference formula. J. of the Soc. of Dyers and Colour., 92(9):338–341, 1976.

Krystian Mikołajczyk and Cordelia Schmid. A performance evaluation of local descriptors. IEEE transactions on pattern analysis and machine intelligence, 27(10):1615–1630, 2005.

Hartmut Neven, Geordie Rose, and William G Macready. Image recognition with an adiabatic quantum computer i. mapping to quadratic unconstrained binary optimization. arXiv preprint arXiv:0804.4457, 2008.

Aoxin Ni, Arian Azarang, and Nasser Kehtarnavaz. A review of deep learning-based contactless heart rate measurement methods. Sensors, 21(11):3719, 2021.

Hyeonwoo Noh, Andre Araujo, Jack Sim, Tobias Weyand, and Bohyung Han. Large-scale image retrieval with attentive deep local features. In Proceedings of the IEEE international conference on computer vision, pages 3456–3465, 2017.
Xiaorong Pu, Ke Fan, Xiong Chen, Loping Ji, and Zhihu Zhou. Facial expression recognition from image sequences using twofold random forest classifier. *Neurocomputing*, 168:1173–1180, 2015.

Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint*, 2018.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

Rasmus Rothe, Radu Timofte, and Luc Van Gool. Deep expectation of real and apparent age from a single image without facial landmarks. *Int. J. of Comp. Vis.*, 126(2-4):144–157, 2018.

Paul-Eduard Sarlin, Daniel DeTone, Tomaso Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4938–4947, 2020.

Gulbadan Sikander and Shahzad Anwar. Driver fatigue detection systems: A review. *IEEE Transactions on Intelligent Transportation Systems*, 20(6):2339–2352, 2018.

Edgar Simo-Serra, Eduard Trulls, Luis Ferraz, Iasonas Kokkinos, Pascal Fua, and Francesco Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *Proceedings of the IEEE international conference on computer vision*, pages 118–126, 2015.

Marzuraakah Mohd Stofa, Mohd Asyraf Zulkiflie, and Muhammad Annuirul Atiq Mohd Zainuri. Skin lesions classification and segmentation: A review. *International Journal of Advanced Computer Science and Applications*, 12(10), 2021.

Peifeng Su, Daizhi Liu, Xihai Li, and Zhigang Liu. A saliency-based hand selection approach for hyperspectral imagery inspired by scale selection. *IEEE Geoscience and Remote Sensing Letters*, 15(4):572–576, 2018.

Jiaming Sun, Zehong Shen, Yuan Wang, Hujun Bao, and Xiaowei Zhou. Loftx: Detector-free local feature matching with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8922–8931, 2021.

Christian Szegedy, Liu Wei, Jia Yangqing, Sermanet Pierre, Reed Scott, Anguelov Dragomir, Erhart Dumitru, Vanhoucke Vincent, and Rabinovich Andrew. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. *Popul. Health Manag.*, 18(3):186–191, 2015. ISSN 1942-7891. doi: 10.1089/pop.2014.0089. URL http://online.liebertpub.com/doi/10.1089/pop.2014.0089.

Shaharyar Ahmed Khan Tareen and Zahra Saleem. A comparative analysis of sift, surf, kaze, akaze, orb, and brisk. In 2018 *International conference on computing, mathematics and engineering technologies (iCoMET)*, pages 1–10. IEEE, 2018.

Jose Miguel Tarongi and Adriano Camps. Normality analysis for rfi detection in microwave radiometry. *Remote Sensing*, 2(1):191–210, 2010.

George Brinton Thomas and Ross L Finney. *Calculus And Analytic Geometry*. Addison-Wesley Publishing Company, 1900 E Lake Ave Glenview, IL 60025 United States, 1961.

Quoc-Viet Tran, Shun-Feng Su, and Van-Truong Nguyen. Pyramidal Lucas-kanade-based noncontact breath motion detection. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 50(7):2659–2670, 2018.

Marco C. Uchida, Renato Carvalho, Vitor Daniel Tessutti, Reury Frank Pereira Bacurau, Helio José Coelho-Júnior, Luciane Portas Capelo, Heloiza Prando Ramos, Marcia Calixto dos Santos, Luís Felipe Milano Teixeira, and Paulo Henrique Marchetti. Identification of muscle fatigue by tracking facial expressions. *PLoS ONE*, 13(12):1–11, 2018. ISSN 19326203.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

Chien-Chih Wang, Non-contact heart rate measurement based on facial videos. Master’s thesis, National Cheng Kung University, No. 1, Dasyue Rd, East District, Tainan City, 701, 2020.

Nannan Wang, Xinbo Gao, Dacheng Tao, Heng Yang, and Xuelong Li. Facial feature point detection: A comprehensive survey. *Neurocomputing*, 275:50–65, 2018. ISSN 18728286.

Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. Deepflow: Large displacement optical flow with deep matching. In *Proceedings of the IEEE international conference on computer vision*, pages 1385–1392, 2013.

Xue JiaoWang and Manoranjan Majji. Uncertainty quantification of lucas kanade feature track and application to visual odometry. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 950–958. IEEE, 2017.

Jure Zhontar and Yann LeCun. Stereo matching by training a convolutional neural network to compare image patches. *J. of Mach. Learn. Res.*, 17(1):2287–2318, 2016.

Matthew D Zeiler, Dilip Krishnan, Graham W Taylor, and Rob Fergus. Deconvolutional networks. In *2010 IEEE Computer Society Conference on computer vision and pattern recognition*, pages 2526–2533. IEEE, 2010.

Zhifei Zhang, Yang Song, and Hairong Qi. Age progression/regression by conditional adversarial autoencoder. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017.

Amy Zhao, Frédéric Durand, and John Guttag. Estimating a small signal in the presence of large noise. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2016-Febru, pages 671–676, 2016. ISBN 9781467383905.

Jianqing Zhao, Xiaohu Zhang, Chenxi Gao, Xialei Qu, Yongchao Tian, Yan Zhu, and Weixing Cao. Rapid mossicking of unmanned aerial vehicle (uav) images for crop growth monitoring using the sift algorithm. *Remote. Sens.*, 11(10):1226, 2019.

Liang Zheng, Yi Yang, and Qi Tian. Sift meets cnn: A decade of Mach. Learn. Res.*, 17(1):2287–2318, 2016.

Matthew D Zeiler, Dilip Krishnan, Graham W Taylor, and Rob Fergus. Deconvolutional networks. In *2010 IEEE Computer Society Conference on computer vision and pattern recognition*, pages 2526–2533. IEEE, 2010.

Zhifei Zhang, Yang Song, and Hairong Qi. Age progression/regression by conditional adversarial autoencoder. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017.

Amy Zhao, Frédéric Durand, and John Guttag. Estimating a small signal in the presence of large noise. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2016-Febru, pages 671–676, 2016. ISBN 9781467383905.

Jianqing Zhao, Xiaohu Zhang, Chenxi Gao, Xialei Qu, Yongchao Tian, Yan Zhu, and Weixing Cao. Rapid mossicking of unmanned aerial vehicle (uav) images for crop growth monitoring using the sift algorithm. *Remote. Sens.*, 11(10):1226, 2019.

Liang Zheng, Yi Yang, and Qi Tian. Sift meets cnn: A decade survey of instance retrieval. *IEEE transactions on pattern analysis and machine intelligence*, 40(5):1224–1244, 2017.

Yan-Tao Zheng, Ming Zhao, Yang Song, Hartwig Adam, Ulrich Buddemeier, Alessandro Biessacco, Fernando Brucher, Tat-Seng Chua, and Hartmut Neven. Tour the world: building a web-scale landmark recognition engine. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1085–1092. IEEE, 2009.
