TASK - LBP

create()

```
static Ptr<LBPHFaceRecognizer> cv::face::LBPHFaceRecognizer::create ( int    radius = 1 ,
                                                                      int    neighbors = 8 ,
                                                                      int    grid_x = 8 ,
                                                                      int    grid_y = 8 ,
                                                                      double threshold = DBL_MAX
                                                                    )
```

static

Python:

```
cv.face.LBPHFaceRecognizer_create( [, radius[, neighbors[, grid_x[, grid_y[, threshold]]]]] ) -> retval
```

Parameters

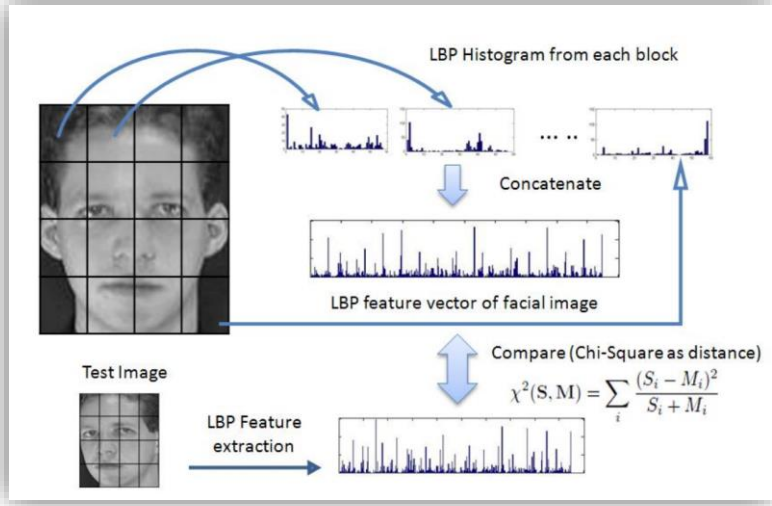
- **radius** The radius used for building the Circular Local Binary Pattern. The greater the radius, the smoother the image but more spatial information you can get.
- **neighbors** The number of sample points to build a Circular Local Binary Pattern from. An appropriate value is to use `8` sample points. Keep in mind: the more sample points you include, the higher the computational cost.
- **grid_x** The number of cells in the horizontal direction, 8 is a common value used in publications. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector.
- **grid_y** The number of cells in the vertical direction, 8 is a common value used in publications. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector.
- **threshold** The threshold applied in the prediction. If the distance to the nearest neighbor is larger than the threshold, this method returns -1.

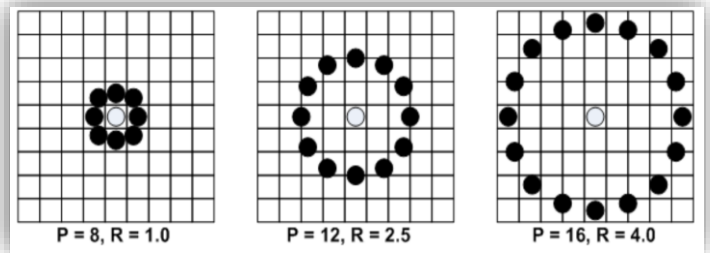
Notes:

- The Circular Local Binary Patterns (used in training and prediction) expect the data given as grayscale images, use cvtColor to convert between the color spaces.
- This model supports updating.

Model internal data:

- radius see **LBPHFaceRecognizer::create**.
- neighbors see **LBPHFaceRecognizer::create**.
- grid_x see LLBPHFaceRecognizer::create.
- grid_y see **LBPHFaceRecognizer::create**.
- threshold see **LBPHFaceRecognizer::create**.
- histograms Local Binary Patterns Histograms calculated from the given training data (empty if none was given).
- labels Labels corresponding to the calculated Local Binary Patterns Histograms.





https://docs.opencv.org/4.x/df/d25/classcv_1_1face_1_1LBPHFaceRecognizer.html

TASK - LBP

predict() [2/3]

void cv::face::FaceRecognizer::predict (**InputArray** src,
int & label,
double & confidence
) **const**

Python:

cv.face.FaceRecognizer.predict(src) -> label, confidence
cv.face.FaceRecognizer.predict_collect(src, collector) -> None
cv.face.FaceRecognizer.predict_label(src) -> retval

Predicts a label and associated confidence (e.g. distance) for a given input image.

Parameters

src Sample image to get a prediction from.
label The predicted label for the given image.
confidence Associated confidence (e.g. distance) for the predicted label.

train()

virtual void cv::face::FaceRecognizer::train (**InputArrayOfArrays** src,
InputArray labels
)

Python:

cv.face.FaceRecognizer.train(src, labels) -> None

labels = [0, 0, 0, 1, 1, 1]
np.array(labels)

Trains a **FaceRecognizer** with given data and associated labels.

Parameters

src The training images, that means the faces you want to learn. The data has to be given as a vector<**Mat**>.
labels The labels corresponding to the images have to be given either as a vector<int> or a **Mat** of type CV_32SC1.

write() [1/2]

virtual void cv::face::FaceRecognizer::write (const **String &** filename) const

Python:

cv.face.FaceRecognizer.write(filename) -> None

Saves a **FaceRecognizer** and its model state.

Saves this model to a given filename, either as XML or YAML.

Parameters

filename The filename to store this **FaceRecognizer** to (either XML/YAML).

read() [1/2]

virtual void cv::face::FaceRecognizer::read (const **String &** filename)

Python:

cv.face.FaceRecognizer.read(filename) -> None

Loads a **FaceRecognizer** and its model state.

Loads a persisted model and state from a given XML or YAML file .

https://docs.opencv.org/4.x/dd/d65/classcv_1_1face_1_1FaceRecognizer.html#aa2d2f02faffab1bf01317ae6502fb631

TASK - LBP

Solve parking lot occupation using LBP:

- Experiment with at least 3 LBP configurations
- For each configuration calculate accuracy and processing time
- You can try to create the combination of the previous detectors + LBP

Training data:
- http://mrl.cs.vsb.cz/data/vyuka/zao/parking/free_full.zip

Solve open/close eye recognition using LBP:

- Experiment with at least 3 LBP configurations
- For each configuration calculate processing time and try to observe accuracy

  - You can use the following video or create new:
    http://mrl.cs.vsb.cz/data/vyuka/zao/fusek_face_car_01.zip

  - **Eye detector**: http://mrl.cs.vsb.cz/data/vyuka/zao/eye_cascade_fusek.zip