

the 1990s, the number of people in the world who are under 15 years of age is expected to increase from 1.1 billion to 1.5 billion.

As the world's population grows, the demand for food and other resources will increase. This will put pressure on the environment and on the world's food supply.

One way to meet this demand is to increase the amount of food that is produced. This can be done by using more land for agriculture.

Another way to meet this demand is to increase the efficiency of food production. This can be done by using better farming techniques.

There are many other ways to meet this demand, but the most important is to ensure that everyone has access to food.

This is why it is so important to support sustainable agriculture and to ensure that everyone has access to food.

By doing this, we can ensure that the world's food supply is secure for the future.

It is our responsibility to ensure that everyone has access to food, and we must do everything we can to make this a reality.

Only then can we ensure that the world's food supply is secure for the future.

Let us all join together to ensure that everyone has access to food, and let us ensure that the world's food supply is secure for the future.

Only then can we ensure that the world's food supply is secure for the future.

Let us all join together to ensure that everyone has access to food, and let us ensure that the world's food supply is secure for the future.

Only then can we ensure that the world's food supply is secure for the future.

Let us all join together to ensure that everyone has access to food, and let us ensure that the world's food supply is secure for the future.

Only then can we ensure that the world's food supply is secure for the future.

Let us all join together to ensure that everyone has access to food, and let us ensure that the world's food supply is secure for the future.

Only then can we ensure that the world's food supply is secure for the future.

Let us all join together to ensure that everyone has access to food, and let us ensure that the world's food supply is secure for the future.

Only then can we ensure that the world's food supply is secure for the future.

Let us all join together to ensure that everyone has access to food, and let us ensure that the world's food supply is secure for the future.

Only then can we ensure that the world's food supply is secure for the future.

Let us all join together to ensure that everyone has access to food, and let us ensure that the world's food supply is secure for the future.

Only then can we ensure that the world's food supply is secure for the future.

Let us all join together to ensure that everyone has access to food, and let us ensure that the world's food supply is secure for the future.

Only then can we ensure that the world's food supply is secure for the future.

Let us all join together to ensure that everyone has access to food, and let us ensure that the world's food supply is secure for the future.

Only then can we ensure that the world's food supply is secure for the future.

USING DETECTION METHOD

FUNCTIONS USED IN THE PROGRAM

① $\text{dev}(x, y);$

returns the deviation of x from y .
how it works

1. Assuming $\text{format}(\text{img}) == \text{format}(\text{DB. img}())$
2. Assuming the image is noise free.

② $\text{avg}(\text{arr}[i]);$

returns average of an array

③ $\text{Format}(x);$

returns format of x ;

④ $\text{Concat}(x, y);$

Adds x & y together

⑤ $\text{Convert}(x, y);$

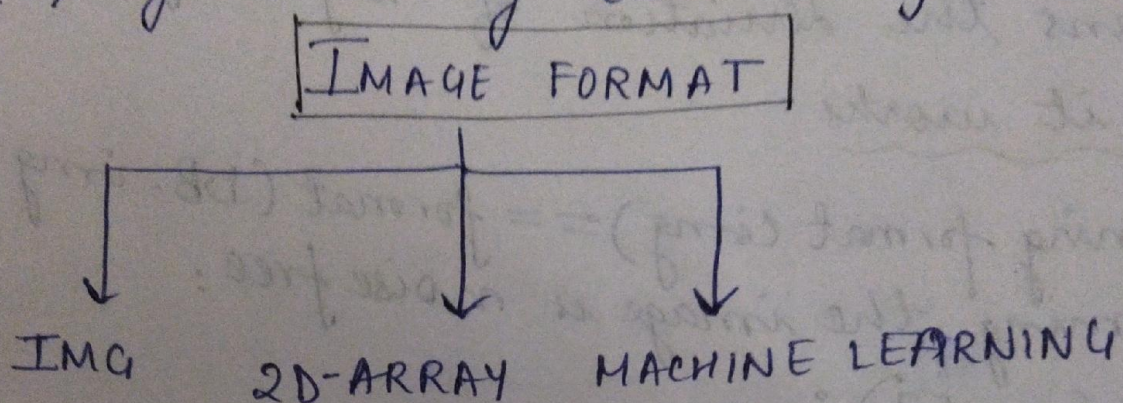
returns converted form of ' x '. i.e. converts x into ' y '. file. where;

$y \Rightarrow \text{format}$

$x \Rightarrow \text{any format input}$

COMMONLY USED TERMS

$Img \Rightarrow$ signature image extracted from cheque.



$Format(DB. img[] == Img)$

~~if not~~ NOTE: If the given condition is false, we would use $(Img, format(DB. img[]))$

$DB. img[] \Rightarrow$ This would store 'n' number of signatures in the database, which acts as an input from the user.

ALGORITHM

(3)

{

dev = db; dev = img; avg = dev; x; // auxiliary variable

flag = 0;

functions()

for (i = 0 to m)

deviation_ar[i] = dev(DB_img[i], DB_img[n-i]);

// deviation of ith element from rest all
// element is stored in the ith position of
deviation_ar[i];

avg_dev = avg(deviation_ar[n]);

// we have average deviation of Database

// stored signatures

// assume from format is matched after conversion
using convert();

for (i = 0 to n)

if (dev(img, DB_img[i]) >

(avg_dev + x))

flag = 1;

break;

// sign is forged now

}

A.I. GENERATION OF SIGNATURES

FUNCTIONS USED

① A.generate(a, x);

generates 'a' no. of signatures deviating
($\pm x$) \Rightarrow range) % from its parent 'A'.

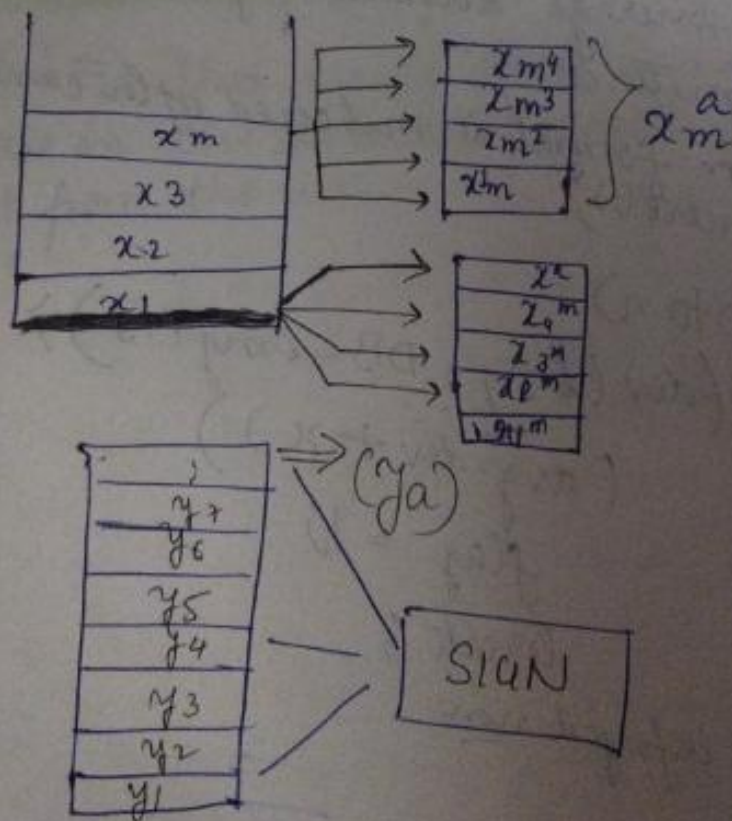
TERMS USED

① SIFT

Scale Invariant Feature Transform. Computer vision algorithm to detect, describe and match local features in image.

PSEUDOCODE FLOW DIAGRAM

AI(img, DB - img[])



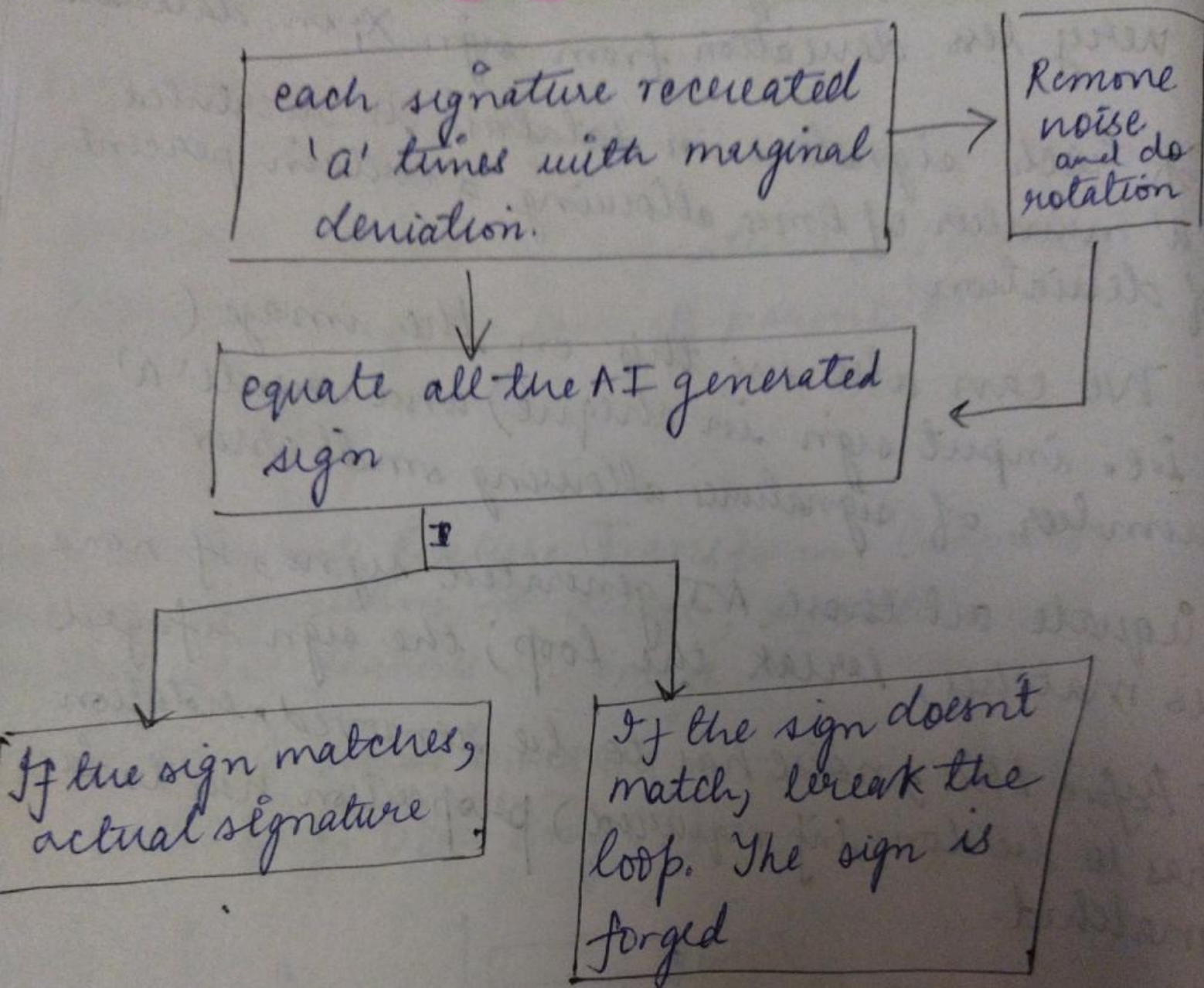
NOTE: Here,

format(img) == format(DB image)

METHOD

- * 'a' number of AI generated signatures having very less deviation from sign X_i in database
- * Each signature in database is reculated 'a' number of times allowing a certain percent of deviation.
- * We can also use this on the image (i.e. input sign in cheque) and create 'a' number of signatures allowing small error.
- * Equate all those AI generated signs, if none is matches, break the loop; the sign is forged
- * Before this, noise has to be removed, rotation has to be done, (if required) proportion has to be matched.

FLOW CHART




```
for (i=0 to n and flag==0) {  
    for (j=0 to a) { // check if signs  
                        matches or not  
        if (match[AI-DB-ar[i][j],  
            AI-unknown-ar[i]]) {
```

```
        for flag=1
```

```
        break; // sign is real
```

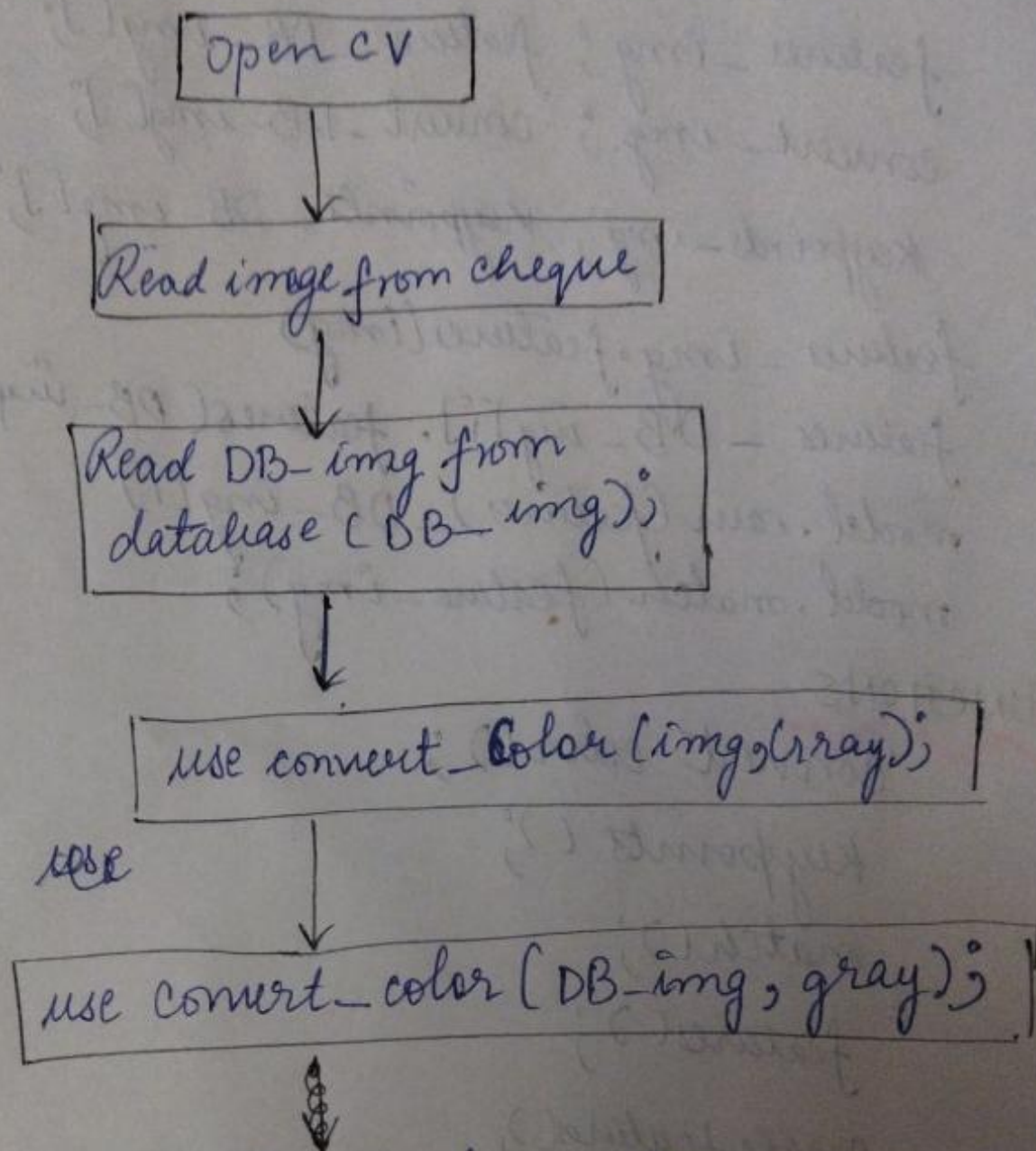
```
    }
```

```
}
```

```
}
```


USING SIFT, SVM AND OPENCV

~~Pseudocode~~
PSEUDOCODE



NOTE: ① We have to store and convert every DB image and store into another array and store into another array.

② store key points using img, DB-img into x_1 & x_2

ALGORITHM

```
// img → cheque sign image  
// DB_img[] → array of Database images,  
features - img; features - DB_img[];  
convert - img; convert - DB_img[];  
keypoints - img; keypoints - DB_img[];  
features - img.features(img)  
features - DB_img[i].features(DB_img[i])  
model.sain(features) - DB_img[i]  
model.match(feature - img);
```

FUNCTIONS

```
convert_color();  
keypoints();  
match();  
features();  
create_features();  
predict();
```