Facial Image Feature Extraction Using Radial Basis Function Neural Network and Adaptive SIFT-SURF Algorithm

Vincent A. Akpan[1,†] and Joshua B. Agbogun[2]

[1]Department of Biomedical Technology,
The Federal University of Technology,
P.M.B. 704 Akure, Ondo State,
Nigeria.
[2]Department of Computer Science and Mathematics,
Godfrey Okoye University,
P.M.B. 01024, Enugu, Enugu State,
Nigeria
[†]Corresponding Author's Email: vaakpan@futa.edu.ng

ABSTRACT

This paper presents a novel facial image feature extraction technique using radial basis function neural network (RBFNN) and adaptive SIFT-SURF algorithm specifically for facial image feature extraction. The RBFNN adaptive SIFT-SURF algorithm is formulated as a derivative-based optimization problem which is trained using adaptive recursive least squares (ARLS) algorithm. The MATLAB interface for the implementation and evaluation of the proposed RBFNN adaptive SIFT-SURF algorithm for facial image feature extraction is also presented. A total of 1500 facial image samples were taken under different light intensities and postures to investigate the efficiency of the proposed algorithm. The performance of the proposed RBFNN adaptive SIFT-SURF algorithm has been compared with the standard and well established SIFT algorithm on the 1500 facial image samples. The simulation results show the superior performance of the proposed RBFNN adaptive SIFT-SURF algorithm when compared to the standard SIFT algorithm in terms of several evaluation parameters especially with the lowest computation time and a large number of descriptors. The efficiency of the proposed RBFNN adaptive SIFT-SURF algorithm indicates that it can be deployed for the design of real-time feature extract systems for facial images, signatures, thumbprints with so on.

Keywords: Facial Images, Feature Extraction, Radial Basis Function Neural Network (RBFNN), Scale-Invariant Feature Transform (SIFT), Speed-Up Robust Feature (SURF)

African Journal of Computing & ICT Reference Format

Vincent A. Akpan and Joshua B. Agbogun (2021), Facial Image Feature Extraction Using Radial Basis Function Neural Network and Adaptive SIFT-SURF Algorithm,
Afr. J. Comp. & ICT, Vol. 14, No. 2, pp. 13 –.28

1. INTRODUCTION

Features can be seen as the interesting part of an image [1, 2]. In particular, human faces are complex objects with features that can vary over time. Moreover, we humans have a neural ability to recognize faces and identify persons at a glance. In pattern recognition and image processing, feature extraction is a special form of dimensionality reduction. For any object in an image, interesting points on the object can be extracted to provide what is known as the "feature description" of the object. This description, extracted from a training image, can then be used to identify the object when attempting to locate the object in a test image containing many other objects. When the input data to an algorithm is too large to be processed and it is suspected to be notoriously redundant (e.g. the same measurement in both feet and meters) then the input data will be transformed into a reduced representation set of features (also named features vector) [3, 4].

Transforming the input data into the set of features is called *feature extraction*. If the features extracted are carefully chosen, it is expected that the features set will extract the relevant information from the input data to perform the desired task using this reduced representation instead of the full size input; this is the reason why feature extraction is needed. To perform an efficient image recognition, it is important that the features extracted from the training image possess the following essential properties [5, 6]:

1). Identifiability: This simply means that features extracted must be invariant to translation and rotation i.e. the location, rotation and scale changes in the shape of the image must not affect the extracted features.
2). Noise Resistance: This property implies that the features must be strong and unaffected by noise, i.e. the features must be independent of the noise in a given range that affects the pattern of the image.
3). Reliability: As long as the pattern in an image is the same, the extracted features must also be the same.
4). Affine Transform Invariance: Affine transformation is a transformation that preserves the relation of parallelism between lines. It transforms a linear mapping from 2-Dimensions (2-D) coordinates to other 2-D coordinates such that the "straightness" and "parallelism" of lines are maintained. It can be constructed with a sequence of translations, scales, flips, rotations and shears. The extracted features must be the same as possible with affine transforms.
5). Statistically Independent: Two features must be statistically independent. This represents the compactness of the representation; and
6). Occultation Invariance: when some parts of a shape are occulted by other objects, the features of the remaining part must not change compared to the original shape.

Feature extraction involves simplifying the number of resources required to describe a large set of data accurately. When performing analysis of complex data, one of the major problems stems from the number of variables involved [7, 8]. Analysis with a large number of variables generally requires a large amount of memory and computation power or a classification algorithm which overfits the training sample and generalizes poorly to new samples. Feature extraction is a general term for methods of constructing combinations of the variables to get around these problems while still describing the data with sufficient accuracy. Best results are achieved when an expert constructs a set of application-dependent features. Nevertheless, if no such expert knowledge is available general dimensionality reduction techniques may help using principal component analysis (PCA) or semi definite embedding multifactor dimensionality reduction.

Throughout our life time, many faces are seen and stored naturally in our memories forming a kind of database [5, 8]. Machine recognition of faces requires also a database which is usually built using facial images, where sometimes different face images of one person are included to account for variations in facial features. The development of an intelligent face extraction system requires providing sufficient information and meaningful data during machine learning of a face.

Many facial image feature extraction algorithms have been proposed. They can be mainly classified into three categories, namely: feature based [9], appearance based [10, 11] and template based [12] approaches. Each has its advantages and limitations on its own right with respect to locating the face region, computing the cost functions and extracting the features. However, all approaches demand a fairly sophisticated model, which is often not readily available. Facial image feature extraction using genetic algorithm has been proposed in Yen and Nithianandan [13] but with heavy computational overhead which may limit its deployment for real-time applications.

The main idea for finding features includes two things, namely: identification and verification [14]. Verification means to check whether the person is authenticated to use the service for which he is claiming and identification means to find out the identity of an individual by comparing his/her face with a database of images of individuals. The related work done in this area includes: geometric feature based methods [15, 16], template based methods [17], correlation based methods, support vector machine (SVM) approach [18–20], feature based methods [14], and holistic based methods [14].

Furthermore, existing literature clusters the present techniques into four broad groups where the extractions are based on geometric feature [21], template, color segmentation and appearance based approaches [22]. Geometric feature based extraction is carried out by using relative positions and sizes of the important components of the face such as eyes, nose, mouth, etc. Valley detection filters and analysis of horizontal and vertical edge integral projections are such examples. Template based approaches focus on a template function and an energy function. The best match of a template in a facial image corresponds to minimum energy. For example, deformable templates [23] and genetic algorithms [13]. Color segmentation [24, 25] based feature extraction uses skin color to isolate the face. Any non-skin color region within the face region is viewed as a candidate for "eyes" and/or "mouth" etc. Finally the appearance based schemes [26] aims to find basis vectors to represent the face using linear transformation and statistical methods.

The scale-invariant feature transform (or SIFT) is an algorithm in computer vision to detect and describe local features in images [8]. The SIFT algorithm was first proposed and published by David Lowe in 2004 [27]. The algorithm is applied in object recognition, robotic mapping and navigation, image stitching, 3-D modeling, gesture recognition, video tracking, individual identification of wildlife and match moving. The SIFT algorithm is robust for identifying stable key locations in the scale-space of a grey scale image. The SIFT algorithm uses the following four steps to extract the set of descriptors from a given image: *1).* scale-Space extrema detection; *2).* accurate key point localization; *3).* orientation assignment; and *4).* keypoint description.

SURF is an acronym for Speeded-Up Robust Features. It is also another robust local feature detector which was first presented by Herbert Bay and co-workers in 2008 [28]. It is a performant scale- and rotation-invariant interest point detector and descriptor and can be used in computer vision tasks like object recognition or 3-D reconstruction. It approximates or even outperforms previously proposed schemes with respect to repeatability, distinctiveness, and robustness, yet can be computed and compared much faster than SIFT. This is achieved by: *1).* relying on integral images for image convolution; *2).* building on the strengths of the leading existing detectors and descriptors (using Hessian matrix based measure for the detector, and a distribution-based descriptor); and *3).* simplifying these methods to the basic building blocks or basic concepts. This leads to a combination of novel detection, description and matching steps. SIFT features have been used in pattern recognition and classification, mostly in object recognition. The SURF algorithm is used for face authentication using frontal view templates and evaluated for recognition of graffiti tags in both with good results.

Here, we shall be looking at a special algorithms required in the description of the various acquired extracted features from images; which are the SIFT and SURFF as well as the proposed RBFNN adaptive SIFT-SURF. The paper is organized as follows. The proposed RBFNN adaptive SIFT-SURF algorithm is presented in Section 2. The strategy for the implementation of the proposed RBFNN adaptive SIFT-SURF is described in Section 3 together with the materials, methods as well as the results and its discussion. Section 4 concludes the paper with some highlights on future directions as part of recommendations.

2. LITERATURE REVIEW

2.1 Architecture of RBFNN Framework

Radial basis function neural network (RBFNN) is trained to perform a mapping from an *m*-dimensional input space to an *n*-dimensional output space. RBFNN can be used for discrete pattern classification, function approximation, signal processing, control, and/or other applications which require a mapping from an input space to an output space. A RBFNN having the typical structure illustrated in Figure 1 has been proven to be a universal function approximator and an alternative to multilayer perceptron (MLP) feedforward neural network (FNN) [29]. The RBFNN is a multidimensional nonlinear function that maps the inputs to the outputs depending on the distance between the input vector and the center vector. Therefore, the RBFNN can perform similar function mappings as the MLP FNN but its architecture and functionality are very different.

As shown in Fig. 1, the RBFNN consists of the *m*-dimensional input *x* being passed directly to a hidden layer. Suppose there are *c* neurons in the hidden layer. Each of the *c* neurons in the hidden layer applies an activation function which is a function of the Euclidean distance between the input and an *m*-dimensional prototype vector. Each hidden neuron contains its prototype vector as a parameter. The output of each hidden neuron is then weighted and passed to the output layer. The outputs of the network consist of sums of the weighted hidden layer neurons. It can be seen from Figure 1 that the design of an RBFNN requires several decisions, including but not limited to the following [30]

1). How many hidden units will reside in the hidden layer (i.e., what is the value of the integer *c*);
2). What are the values of the prototypes (i.e., what are the values of the *v* vectors);
3). What function will be used at the hidden units (i.e., what is the function $g(\cdot)$); and
4). What weights will be applied between the hidden layer and the output layer (i.e., what are the values of the w weights).

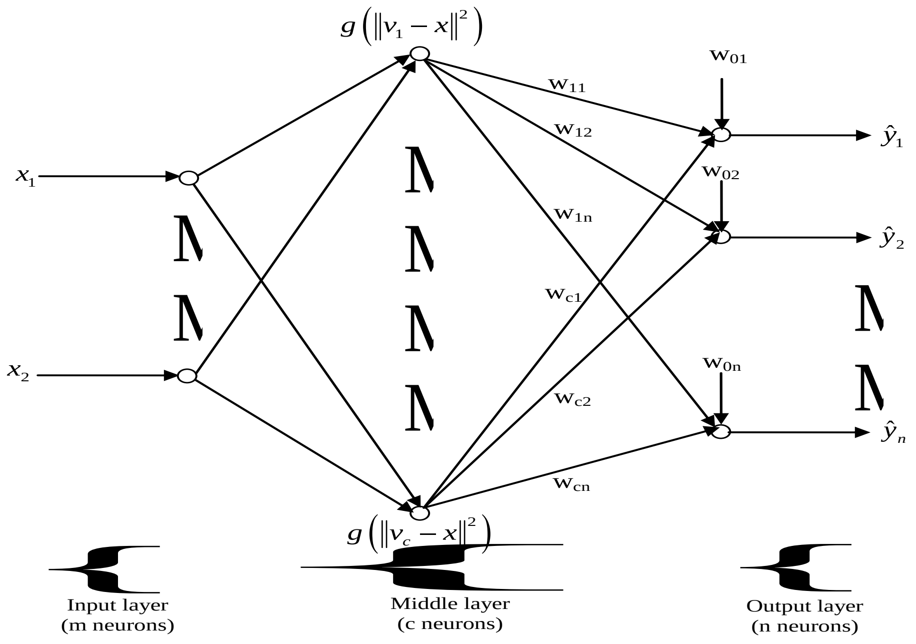


Fig. 1: Architecture of the radial basis function neural network (RBFNN).

The performance of an RBFNN network depends on the number and location (in the input space) of the centers, the shape of the RBF functions at the hidden units, and the method used for determining the network weights. Some researchers have trained RBFNN networks by selecting the centers randomly from the training data [31]. Others have used unsupervised procedures (such as the k-means algorithm) for selecting the RBFNN centers [32]. Still others have used supervised procedures for selecting the RBFNN centers [33].

Several training methods separate the tasks of prototype determination and weight optimization. This trend probably arose because of the quick training that could result from the separation of the two tasks. One of the primary contributors to the popularity of RBFNNs was probably their fast training times as compared to gradient descent training (include backpropagation). Furthermore, it can be seen in Figure 1 that once the prototypes are fixed and the hidden layer function $g(g)$ is known, the network is linear in the weight parameters *w*. At that point training the network becomes a quick and easy task that can be solved via linear least squares. This is similar to the popularity of the optimal interpolative net that is due in large part to the efficient non-iterative learning algorithms that are available [34, 35].

Training methods that separate the tasks of prototype determination and weight optimization often do not use the input–output data from the training set for the selection of the prototypes. For instance, the random selection method and the k-means algorithm result in prototypes that are completely independent of the input–output data from the training set. Although this results in fast training, it clearly does not take full advantage of the information contained in the training set.

Gradient descent training of RBFNNs has proven to be much more effective than more conventional methods.[35] However, gradient descent training can be computationally expensive and the global minimum may never be reached. Rather than training the RBFNN with extended Kalman filter as proposed in Simon[32] which introduces additional computational burden; this paper extends the results of [36, 37] to formulate a training method for the proposed RBFNN adaptive SIFT-SURF algorithm based on adaptive recursive least squares (ARLS) algorithm which have been proven to be suitable for online applications as shown in Akpan [36] as well as Akpan and Hassapis [37].

2.2 Mathematical Formulation of the RBFNN Adaptive SIFT-SURF Algorithm

There have been a number of popular choices for the function $g(g)$ at the hidden layer of RBFNNs as evident in Figure 1 [38, 39]. The most common choice is a Gaussian function of the form [30, 38, 39]:

$$g(v) = \exp\left(-\frac{v}{\beta^2}\right) \quad (1)$$

where β is a real constant. Other hidden layer functions that have often been used are the thin plate spline function given by

$$g(v) = v \log \sqrt{v} \quad (2)$$

the multiquadric function

$$g(v) = \left(v^2 + \beta^2\right)^{1/2} \quad (3)$$

and the inverse multiquadric function

$$g(v) = \left(v^2 + \beta^2\right)^{-1/2} \quad (4)$$

It has been proposed that since RBFNN prototypes are generally interpreted as the centers o receptive fields, hidden layer functions should have the following properties [33, 39]:

1). The response at a hidden neuron is always positive;
2). The response at a hidden neuron becomes stronger as the input approaches the prototype;
3). The response at a hidden neuron becomes more sensitive to the input as the input approaches the prototype.

With these desired properties in mind, an RBFNN's hidden layer function should be of the general form

$$g(v) = \left[g_0(v)\right]^{\frac{1}{(1-p)}} \quad (5)$$

and one of two sets of conditions on the so-called generator function $g_0(v)$ should hold. If p is a real number greater than 1, then the generator function $g_0(v)$ should satisfy the following set of conditions:

1). $g_0(v) > 0, \quad \forall v \in (0, \infty);$
2). $g_0'(v) > 0, \quad \forall v \in (0, \infty);$
3). $\frac{p}{p-1}\left[g_0'(v)\right]^2 - g_0(v) g_0'' > 0, \quad \forall v \in (0, \infty).$

The last two conditions ensure that $g'(v) < 0$ and $g''(v) > 0, \ \forall v \in (0, \infty)$. If p is a real number less than 1, then the generator function $g_0(v)$ should satisfy the following set of conditions:

1). $g_0(v) > 0, \quad \forall v \in (0, \infty);$
2). $g_0'(v) < 0, \quad \forall v \in (0, \infty);$
3). $\frac{p}{p-1}\left[g_0'(v)\right]^2 - g_0(v) g_0'' < 0, \quad \forall v \in (0, \infty).$

The last two conditions again ensure that $g'(v) < 0$ and $g''(v) > 0, \ \forall v \in (0, \infty)$. It can be shown that the functions of (1) and (4) satisfy these conditions, but the function of (2) and (3) do not. One new generator function that satisfies the above conditions is the linear function [33] given by:

$$g_0(v) = av + b \quad (6)$$

where $a > b$ and $b \geq 0$. If $a = 1$ and $p = 3$, then the hidden layer function reduces to the inverse multiquadric function of (4). Similar to work of Simon [30], in this paper, we concentrate on the hidden layer functions of the form of (5) with special emphasis on the inverse multiquadric function of (4).

The predicted output response $(\hat{y})$ of the RBFNN of the form of Figure 1, where the hidden layer functions $g(\cdot)$ have the form of (5), can be written as

$$\hat{y} = \begin{bmatrix} w_{10} & w_{11} & \cdots & w_{1c} \\ w_{20} & w_{21} & \cdots & w_{2c} \\ \vdots & \vdots & \vdots & \vdots \\ w_{n0} & w_{n1} & \cdots & w_{nc} \end{bmatrix} \begin{bmatrix} 1 \\ g\left(\|x - v_c\|^2\right) \\ \vdots \\ g\left(\|x - v_c\|^2\right) \end{bmatrix} \quad (7)$$

We will use the following notation as shorthand for the weight matrix on the right-hand side of (7)

$$\hat{y} = \begin{bmatrix} w_{10} & w_{11} & \cdots & w_{1c} \\ w_{20} & w_{21} & \cdots & w_{2c} \\ \vdots & \vdots & \vdots & \vdots \\ w_{n0} & w_{n1} & \cdots & w_{nc} \end{bmatrix} = \begin{bmatrix} w_1^T \\ w_2^T \\ \vdots \\ w_n^T \end{bmatrix} = W \quad (8)$$

If we are given a training set of M desired input–output responses $\{x_i, y_i\}$ $(i, -1, \ldots, M)$, then we can augment M equations of the form of (7) as follows:

$$\left[\hat{y}_1, \ldots, \hat{y}_M\right] = W \begin{bmatrix} 1 & \cdots & 1 \\ g\left(\|x_1 - v_1\|^2\right) & \cdots & g\left(\|x_M - v_1\|^2\right) \\ \vdots & \vdots & \vdots \\ g\left(\|x_1 - v_c\|^2\right) & \cdots & g\left(\|x_M - v_c\|^2\right) \end{bmatrix} \quad (9)$$

We will introduce the notation for the matrix on the right-hand side of (9) as:

$$h_{0k} = 1, \qquad (k = 1, \ldots, M) \quad (10)$$

$$\left.\begin{array}{l} h_{jk} = g\left(\|x_k - v_j\|^2\right), \\ \qquad (k = 1, \ldots, M), \ (j = 1, \ldots, c) \end{array}\right\} \quad (11)$$

in which case we can write the on the right-hand side of (9) as

$$\begin{bmatrix} h_{01} & \cdots & h_{0M} \\ h_{11} & \cdots & h_{1M} \\ \vdots & \vdots & \vdots \\ h_{c1} & \cdots & h_{cM} \end{bmatrix} = \left[h_1 \ldots, h_M\right] = H \quad (12)$$

In this case we can write (9) as

$$\hat{Y} = WH \quad (13)$$

Now, if we want to use gradient descent to minimize the training error, we can define the error function

$$E = \frac{1}{2}\left\| Y - \hat{Y} \right\|_F^2 \qquad (14)$$

where Y is the matrix of target (desired) values for the RBFNN output, and $\left\| g \right\|_F^2$ is the square of the Froebinius norm of a matrix, which is equal to the sum of the squares of the elements of the matrix. It has been shown in Simon[35] that in this case

$$\frac{\partial E}{\partial w_i} = \sum_{k=1}^{M} \left(\hat{y}_{ik} - y_{ik} \right) h_k, \qquad \left(i = 1, \mathrm{K}\ , n \right) \qquad (15)$$

$$\left.\begin{aligned} \frac{\partial E}{\partial v_j} &= \sum_{k=1}^{M} 2g'\left(\left\| x_k - v_j \right\|^2 \right)\left(x_k - v_j \right) \\ &\sum_{k=1}^{M} \left(y_{ik} - \hat{y}_{ik} \right) w_{ij}, \qquad \left(j = 1, \mathrm{K}\ , c \right) \end{aligned}\right\} \qquad (16)$$

where $\hat{y}_{ik}$ is the element in the *i*th row and *k*th column of the $\hat{Y}$ matrix of (13), and y_{ik} is the corresponding element in the *Y* matrix. Now we can optimize the RBFNN with respect to the rows of the weight matrix *W* and the prototype locations v_j by iteratively computing the above partials and performing the following updates:

$$\left.\begin{aligned} w_i &= w_i - \eta \frac{\partial E}{\partial w_i}, \qquad \left(i = 1, \mathrm{K}\ , n \right) \\ v_j &= v_j - \eta \frac{\partial E}{\partial v_j}, \qquad \left(j = 1, \mathrm{K}\ , c \right) \end{aligned}\right\} \qquad (17)$$

where η is the step size of the gradient descent method. This optimization stops when w_i and v_j reach local minima.

2.3 The Strategic Implementation of the RBFNN Adaptive SIFT-SURF Algorithm

1). Integral image computation and extraction.
2). Computation of the Hessian matrix with guaranteed positive definiteness of the Hessian matrix.
3). Construction of the SURF descriptor.
4). Data structure and analysis: *a)* Hessian data and *b)* interest point data.
5). Apply the neuroscaling PPCA-based radial basis function neural network to compute the optimal features according to the following sequence:
  i). Re-evaluate the Hessian.
  ii). Compute the network output data $\hat{\mathbf{y}}_i$ and match the result with the actual image data $\mathbf{y}_i$.
  iii). Minimize the error according (18) as follows:

$$\hat{\mathbf{y}}_i = \mathbf{y}_i - \eta \frac{\partial E}{\partial \mathbf{y}_i} \qquad (18)$$

  iv). The network is trained with the adaptive recursive least squares (ARLS) algorithm described in Akpan [36] with some concepts from Akpan and Hassapis [37].
6). Apply the MATLAB's "*parfor*" to reduce computational load.

2.4 Summary of the proposed RBFNN Adaptive SIFT-SURF Algorithm Implementation

The SURF algorithm detects landmark points in an image and describe the points by a vector which is robust against (a little bit) rotation, scaling and noise. It can be used in the same way as the SIFT algorithm to align (register) two or more images based on corresponding points, or make 3-D reconstructions. The detailed implementation of the proposed RBFNN adaptive SIFT-SURF algorithms are given as in the following sequence.

1). Computation of the Hessian matrix;
2). Building the Hessian matrix of the output layer. Here, we implements a low-pass digital filter, normalizes the resulting filter output and decomposes the components into their respective x-, y- and z-axes. Finally, the Hessian response is computed and the sign of the Laplacian is determined which must be greater than or equal to zero. The sequence is implemented as follows:
  i). get the image coordinates;
  ii). compute response components;
  iii). normalize the filter responses with respect to their size; and
  iv). get the determinant of hessian response & Laplacian sign.
3). Building the Hessian of the hidden layer:
  i). Calculate responses for the first 4 Hessian data octaves;
  ii). Deallocate memory and clear any existing response layers;
  iii). Get image attributes;
  iv). Calculate approximated determinant of hessian values; and
  v). Extract responses from the image.
4). Extraction of the interior points as follows:
  (i) filter index map;
  (ii) Build the response map;
  (iii) Find the maxima across scale and space;
  (iv) loop over middle response layer at density of the most sparse layer (always top) to find the maxima across scale and space;
  (v) Show Laplacian and response maps with found interest-points;
  (vi) Show the response map; and
  (vii) Show the maximum points.
5). Computation of the Laplacian.
6). Computation of the *priori* outputs response.
7). Extrapolating for the extremum in the following sequence:
  (i) get the offsets from the interpolation;
  (ii) get the step distance between filters; and
  (iii) If point is sufficiently close to the actual extremum,

scale the system and compute the Laplacian.

8). Validation of the extremum of the *posteriori* Hessian estimate:
- (i) bounds check;
- (ii) check the candidate point in the middle layer is above thresh ;
- (iii) Check if any response in 3x3x3 is greater than the candidate is not a maximum otherwise Clamp to boundary; and
- (iv) Note that the original SURF doesn't contain this boundary clamp because if you process one coordinate at the time you already returned on the boundary check.

9). Extraction of the output response using the Hessian data results.

10). Image integration using the *Box Integral* algorithm summarized below:
- (i) Get integer coordinates;
- (ii) Get the corner coordinates of the box integral;
- (iii) Get the values at the corners of the box integral (fast 1-D index look-up-table);
- (iv) If the coordinates are outside at the top or left, the value must be zero; and
- (v) Minimum value of the integral is zero.

11). Decomposition of image integrand into the x-component based on the *Box Integralalgorithm* and the *Haar formulation*.

12). Decomposition of image integrand into the y-component based on the *Box Integralalgorithm* and the *Haar formulation*.

13). Cumulative sum by double integration using integration algorithm:
- (i) Convert Image to double;
- (ii) Convert Image to grayscale; and
- (iii) Make the integral image.

14). MATLAB® Implementation of the proposed RBFNN adaptive SIFT-SURF algorithm. First, the following six options must be specified as stated below:

```
% For testing the system: Initialize the
% RBFNN adaptive SIFT-SURF algorithm
% parameters
  Options.verbose     = false;
% If set to true then useful information
% is displayed (default false)
  Options.upright     = true;
% Boolean which determines if we want
% non-rotation invariant result (default
% false)
 Options.extended     = false;
% Add extra landmark point information
% to the descriptor (default false)
  Options.tresh       = 0.0001;
% Hessian response threshold (default
% 0.0002)
  Options.octaves     = 5;
% Number of octaves to analyze(default 5)
  Options.init_sample = 2;
% Initial sampling step in the image
% (default 2)
```

Process inputs are constructed based on the options specified above which performs the following three functions, namely: *1).* Create Integral Image; *2).* Extract the interest points; *3).* Describe the interest points. The outputs of the RBFNN adaptive SIFT-SURF algorithm are collected as a structure with the information about all detected Landmark points as listed below:

```
% The output is a structure with the
% information about all detected
Landmark % points:
  num_x         = Ipts1.x
  % The landmark position on the x-axis
  num_y         = Ipts1.y
  % The landmark position on the y-axis
  num_scale     = Ipts1.scale
  % The scale of the detected landmark
  num_laplacian = Ipts1.laplacian
% The laplacian of the landmark
  neighborhood
  num_orient    = Ipts1.orientation
% Orientation in radians
  num_descript  = Ipts1.descriptor;
% The descriptor for corresponding point
  matching
```

15). Create possible regions of the interest points using the descriptor algorithm
- (i) Display only information about the first 50 points;
- (ii) Determine descriptor size;
- (iii) Get the orientation; and
- (iv) Extract the RBFNN adaptive SIFT-SURF descriptor.

16). Obtain the complete description of the point as follows:
- (i) Get rounded *InterestPoint* data;
- (ii) Basis coordinates of samples, if coordinate 0, 0, and scale 1;
- (iii) 2-D matrices instead of double for-loops, i_l, j_l;
- (iv) Calculate descriptor for this interest point;
- (v) Coordinates of samples (not rotated);
- (vi) Get coordinates of sample point on the rotated axis;
- (vii) Get the Gaussian weighted x and y responses;
- (viii) Get the Gaussian weighted x and y responses on the aligned axis;
- (ix) Get the Gaussian scaling;
- (x) split x responses for different signs of y;
- (xi) Split y responses for different signs of x; and
- (xii) Convert to unit vector.

(a). The inputs to this algorithm are:

- (i) ip: Interest Point (x, y, scale, orientation);
- (ii) bUpright: If true not rotation invariant descriptor;
- (iii) bExtended If true make a 128 values descriptor;
- (iv) iimg: Integral image; and
- (v) verbose: If true show additional information.

(b). The output from this implementation is:

- (i) descriptor: Descriptor of interest point length 64 or 128 (extended).

17). Obtain the orientation for the prescribed image points as follows:

- (i) Get rounded *InterestPoint* data;

Table 1: MATLAB® implementation and evaluation of the proposed RBFNN adaptive SIFT-SURF algorithm for facial image feature extraction

```
% 1. === Compare SIFT and RBFNN Adaptive SIFT_SURF implementation and computational analysis ===
  clear all;close all;clc;
  warning off
  in_i = input('Please enter between 1 & 150 corresponding to the folder           =   ');
  in_j = input('Please enter between 1 & 10 corresponding to the actual Face       =   ');
  in_k = input('Please enter between 1 & 150 corresponding to the folder           =   ');
  in_t = input('Please enter between 1 & 10 same or different from the actual face =   ');

 % 2. ======== Load images for the SIFT implementation ================
  sift_comp_start = cputime;
 % Extract features from the original image
  [image,  descrips,  locs]  =  original_sift(strcat('F',num2str(in_i),'/R',num2str(in_i),  '_','Face',
                          num2str(in_j),'.pgm'));
  showkeys(image, locs);
  title('Features  Identified  in  the  Original  Image','FontWeight','bold',  'Fontsize',16,  'color',
        'red')

 % 3. ===== Implement the original SIFT feature extraction and matching algorithm =====
  image_run  = strcat('F',num2str(in_i),'/R',num2str(in_i),'_','Face',num2str(in_j),'.pgm');
 % For building the system
  image_test = strcat('F',num2str(in_k),'/R',num2str(in_k),'_','Face',num2str(in_t),'.pgm');
 % For testing the system
  match(image_run,image_test)
    title('SIFT  Feature  Extraction  and  Matching  Algorithm',  'FontWeight',  'bold',  'Fontsize',16,
        'color','red')
  sift_comp_time = cputime - sift_comp_start

 % 4. ======== Load images for the Adaptive SIFT-SURF implementation ==========
  adapt_comp_start = cputime;
  I1=imread(strcat('F',num2str(in_i),'/R',num2str(in_i),'_','Face',num2str(in_j),'.png'));
 % 5. ====== For Building the System ===============
  I2=imread(strcat('F',num2str(in_k),'/R',num2str(in_k),'_','Face',num2str(in_t),'.png'));

 % 6. === For testing the system: Initialize the RBFNN adaptive SIFT-SURF algorithm parameters ==
  Options.verbose     = false;  % If set to true then useful
 % information is displayed (default false)
  Options.upright     = true;    % Boolean which determines if we want non-rotation
 % invariant result (default false)
   Options.extended    = false;  % Add extra landmark point information to the
 % descriptor (default false)
  Options.tresh       = 0.0001; % Hessian response threshold (default 0.0002)
  Options.octaves     = 5;      % Number of octaves to analyze(default 5)
  Options.init_sample = 2;      % Initial sampling step in the image (default 2)

  script_rbfnn_adapt_sift_surf;
  title('The  RBFNN  Adaptive  SIFT-SURF  Feature  Extraction  and  Matching  Algorithm','FontWeight',
        'bold','Fontsize',16,'color','red')

  adapt_normal_time = (cputime - adapt_comp_start)

 % 7. ==== The output is a structure with the information about all detected Landmark points and plot
        the corresponding descriptors from the Standard SIFT and RBFNN adaptive SIFT-SURF algorithms
        ====
     num_x         = Ipts1.x            % The landmark position on the x-axis
     num_y         = Ipts1.y            % The landmark position on the y-axis
     num_scale     = Ipts1.scale        % The scale of the detected landmark
     num_laplacian = Ipts1.laplacian    % The laplacian of the landmark neighborhood
```

```
      num_orient    = Ipts1.orientation % Orientation in radians
      num_descript  = Ipts1.descriptor;  % The descriptor for corresponding point matching
      adapt_normal_time = (cputime - adapt_comp_start)

 % 8. ==== Extraction of the six parameters in Step 7 above and plotting the respective descriptors
          for the Standard SIFT and the proposed RBFNN adaptive SIFT-SURF algorithms ====
    info_detected_image;
```

(ii) Calculate *Haar* responses for points within radius of 6*scale;
(iii) Get Gaussian filter (by mirroring gauss25);
(iv) loop slides pi/3 window around feature point;
(v) Repmat is used to check for all angles (x-direction) and all responses (y-direction) without for-loops;
(vi) Determine whether the point is within the window; and
(vii) Find the most dominant direction.
(a). The inputs to this algorithm are:
(i) ip: InterestPoint data, (x,y,scale)
(ii) img: Integral Image
(iii) verbose: If true, show additional information
(b). The output from this implementation is:
(i) Orientation: Orientation of interest point (radians)

18). Display results with mappings and feature extractions compared with original image.

3. MATERIALS

The materials employed in this study are one thousand five hundred (1,500) joint photographic expert group (JPG) face image samples taken with a NOKIA 2700C, 2.0 mega pixel digital camera phone. The 1,500 facial images consist of 10 different face samples of the different individuals was taken at different locations, different posture and different light intensities to investigate the efficiency of the proposed algorithm. The proposed evaluation algorithm in this study are written, compiled and analyzed using MATLAB R2021a, from The MathWorks Inc. [40], software running on a Pentium® Dual-Core CPU T4500 @ 2.30 GHz computer on windows® 8 platform.

The RBFNN adaptive SIFT-SURF algorithm simulated in the MATLAB environment was used in the identification and features extraction from the original image, SIFT feature extraction and matching of images and the RBFNN adaptive SIFT-SURF feature extraction and matching of images are shown in Table 1.

The program of Table 1 is then simulated in the Editor template on MATLAB and MATLAB requests for the following values on the command prompt:

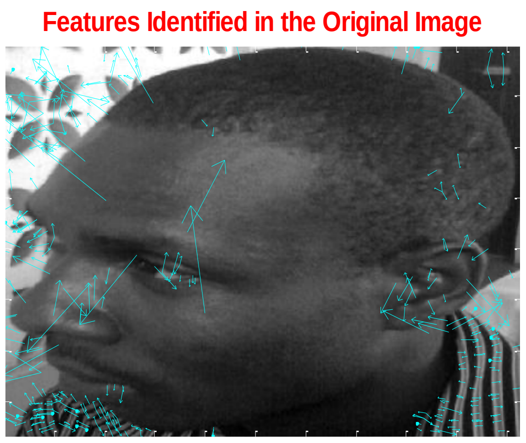
(a) Please enter between 1 & 150 corresponding to the folder =
(b) Please enter between 1 & 10 corresponding to the actual Face =
(c) Please enter between 1 & 150 corresponding to the folder =
(d) Please enter between 1 & 10 same or different from the actual face =

If for example the following numbers (120, 10, 120 and 7) were entered for the above request respectively, then MATLAB uses the algorithm above to find:
(i) the features identified in the original image;
(ii) the SIFT feature extraction and matching; and
(iii) the RBFNN adaptive SIFT-SURF feature extraction and matching as displayed in the following Figures.

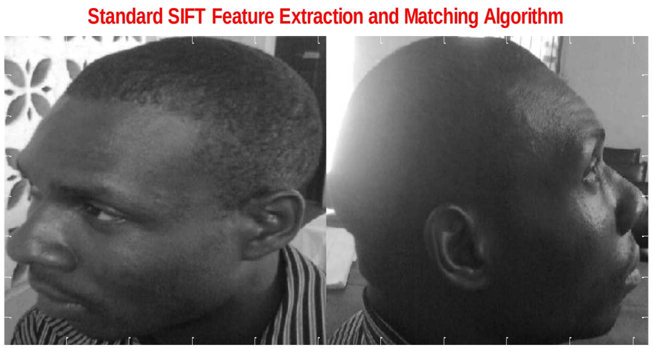
4. RESULTS AND DISCUSSION

From the results obtained due to the simulation of the above algorithm and provision of the necessary folder and actual face numbers, four different cases are as discussed follows.

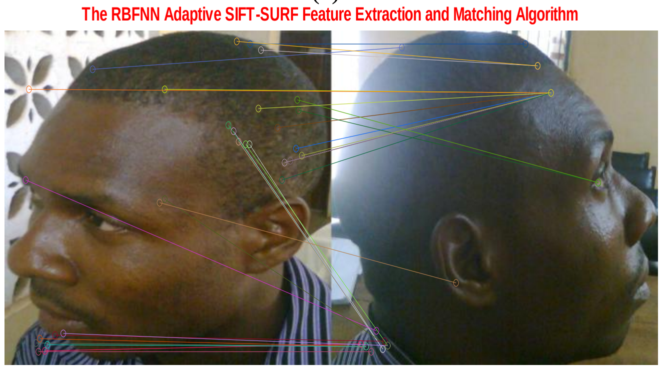
CASE I:



(a)



(b)


The RBFNN Adaptive SIFT-SURF Feature Extraction and Matching Algorithm

(c)

Fig 2: CASE 1: (a) Feature extraction of image from Folder 120 and corresponding image 10; (b) Standard SIFT feature extraction and matching of image from Folder 120 and corresponding images 10 and 7; and (c) The RBFNN adaptive SIFT-SURF feature extraction of image from Folder 120 and corresponding images 10 and 7.

Features of image from *Folder 120* and the corresponding image *10* have been displayed as the true facial image. The facial feature extraction capabilities of the standard SIFT algorithm and that of the proposed RBFNN adaptive SIFT-SURF algorithm are shown in Fig. 2(b) and (c) respectively for facial image *7* as a test image from the same *Folder 120*. The performances of the standard SIFT algorithm and the proposed RBFNN adaptive SIFT-SURF algorithm are summarized in Table 2. The excellent performance of the proposed RBFNN adaptive SIFT-SURF is obvious in Fig. 2(c) with 66 keypoints when compared to the total keypoints of 80 identified from the facial image *10* in *Folder 120* as shown in Table 2 when compared to 0 obtained by the standard SIFT algorithm (i.e. no feature could be captured by the standard SIFT algorithm. Note that the threshold keypoints for this facial image is 53 (which is 2/3 of total keypoints identified and is arbitrarily chosen for this study). The number of descriptors from the standard SIFT and the proposed RBFNN adaptive SIFT-SURF algorithms are shown together in Fig. 6(a) for quick comparison. Furthermore, the proposed RBFNN adaptive SIFT-SURF algorithm has the least computation time of 1.1513 with MATLAB's *parfor* command for parallel implementation of the proposed algorithm when compared to the standard SIFT algorithm. Please note that the first and second columns in CASE 1 column corresponds to the performance evaluation for the parameters considered for the standard SIFT and the proposed RBFNN adaptive SIFT-SURF algorithm respectively.

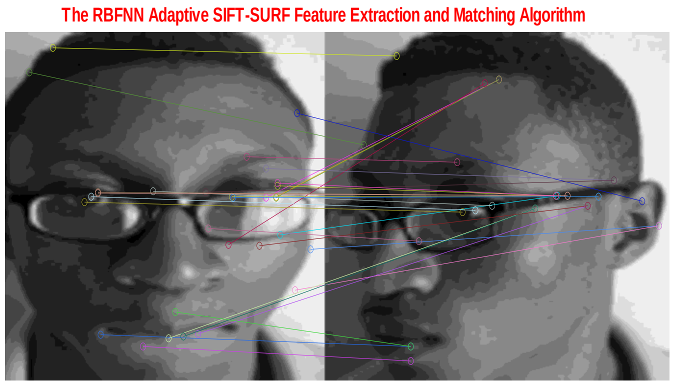
<u>CASE 2:</u>

For the image displayed in Fig. 3(a) it can be seen that the features of image from *Folder 100* and the corresponding image *3* has been displayed as the true facial image. The facial feature extraction capabilities of the standard SIFT algorithm and that of the proposed RBFNN adaptive SIFT-SURF algorithm are shown in Fig. 3(b) and (c) respectively for facial image *9* as a test image from the same *Folder 100*. The performances of the standard SIFT algorithm and the proposed RBFNN adaptive SIFT-SURF algorithm are summarized in Table 2. The excellent performance of the proposed RBFNN adaptive SIFT-SURF is obvious in Fig. 3(c) with 16 keypoints when compared to the total keypoints of 19 identified from the facial image *3* in *Folder 100* as shown in Table 2 when compared to 1 keypoint obtained by the standard SIFT algorithm (i.e. no feature could be captured by the standard SIFT algorithm. Note that the threshold keypoints for this facial image is 13 (which is 2/3 of total keypoints identified and is arbitrarily chosen for this study). The number of descriptors from the standard SIFT and the proposed RBFNN adaptive SIFT-SURF algorithms are shown together in Fig. 6(b) for quick comparison. Furthermore, the proposed RBFNN adaptive SIFT-SURF algorithm has the least computation time of 1.0046 with MATLAB's *parfor* command for parallel implementation of the proposed algorithm when compared to the standard SIFT algorithm. Please note that the first and second columns in CASE 2 column corresponds to the performance evaluation for the parameters considered for the standard SIFT and the proposed RBFNN adaptive SIFT-SURF algorithm respectively.

<u>CASE 3:</u>

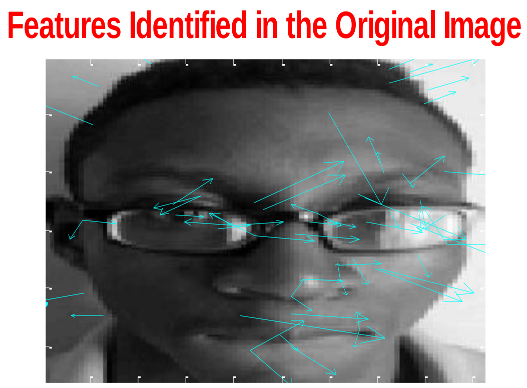
For the image displayed in Fig. 4(a) (see Appendix), it can be seen that the features of image from *Folder 10* and the corresponding image *10* has been displayed as the true facial image. The facial feature extraction capabilities of the standard SIFT algorithm and that of the proposed RBFNN adaptive SIFT-SURF algorithm are shown in Fig. 4(b) and (c) respectively (see Appendix) for facial image *2* as a test image from the same *Folder 10*. The performances of the standard SIFT algorithm and the proposed RBFNN adaptive SIFT-SURF algorithm are summarized in Table 2. The excellent performance of the proposed RBFNN adaptive SIFT-SURF is obvious in Fig. 4(c) with 46 keypoints when compared to the total keypoints of 51 identified from the facial image *2* in *Folder 10* as shown in Table 2 when compared to 16 obtained by the standard SIFT algorithm (i.e. no feature could be captured by the standard SIFT algorithm. Note that the threshold keypoints for this facial image is 34 (which is 2/3 of total keypoints identified and is arbitrarily chosen for this study). The number of descriptors from the standard SIFT and the proposed RBFNN adaptive SIFT-SURF algorithms are shown together in Fig. 6(c) for quick comparison. Furthermore, the proposed RBFNN adaptive SIFT-SURF algorithm has the least computation time of 0.8830 with MATLAB's *parfor* command for parallel implementation of the proposed algorithm when compared to the standard SIFT algorithm. Please note that the first and second columns in CASE 3 column corresponds to the performance evaluation for the parameters considered for the standard SIFT and the proposed RBFNN adaptive SIFT-SURF algorithm respectively.

It can be observed that even though the same facial image *2*taken from the same *Folder 10* is considered, the proposed RBFNN adaptive SIFT-SURF outperforms the standard SIFT algorithm in all ramifications especially with an excellent high keypoints identifications of 46 against 16 over a total of 51 keypoints.

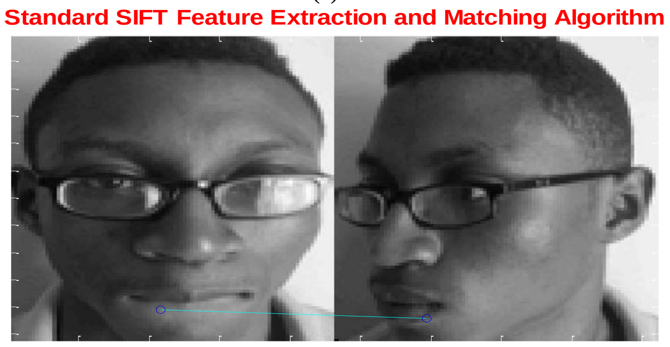

The RBFNN Adaptive SIFT-SURF Feature Extraction and Matching Algorithm

(c)

Fig. 3: CASE 2: (a) Feature extraction of image from folder 100 and corresponding image 3; (b): Standard SIFT feature extraction and matching of image from folder 100 and corresponding images 3 and 9; and (c) The RBFNN adaptive



(a)



(b)

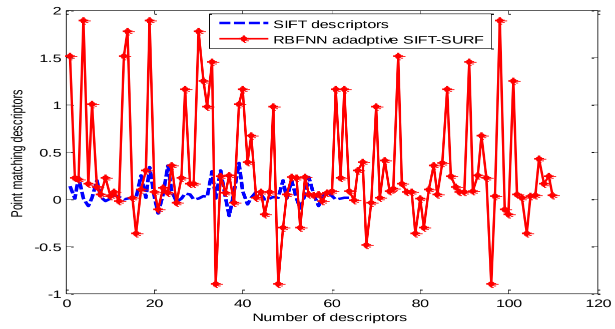
CASE 4:

For the image displayed in Figure 5(a) (see Appendix) it can be seen that the features of an image from *Folder 150* and the corresponding image *10* have been displayed as the true facial image. The facial feature extraction capabilities of the standard SIFT algorithm and that of the proposed RBFNN adaptive SIFT-SURF algorithm are shown in Figures 5(b) and (c) respectively (see Appendix) for facial image *10* as a test image from the same *Folder 150*. The performances of the standard SIFT algorithm and the proposed RBFNN adaptive SIFT-SURF algorithm are summarized in Table 2. The excellent performance of the proposed RBFNN adaptive SIFT-SURF is obvious in Figure
5(c) with 38 keypoints when compared to the total keypoints of 50 identified from the facial image *10* in *Folder 150* as shown in Table 2 when compared to 1 obtained by the standard SIFT algorithm (i.e. no feature could be captured by the standard SIFT algorithm. Note that the threshold keypoints for this facial image is 33 (which is 2/3 of total keypoints identified and is arbitrarily chosen for this study). The number of descriptors from the standard SIFT and the proposed RBFNN adaptive SIFT-SURF algorithms are shown together in Figure 6(d) for quick comparison. Furthermore, the proposed RBFNN adaptive SIFT-SURF algorithm has the least computation time of 1.2355 with MATLAB's *parfor* command for parallel implementation of the proposed algorithm when compared to the standard SIFT algorithm. Please note that the first and second columns in CASE 4 column corresponds to the performance evaluation for the parameters considered for the standard SIFT and the proposed RBFNN adaptive SIFT-SURF algorithm respectively. It can be observed that even though the same facial image *10*taken from the same *Folder*

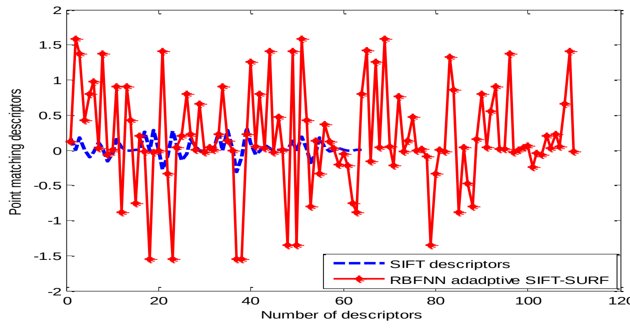
150 is considered, the proposed RBFNN adaptive SIFT-SURF outperforms the standard SIFT algorithm in all ramifications especially with the excellent high keypoints identifications of 38 against 1 compared to the total of 50.

Table 2: Performance comparison of the standard SIFT algorithm with the proposed RBFNN adaptive SIFT-SURF algorithm for facial image feature extraction

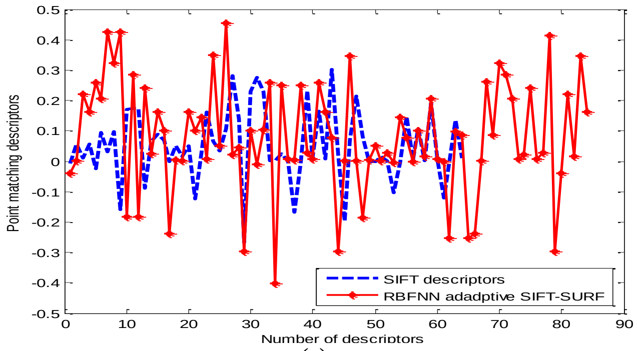
S/N	Parameters	CASE 1		CASE 2		CASE 3		CASE 4	
1.	Folder number	120		100		10		150	
2.	Incoming image number	10		3		2		10	
3.	Test image number	7		9		6		10	
4.	Number of keypoints identified	80		19		51		50	
5.	Threshod keypoints for image to be identified	53		13		34		33	
6.	Keypoints identified by SIFT	0		1		16		1	
7.	Keypoints identified by RBFNN adaptive SIFT-SURF	66		16		46		38	
8.	SIFT computation time	2.4336		2.0436		1.9500		2.3868	
9.	RBFNN adaptive SIFT-SURF computation time	5.78		5.0388		4.4304		6.1932	
10.	RBFNN adaptive SIFT-SURF computation time with *parfor*	1.1513		1.0046		0.8830		1.2355	
11.	num_x	348.4155	146.2459	275.3718	102.3567	142.0739	208.6873	361.9604	167.1032
12.	num_y	12.5757	94.5917	50.9807	135.0591	134.3192	125.7694	16.9392	108.2561
13.	num_scale	1.8924	0.8016	1.9905	0.8732	0.8657	0.7618	1.8054	0.9154
14.	num_laplacian	1	0.0218	0	0.04354	0.4027	0.07209	0	0.05715
15.	num_orient	4.9229	1.5941	1.5657	2.4572	4.6731	1.7340	0.2101	2.4133



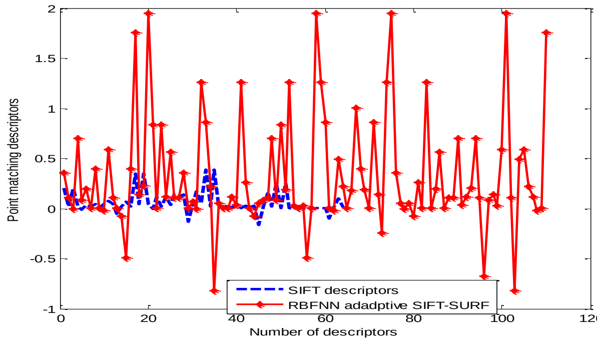
(a)



(b)



(c)



(d)

Fig. 6: Performance comparison of the descriptors form the standard SIFT and the proposed RBFNN adaptive SIFT-SURF algorithms for facial image feature extraction: (a) for CASE 1, (b) for CASE 2, (c) for CASE 3 and (d) for CASE 4.

5. CONCLUSION

A radial basis function neural network-based (RBFNN) adaptive SIFT-SURF algorithm has been proposed and developed in this paper which is suitable for feature extraction from still images. The performance of the proposed algorithm has been demonstrated on 1500 facial images and its efficiency has been compared with that of the standard SIFT algorithm. The simulation results have shown that the proposed RBFNN adaptive algorithm outperforms the standard SIFT algorithm with enhanced optimal facial image feature extraction for easy recognition and identification. The performances of the proposed RBFNN adaptive SIFT-SURF algorithm for feature extraction on the 1500 face image samples taken under different conditions such as angular positions, different postures and light intensities demonstrate the efficiency and adaptability of the proposed algorithm for critical and unmatched situations.

Furthermore, it can be seen from the results obtained that the proposed facial image feature extraction using radial basis function neural networks and adaptive SIFT-SURF algorithms performs much better compared with the results obtained using the standard SIFT feature extraction algorithm as evident in Fig. 2 to Fig. 5. This means that the proposed RBFNN adaptive SIFT-SURF facial feature extraction algorithm is more consistent in the extraction of image features and matching in different illumination conditions and posture for easy tracking, identification and recognition.

The implementation of the proposed adaptive RBFNN adaptive SIFT-SURF algorithm on a real-time embedded reconfigurable computing machines such as digital signal processors (DSPs), field programmable gate arrays (FPGAs) [41], complex programmable logic devices (CPLDs) as well as on other dedicated high performance computing platforms interfaced with real-time camera could facilitate the development and deployment of real-time on-line facial image feature extraction, detection, identification, recognition, tracking and analysis systems from streaming videos and images.

REFERENCES

[1] W. K. Mutlag, S. K. Ali, Z. M. Aydam and B. H. Taher, "Feature Extraction Methods: A Review", *Journal of Physics: Conference Series, FISCAS 2020, IOP Publishing*, vol. 1591, pp. 1 – 11, 2020.

[2] S. K. P. Pooja and K. Veer, "Recent Approaches on Classification and Feature Extraction of EEG Signal: A Review", *Robotica*, vol. 1, pp. 1 – 25, 2021.

[3] J. Lian, M. Zhang , N. Jiang , W. Bi and X. Dong, "Feature Extraction of Kidney Tissue Image Based on Ultrasound Image Segmentation", *Journal of Healthcare Engineering*, vol. 2021, pp. 1 – 16, 2021.

[4] G. K. Sidiropoulos , P. Kiratsa, P. Chatzipetrou and G. A. Papakostas, "Feature Extraction for Finger-Vein-Based Identity Recognition", *Journal of Imaging*, vol. 7, no. 89, pp. 1 – 28, 2021.

[5] N. Mishra and A. Bhatt, "Feature Extraction Techniques in Facial Expression Recognition", *2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS)*, pp. 1247 – 1251, 2021, DOI: 10.1109/ICICCS51141.2021.9432192.

[6] Z. Cheng, B. Cui, T. Qi, W. Yang, and J. Fu, "An Improved Feature Extraction Approach for Web Anomaly Detection Based on Semantic Structure", *Security and Communication Networks*, vol. 2021, pp. 1 – 11, 2021.

[7] B. Hu, "Deep Learning Image Feature Recognition Algorithm for Judgment on the Rationality of Landscape Planning and Design", *Complexity*, vol. 2021, pp. 1 – 15, 2021.

[8] M. Bansal, M. Kumar and M. Kumar, "2D Object Recognition: a Comparative Analysis of SIFT, SURF and ORB Feature

Descriptors", *Multimedia Tools and Applications*, vol. 80, pp. 18839 – 18857, 2021.

[9] R. Brunelli and T. Poggio, "Face recognition: features versus templates", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 10, pp. 1042 -1052, 1993.

[10] D. Shah and S. Marshall, "Statistical coding method for facial feature", *IEEE Proceedings of the Visual Image Signal Processing*, vol. 145, pp. 187-192, 1998.

[11] L. King and L. Xu, "Localized principal component analysis learning for face extraction and recognition", *Proceedings of Workshop on 3-D Computer Vision*, pp. 124 -128, 1997.

[12] R. J. Baron, "Mechanisms of human facial recognition", *International Journal on Man Machine Studies*, vol. 15, pp 137-178, 1981.

[13] G. G. Yen and N. Nithianandan, "Facial feature extraction using genetic algorithm", *IEEE Proceedings of the 2002 Congress on Evolutionary Computation*, May 2002, Honolulu, U.S.A., pp. 1 – 6, 2012.

[14] A. Anjali, A. Kumar, R. Ranjan, "An algorithm for face detection and feature extraction", *International Journal of Science, Engineering and Technology Research*, vol. 3, no. 4, pp. 1159 – 1165, 2014.

[15] J. Lu, K. N. Plataniotis, L. D Harmon, M. K. Khan, M. R. Lasch and P. F. Ramig, "Machine identification of human faces", *Pattern Recognition*, vol.13, no. 2, pp. 97-110 , 1981.

[16] H. B. Kekre, S. D. Thepade and A. Maloo, "Face recognition using texture features extracted from walshlet pyramid", *Int. Journal on Recent Trends in Engineering. & Technology,* vol. 5, no. 1, pp. 186, 2011.

[17] H. B. Kekre, S. D. Thepade and S. Khandelwal, "Face recognition using multilevel block truncation coding," *International Journal of Computer Applications*, vol. 36, no.11, pp. 0975 – 8887, 2011.

[18] A. N. Venetsanopoulos, " Face recognition using kernel direct discriminant analysis algorithms", *IEEE Transactions on Neural Networks,* vol. 14, no. 1, pp.117 – 126, 2003.

[19] R. Brunelli and T. Poggio, "Face recognition: Features versus templates," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 15, pp. 1042–1052, 1993.

[20] A. V. Nefian and M. H. Hayes, "Hidden markov models for face detection and recognition", *IEEE Transactions on Pattern Analysis And Machine Intelligence,* Vol.1, pp. 141 – 145, 1999.

[21] S. Liao, W. Fan, A. C. S. Chung, D. Y. Yeung, "Facial expression recognition using advanced local binary patterns: Tsallis entropies and global appearance features", *In Proceedings of IEEE International Conference on Image Processing*, pp. 665 – 668, 2006.

[22] S. K. Datta, P. Morrow and B. Scotney, "Facial feature extraction using a 4d stereo camera system", *S. Patnaik and Y. M. Yang (Eds): Soft Computing Techniques in Vision Science*, SCI 395, pp. 209 – 2018, Springer-Verlag Berlin Heidelberg, 2012.

[23] Z. Wang, F. K. Huangfu, and J. W. Wan, "Human face feature extraction using deformable templates", *Journal of Computer Aided Design and Computer Graphics of China*, vol. 12, no. 5, pp. 333 – 336, 2000.

[24] S. L. Phung, A. Bouzerdoum and D. Chai, "Skin segmentation using color and edge information", *In Proceedings of International Conference on Signal Processing and its Applications*, vol. 1, pp. 525–528, 2003.

[25] T. Sawangsri, V. Patanavijit and S. Jitapunkul, "Face segmentation using novel skin-color map and morphological technique", *In Proceedings of World Academy of Science, Engineering and Technology*, vol. 2, 2005.

[26] H. I. Thai, M. N. Tri and T. N. Hang, "Proposal of a new method of feature extraction for face recognition", *In Proceeding of National Conference about Information Technology*, DaLat City, 2006.

[27] D. G. Lowe, "Distinctive image features from scale-invariant keypoints", *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91 – 110, 2004.

[28] H. Bay, A. Ess, T. Tuytelaars and L. van-Gool, "Speeded-up robust features (surf)", *Journal of Computer Vision*, vol. 110, no. 3, pp. 346 – 359, 2008.

[29] J. Park and L. Sandberg, "Universal approximation using radial-basis-function networks" *Neural Computation*, vol. 3, pp. 246-257, 1991.

[30] D. Simon, "Training radial basis function neural networks with the extended Kalman filter", *Neurocomputing*, vol. 48, pp. 455 – 475, 2002.

[31] D. Broomhead and D. Lowe, "Multivariable functional interpolation and adaptive networks", *Complex Systems*, vol. 2, pp. 321–355, 1988.

[32] J. Moody and C. Darken, "Fast learning in networks of locally-tuned processing units", *Neural Computations*, vol. 1, pp. 289–303, 1989.

[33] N. Karayiannis, "Reformulated radial basis neural networks trained by gradient descent", *IEEE Transactions on Neural Networks*, vol. 3, pp. 657–671, 1999.

[34] D. Simon, "Distributed fault tolerance in optimal interpolative nets", *IEEE Transactions on Neural Networks*, vol. 12, no. 6, pp. 1348 – 1357, 2001.

[35] S. Sin and R. DeFigueiredo, "Efficient learning procedures for optimal interpolative nets", *Neural Networks*, vol. 6, pp. 99–113, 1993.

[36] V. A. Akpan, "*Development of new model adaptive predictive control algorithms and their implementation on real-time embedded systems"*, Ph.D. Dissertation, 517 pages, 2011. Available [Online]: http://invenio.lib.auth.gr/record/127274/files/GRI-2011-7292.pdf.

[37] V. A. Akpan and G. D. Hassapis, "Training dynamic feedforward neural networks for online nonlinear model identification and control applications", *International Reviews*

of Automatic Control: Theory & Applications, vol. 4, no. 3, pp. 335 – 350, 2011.

[38] S. Chen, C. Cowan and P. Grant, "Orthogonal least squares learning algorithm for radial basis function networks", *IEEE Transactions on Neural Networks*, vol. 2, pp. 302–309, 1991.

[39] V. A. Akpan, J. B. Agbogun, M. T. Babalola and B. A. Oluwade, "Radial basis function neuroscaling algorithm for efficient facial image recognition", *Machine Learning Research*, Vol. 2, no. 4, pp. 152 – 168, 2018.

[40] The MathWorks Inc., *MATLAB & Simulink® R2021a*, 3 Maple Drive, California, U. S. A. http://www.mathworks.com.

[41] V. A. Akpan, "Hard and soft embedded FPGA processor systems design: Design considerations and performance comparisons", *International Journal of Engineering and Technology*, vol. 3, no. 11, pp. 1000 – 1020, 2013. Available [Online]: http://www.iet-journals.org/archive/2013/november_vol_3_no_11/7153671377348526%e2%80%8f.pdf.

ABOUT THE AUTHORS

Vincent A. Akpan holds a Ph. D. degree in Electrical & Computer Engineering from the Aristotle University of Thessaloniki (AUTH), Thessaloniki, Greece in 2011.

He is currently an Associate Professor and the Head of Biomedical Technology Department, FUTA, Akure, Nigeria. His research interest is in advanced instrumentation, intelligent control and embedded systems engineering. He is the co-author of a book and has authored and/or co-authored more than 60 articles in refereed journals and conference proceedings.

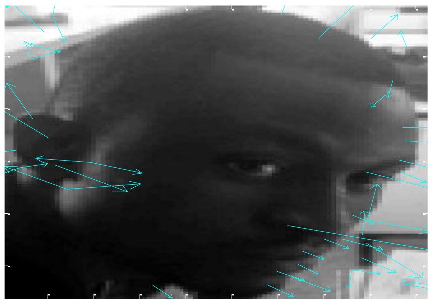
Dr. Akpan is a member of several local and international professional societies. Dr. Akpan is Fellow of the College of Biomedical Engineering & Technology (CBET), Nigeria; and a Fellow of the IPMD, Nigeria.

Joshua B. Agbogun holds a Ph.D. degree in Computer Science from Kogi State University, Anyigba, Nigeria in 2019.

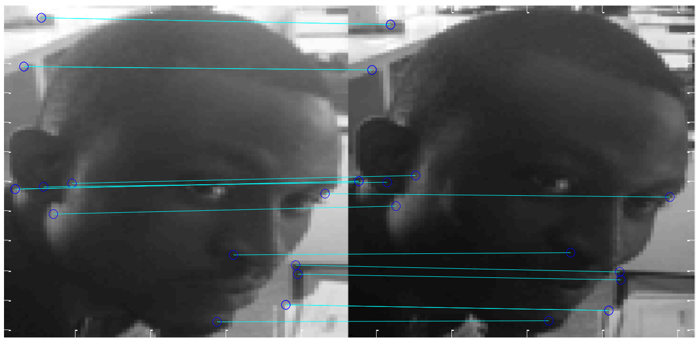
He is currently a Lecturer I and the Acting Head of Department of Computer Science and Mathematics, Godfrey Okoye University Enugu, Enugu State, Nigeria. His research interest is in Machine Learning. He is the author of a book and has authored and/or co-authored 21 articles in refereed Journals and conference proceedings.

Dr. Agbogun is a member of Nigeria Computer Society (NCS) and has served as the NCS Chairman, Kogi State Chapter from 2013 to 2019.

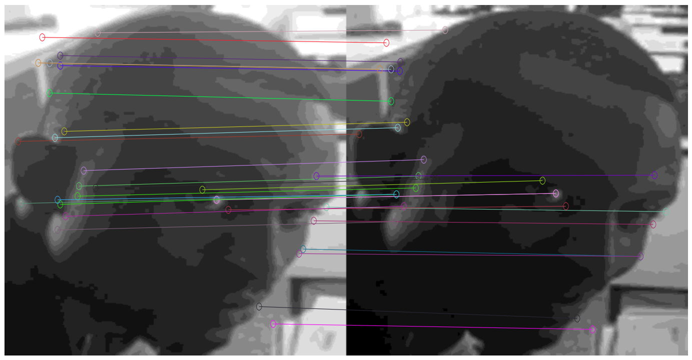
APPENDIX



(a)

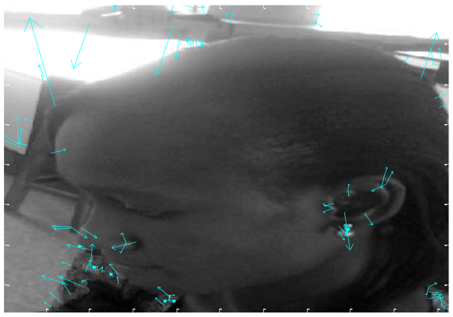


(b)

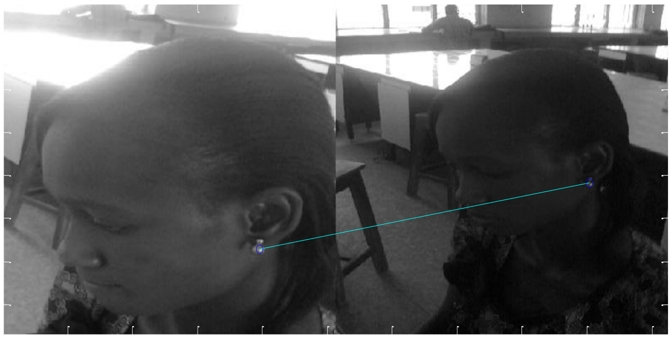

The RBFNN Adaptive SIFT-SURF Feature Extraction and Matching Algorithm

(c)

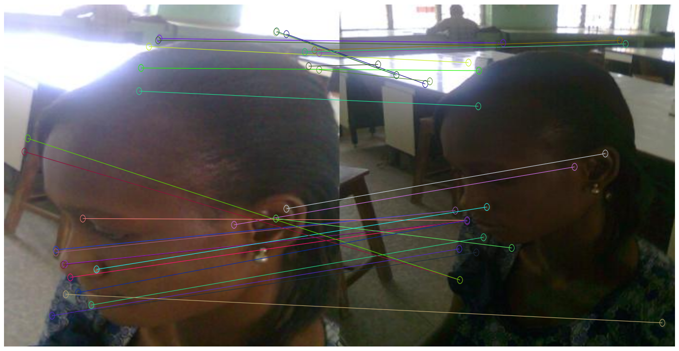
Fig. 4: CASE 3: (a) Feature extraction of image from *Folder 10* and corresponding image *2*; (b): Standard SIFT feature extraction and matching of image from *Folder 10* and corresponding images *2* and *2*; and (c) The RBFNN adaptive SIFT-SURF feature extraction of image from *Folder 10* and corresponding images *2* and 2.



(a)



(b)


The RBFNN Adaptive SIFT-SURF Feature Extraction and Matching Algorithm

(c)

Fig. 5: CASE 4: (a) Feature extraction of image from folder 150 and corresponding image 10; (b) Standard SIFT feature extraction and matching of image from folder 150 and corresponding images 10 and 10.and (c) The RBFNN adaptive SIFT-SURF feature extraction of image from folder 150 and corresponding images 10 and 10.