REPORT

SIGNS WITH SMART CONNECTIVITY FOR BETTER ROAD SAFETY

Team Id: **PNT2022TMID27042**

Improve efficiency and security while reducing costs

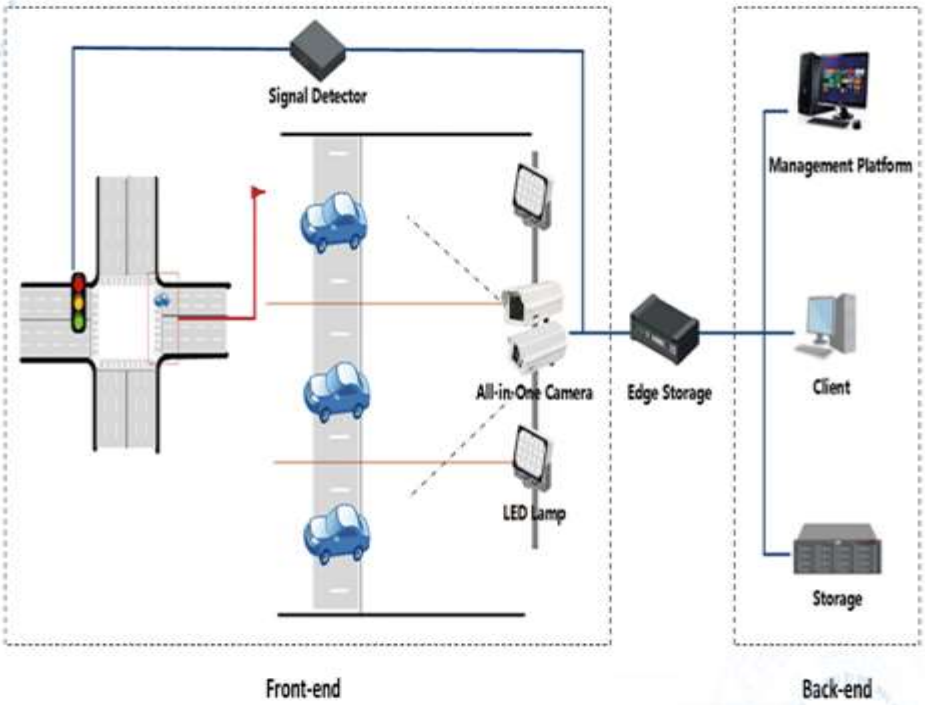
Automatic number-plate recognition(ANPR) is a technology that uses optical character recognition on images to read vehicle registration plates.

Rapid technological developments have allowed DaHua to enable camera-side integration of ANPR technology. This means it can be wide used to improve efficiency and security in defferent solutions. In anddition, this camera-side ANPR is more cost-effective than traditional server-side analysis solutions.

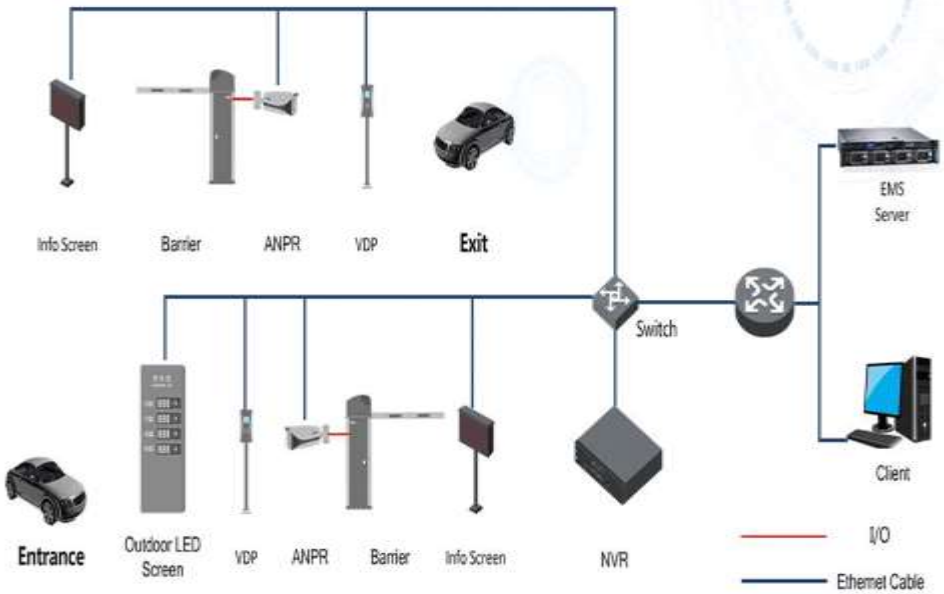
Dahua ANPR solutions are include Dahua cameras and software that runs on either the camera or a server. It automatically captures license plates in real time
Relying on big image data and well train deep-learning models, the ANPR engine includes following features

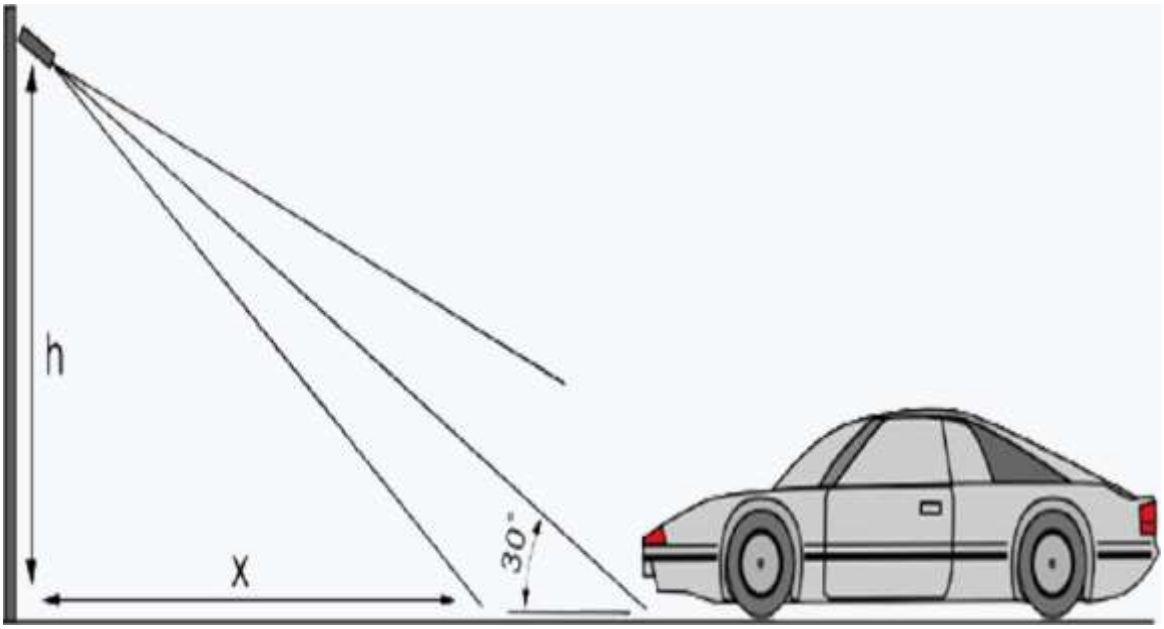
- • Multilingual Recognition Recognizes English, Arabic, Chinese, Cyrillic, Thai, and other languages
- • High Recognition Rate – 95%-98% with alphanumeric number plates
- • Fast Recognition - The full reading process takes around 100 milliseconds
- • Cost-effective Technology - Outstanding price/performance ratio

Road Safety Enforcement



Entrance & Exit Control





At the front-side installation, the horizontal angle should be less than 30°, which means if the height is 6m, the minimum snapshot distance is 10.2m. We recommend a standard horizontal distance of 25m.

In side installations, the angle between the camera and lane line should be less than 30°, which means if the height of pole is 2m, the minimum image capture distance should be at least 7m

```
# Importing all required packages
import cv2
import numpy as np
import matplotlib.pyplot as plt % matplotlib inline


# Read in the cascade classifiers for face and eyes
face_cascade = cv2.CascadeClassifier('../DATA / haarcascades /
haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('../DATA / haarcascades /
haarcascade_eye.xml')



# create a function to detect face
def adjusted_detect_face(img):

    face_img = img.copy()

    face_rect = face_cascade.detectMultiScale(face_img,

    scaleFactor = 1.2,

    minNeighbors = 5)

    for (x, y, w, h) in face_rect:
        cv2.rectangle(face_img, (x, y),
                              (x + w, y + h), (255, 255, 255), 10)\

    return face_img


# create a function to detect eyes
def detect_eyes(img):
```

```
    eye_img = img.copy()
    eye_rect = eye_cascade.detectMultiScale(eye_img,

    scaleFactor = 1.2,

    minNeighbors = 5)
    for (x, y, w, h) in eye_rect:
        cv2.rectangle(eye_img, (x, y),
                      (x + w, y + h), (255, 255, 255), 10)
    return eye_img

# Reading in the image and creating copies
img = cv2.imread('../sachin.jpg')
img_copy1 = img.copy()
img_copy2 = img.copy()
img_copy3 = img.copy()

# Detecting the face
face = adjusted_detect_face(img_copy)
plt.imshow(face)
# Saving the image
cv2.imwrite('face.jpg', face)
```