Age Invariant Face Recognition Using Frangi2D Binary Pattern

Rajyaguru Keval(201911020@daiict.ac.in)

Paper Information:

- **Title:** Age invariant face recognition using Frangi2D binary pattern
- **Authors:** Sabah Afroze, M. Parisa Beham, R. Tamilselvi, S M Seeni Mohamed Aliar Maraikkayar, K Rajakumar
- **Research Institute**: Sethu Institute of Technology, Tamilnadu, India.
- **Publication:** ICIGP '19: Proceedings of the 2nd International Conference on Image and Graphics Processing
- Published in **February,2019**

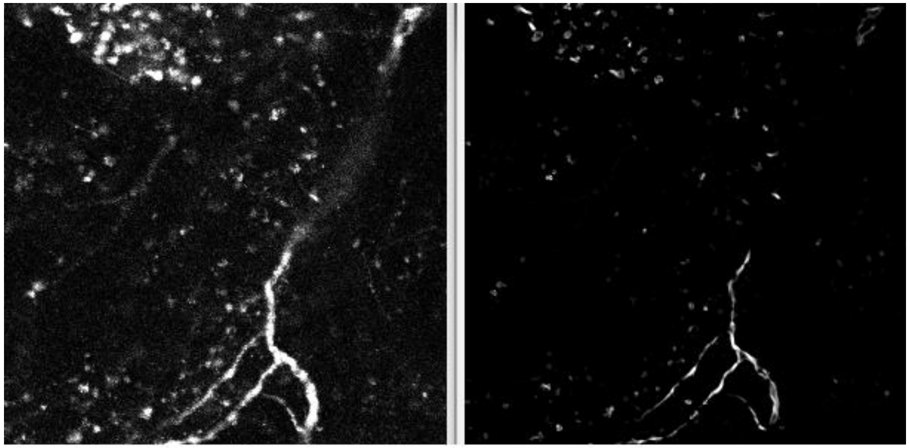
Dataset Used:

- **Dataset Used in Paper**: **MORPH** dataset contains images of people over different age-Not publicly available.

- **Dataset Used For Implementation:**
- **CADC2000**-Contains images of different 2000 celebrities captured over the span of 10-50 years.
- **FGNET:** Contains images of 81 people captured with age variance.

Frangi 2D:-

- Frangi 2D method is used to highlight tubular structure or vessel like structure in images.
- It was mainly developed to highlight different veins of our body from images.



- It uses double derivatives based hessian matrix.

Frangi 2D-Algorithm

- Create hessian matrix of image with Gaussian kernel with σ variance. Mean for Gaussian kernel is taken to be 0.

$$H = \begin{vmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{vmatrix} = \begin{vmatrix} \frac{\partial^2 I_t}{\partial x'^2} & \frac{\partial^2 I_t}{\partial x' \partial y'} \\ \frac{\partial^2 I_t}{\partial y \partial x'} & \frac{\partial^2 I_t}{\partial y'^2} \end{vmatrix}$$

- Hessian is symmetric matrix. In above case x and y are considered to be independent . I is pixel intensity/value at x , y co-ordinates.

Frandi2D-Gaussian Kernel:

- Gaussian kernel for an image is given below with mean 0.

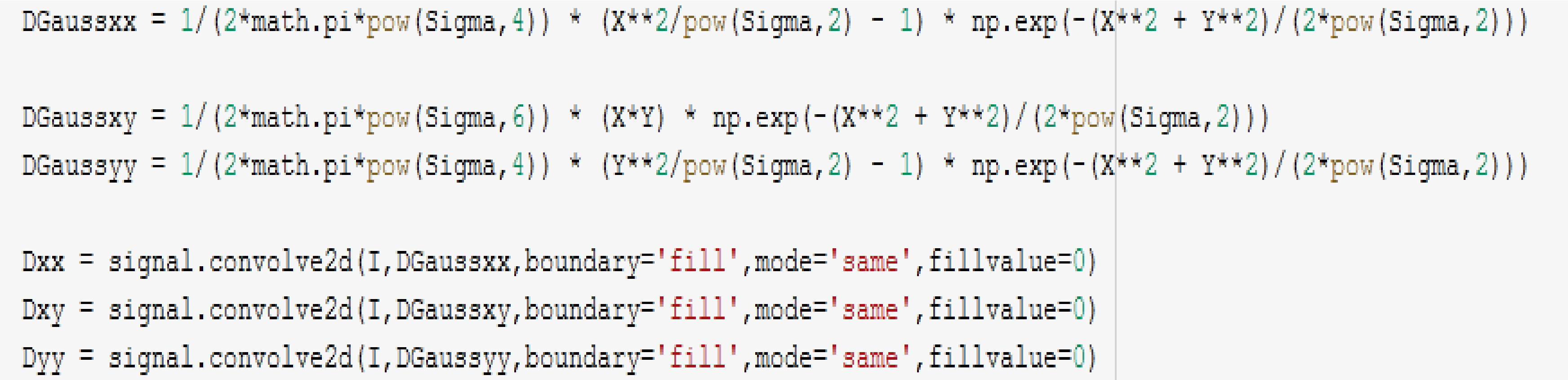
$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- Second derivative of this kernel is calculated and convoluted with image to get $h_{xx,}$ h_{yy} and $h_{xy.}$

- Second derivative of image with respect to Gaussian kernel is defined as convolution of derivative of Gaussian with Image.

Frangi2D-Gaussian Derivative

- **Gaussian Derivatives**:



```
DGaussxx = 1/(2*math.pi*pow(Sigma,4)) * (X**2/pow(Sigma,2) - 1) * np.exp(-(X**2 + Y**2)/(2*pow(Sigma,2)))

DGaussxy = 1/(2*math.pi*pow(Sigma,6)) * (X*Y) * np.exp(-(X**2 + Y**2)/(2*pow(Sigma,2)))
DGaussyy = 1/(2*math.pi*pow(Sigma,4)) * (Y**2/pow(Sigma,2) - 1) * np.exp(-(X**2 + Y**2)/(2*pow(Sigma,2)))

Dxx = signal.convolve2d(I,DGaussxx,boundary='fill',mode='same',fillvalue=0)
Dxy = signal.convolve2d(I,DGaussxy,boundary='fill',mode='same',fillvalue=0)
Dyy = signal.convolve2d(I,DGaussyy,boundary='fill',mode='same',fillvalue=0)
```

- Now Eigen values are calculated of above 3 Image derivatives D_{xx}, D_{yy}, D_{xy}.

Gaussian Kernel And Derivatives

→ $$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

$$G_x(x,y) = \boxed{\frac{-x}{\sigma^2} \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}}$$

$$G_y(x,y) = \boxed{\frac{-y}{\sigma^2} \cdot \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}}$$

$$G_{xx}(x,y) = \frac{-1}{\sigma^2} e^{-(x^2+y^2)/2\sigma^2} + \left(\frac{-x}{\sigma^2}\right)^2 \cdot \frac{1}{2\pi\sigma^2} \cdot e^{-(x^2+y^2)/2\sigma^2}$$

$$= \boxed{\frac{1}{2\pi\sigma^4}\left[\frac{x^2}{\sigma^2} - 1\right] e^{-(x^2+y^2)/2\sigma^2}}$$

$$G_{yy}(x,y) = \boxed{\frac{1}{2\pi\sigma^4}\left[\frac{y^2}{\sigma^2} - 1\right] e^{-(x^2+y^2)/2\sigma^2}}$$

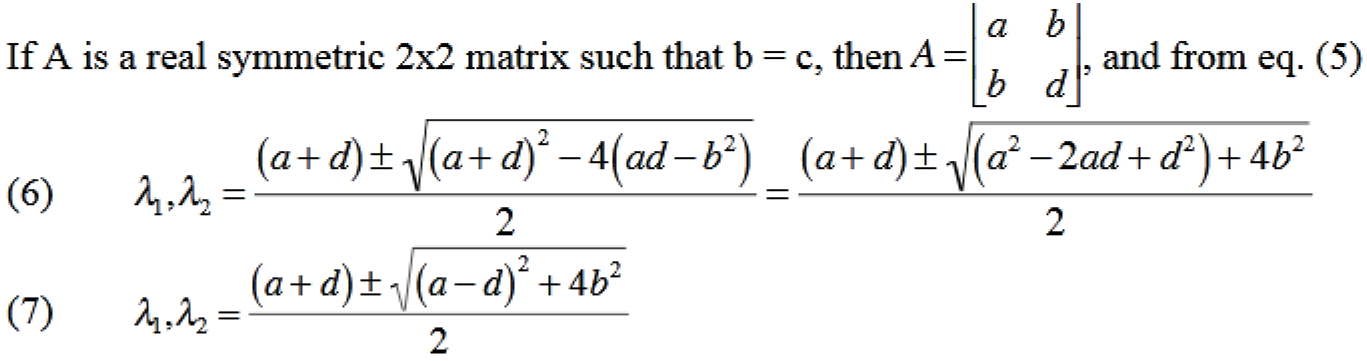
$$G_{xy}(x,y) = -\frac{x}{\sigma^2} \cdot \frac{-y}{\sigma^2} \cdot e^{-(x^2+y^2)/2\sigma^2} \cdot \frac{1}{2\pi\sigma^2}$$

$$= \frac{xy}{\sigma^4} \cdot \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

$$= \boxed{\frac{1}{2\pi\sigma^6} \cdot xy \cdot e^{-(x^2+y^2)/2\sigma^2}}$$

Frangi2D-Eigen Values

- Eigen Values are computed with given equations for symmetric matrix.

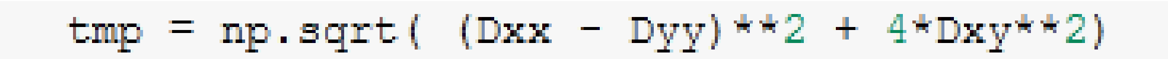


If A is a real symmetric 2x2 matrix such that b = c, then $A = \begin{bmatrix} a & b \\ b & d \end{bmatrix}$, and from eq. (5)

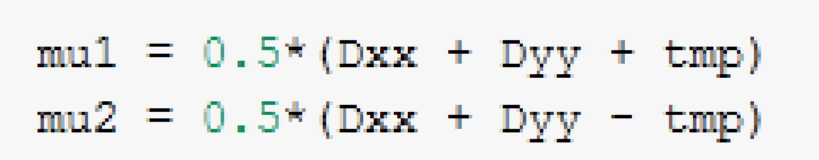
(6) $$\lambda_1, \lambda_2 = \frac{(a+d) \pm \sqrt{(a+d)^2 - 4(ad - b^2)}}{2} = \frac{(a+d) \pm \sqrt{(a^2 - 2ad + d^2) + 4b^2}}{2}$$

(7) $$\lambda_1, \lambda_2 = \frac{(a+d) \pm \sqrt{(a-d)^2 + 4b^2}}{2}$$

- Code:



```
 tmp = np.sqrt( (Dxx - Dyy)**2 + 4*Dxy**2)
```



```
mu1 = 0.5*(Dxx + Dyy + tmp)
mu2 = 0.5*(Dxx + Dyy - tmp)
```

Frangi2D

- Finally Frangi co-efficient for given image is calculated using given 3 formulas.

$$R_B = \frac{\lambda_1}{\lambda_2}$$

$$S = \sqrt{\left(\lambda_1^2 + \lambda_2^2\right)}$$

$$I_B = \begin{cases} 0, & if\ \lambda_2 > 0, \\ \exp\left(-\frac{R_B^2}{2\beta^2}\right) & \left(1 - \exp\left(\frac{-S^2}{2c^2}\right)\right) \end{cases}$$

- R_s here suggests the line like structure(2D).
- In the equation on the right, **β** and **c** are constants. Accuracy of detecting vessel and removal of noise depends on **β** and **c**.
- S combined with c does the noise removal.

Why Frangi Here?

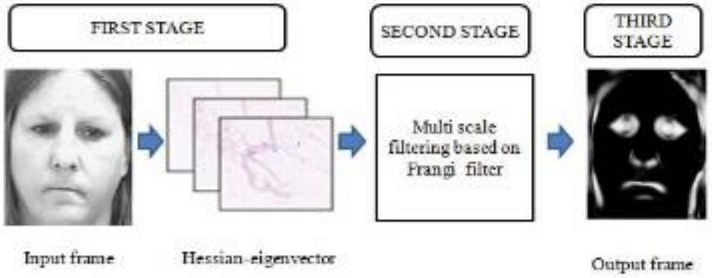
- As stated, Frangi detects vessel structures.
- Age variance of a face of a person can be seen as the wrinkles on the face of a person.

.

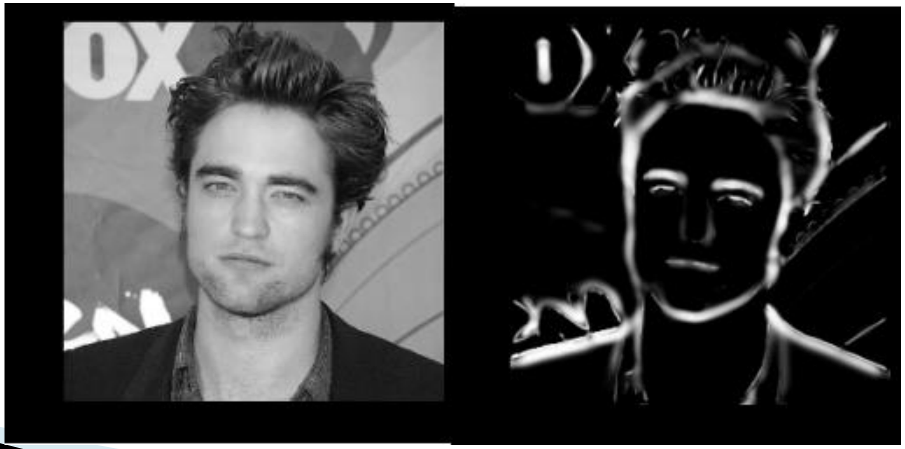
- Different types of sigma will work with different vessels. Now wrinkles on our face may consider as small vessel where as edges that contains our face and eyes and other relevant part is considered as large vessels.
- We have applied σ =1.5 and 4.5 as stated in paper.

Frangi Outputs

- Output of paper:

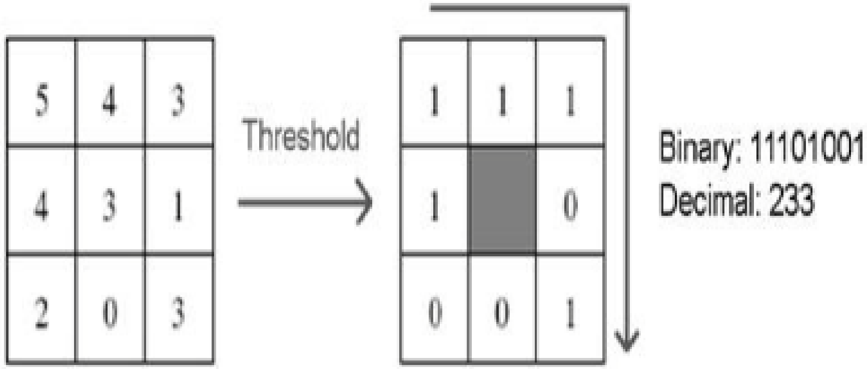


- Output On Celebrity Dataset:



Local Binary Pattern(LBP)

- LBP is method to for extracting textures from image.
- LBP calculates values for each pixel according to its neighbors. So for radius one, we can have 8 neighbors. LBP is calculated as shown below.



5	4	3
4	3	1
2	0	3

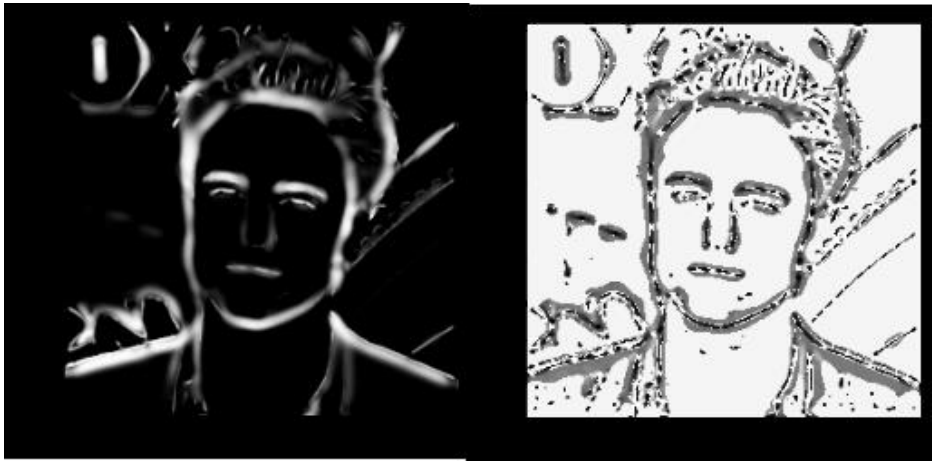
1	1	1
1		0
0	0	1

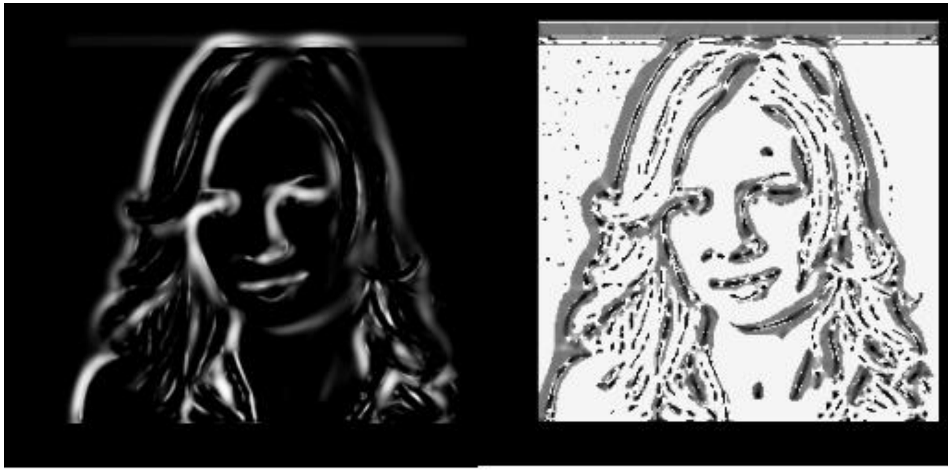
Binary: 11101001
Decimal: 233

Local Binary Pattern

- For each and every pixel above value are calculated.
- One way, here each direction is given some weight. So starting from 2^0 to 2^7 is weight age given to each direction starting from left top corner.
- It highlights the change in surrounding. If one portion taken is total black/white containing all pixel 0/255 then it will assign finally 255 to that value. Other Wise based on the surrounding intensities, LBP assigns a unique value.

Local Binary Pattern-Output





Sparse Representation Classifier

- **Basic Thoughts**:

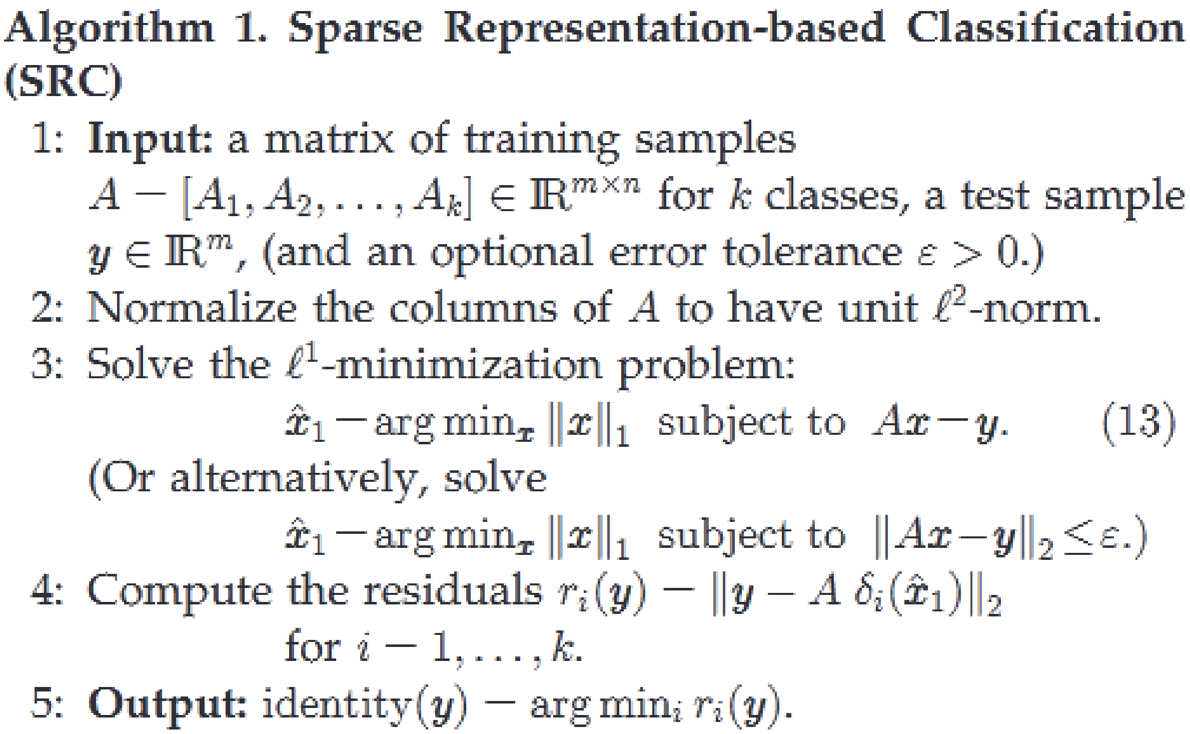
- Any image **I1** which belongs to class **C** can be represented as linear combination of the images that belong to class **C.**

- Mainly used for classifying faces

- Creates sparse representation for different classes to classify images.

- We also applied **PCA** reducing dimensions to 200 features to proceed with SRC faster.

SRC Algorithm:



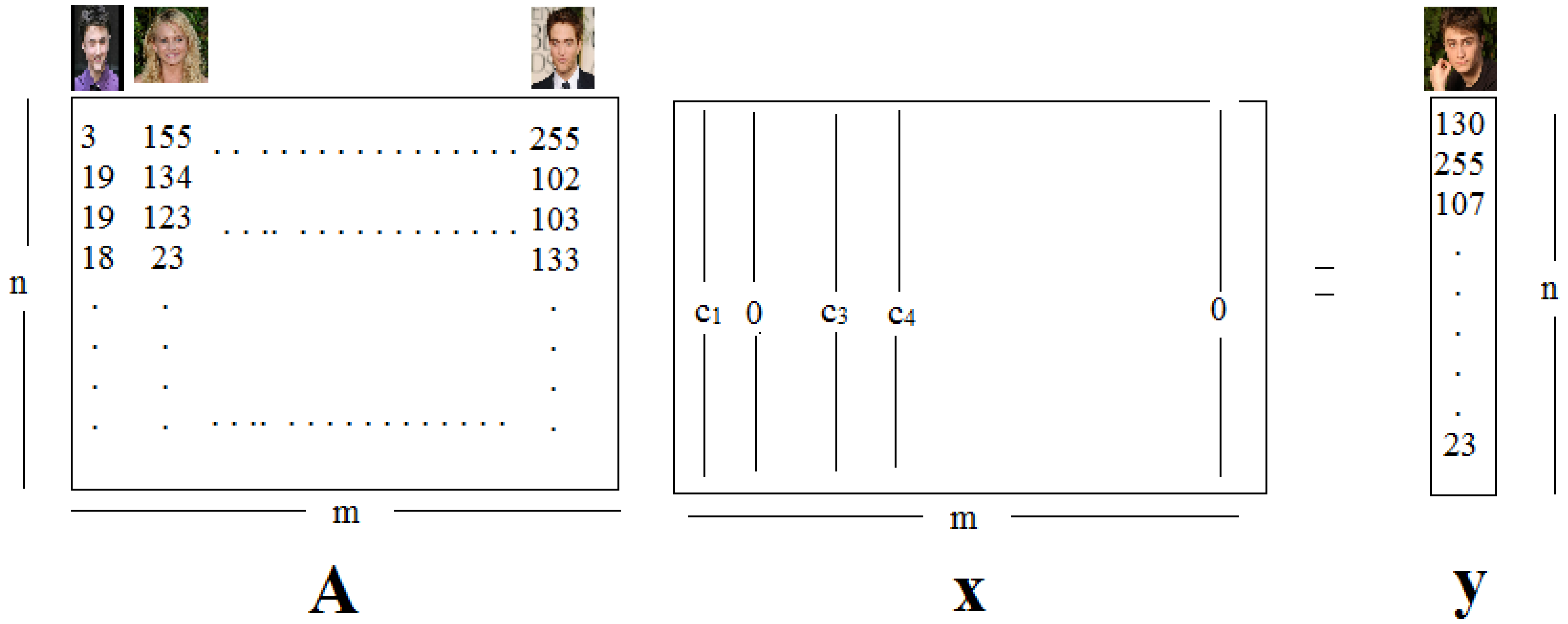
Algorithm 1. Sparse Representation-based Classification (SRC)

```
1: Input: a matrix of training samples
   $A - [A_1, A_2, \ldots, A_k] \in \mathbb{R}^{m \times n}$ for $k$ classes, a test sample
   $y \in \mathbb{R}^m$, (and an optional error tolerance $\varepsilon > 0$.)
2: Normalize the columns of $A$ to have unit $\ell^2$-norm.
3: Solve the $\ell^1$-minimization problem:
         $\hat{x}_1 - \arg\min_x \|x\|_1$ subject to $Ax - y$.     (13)
   (Or alternatively, solve
         $\hat{x}_1 - \arg\min_x \|x\|_1$ subject to $\|Ax - y\|_2 \leq \varepsilon$.)
4: Compute the residuals $r_i(y) - \|y - A\,\delta_i(\hat{x}_1)\|_2$
         for $i - 1, \ldots, k$.
5: Output: identity$(y) - \arg\min_i r_i(y)$.
```

- Ref. “Robust Face Recognition via Sparse Representation(2009)” -IEEE.

SRC

SRC- on Celebrity Dataset

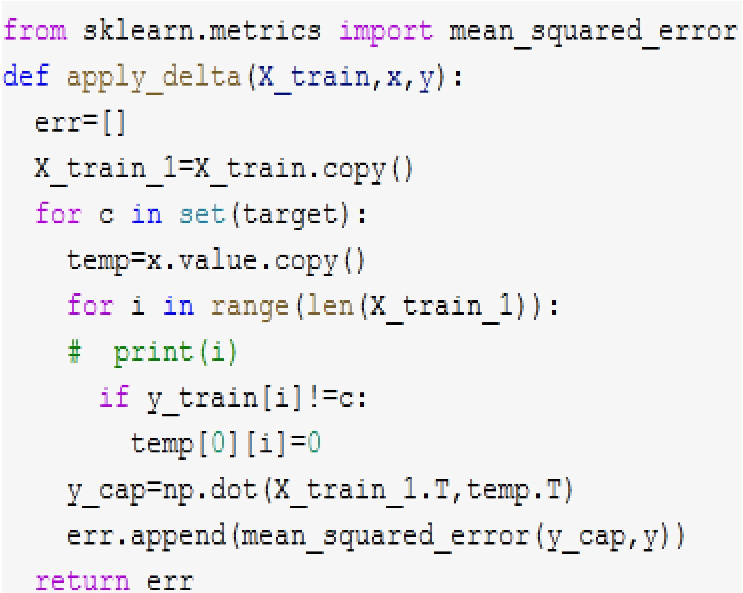


$$\underbrace{\begin{bmatrix} 3 & 155 & \cdots & 255 \\ 19 & 134 & & 102 \\ 19 & 123 & \cdots & 103 \\ 18 & 23 & & 133 \\ . & . & & . \\ . & . & & . \\ . & . & & . \\ . & . & \cdots & . \end{bmatrix}}_{m} \underbrace{\begin{bmatrix} c_1 & 0 & c_3 & c_4 & \cdots & 0 \end{bmatrix}}_{m} = \begin{bmatrix} 130 \\ 255 \\ 107 \\ . \\ . \\ . \\ . \\ . \\ 23 \end{bmatrix}$$

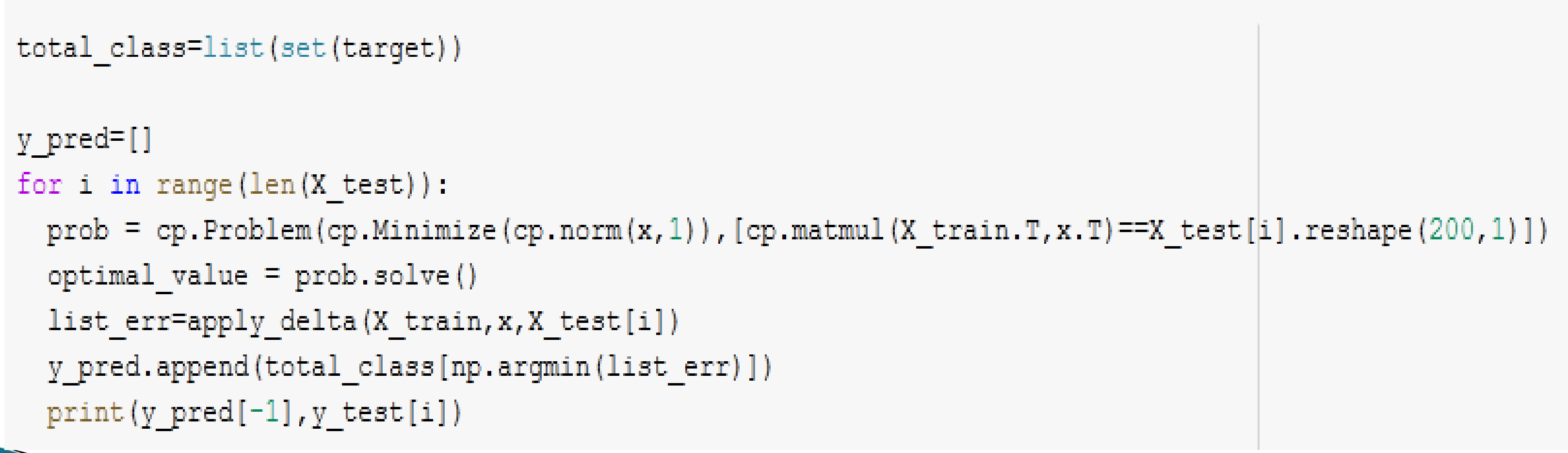
Code:

The Code on the right contains the **function of delta + image prediction + error finding**

The code below **minimizes |x| with respect to Ax=y** and calls delta function and **list out the class** having lower errors



```
from sklearn.metrics import mean_squared_error
def apply_delta(X_train,x,y):
  err=[]
  X_train_1=X_train.copy()
  for c in set(target):
    temp=x.value.copy()
    for i in range(len(X_train_1)):
    #  print(i)
      if y_train[i]!=c:
        temp[0][i]=0
    y_cap=np.dot(X_train_1.T,temp.T)
    err.append(mean_squared_error(y_cap,y))
  return err
```



```
total_class=list(set(target))

y_pred=[]
for i in range(len(X_test)):
  prob = cp.Problem(cp.Minimize(cp.norm(x,1)),[cp.matmul(X_train.T,x.T)==X_test[i].reshape(200,1)])
  optimal_value = prob.solve()
  list_err=apply_delta(X_train,x,X_test[i])
  y_pred.append(total_class[np.argmin(list_err)])
  print(y_pred[-1],y_test[i])
```

Accuracy

- On celebrity dataset we got 42% accuracy. With different variation the stayed between (38 to 42)

- Celebrity dataset also contained backgrounds in images and this adding more irrelevant features to image.

- We also tried SVM for classification and got 61% accuracy.