33. AR Vision

33.1. Overview

Augmented Reality (AR) is a technology that cleverly integrates virtual information with the real world. It widely uses multimedia, three-dimensional modeling, real-time tracking and registration, intelligent interaction, sensing and other technical means to simulate computer-generated text, images, three-dimensional models, music, video and other virtual information and apply them to the real world. The two types of information complement each other, thus achieving "enhancement" of the real world.

The AR system has three outstanding characteristics:

①Integration of information between the real world and the virtual world;

②Real-time interactivity;

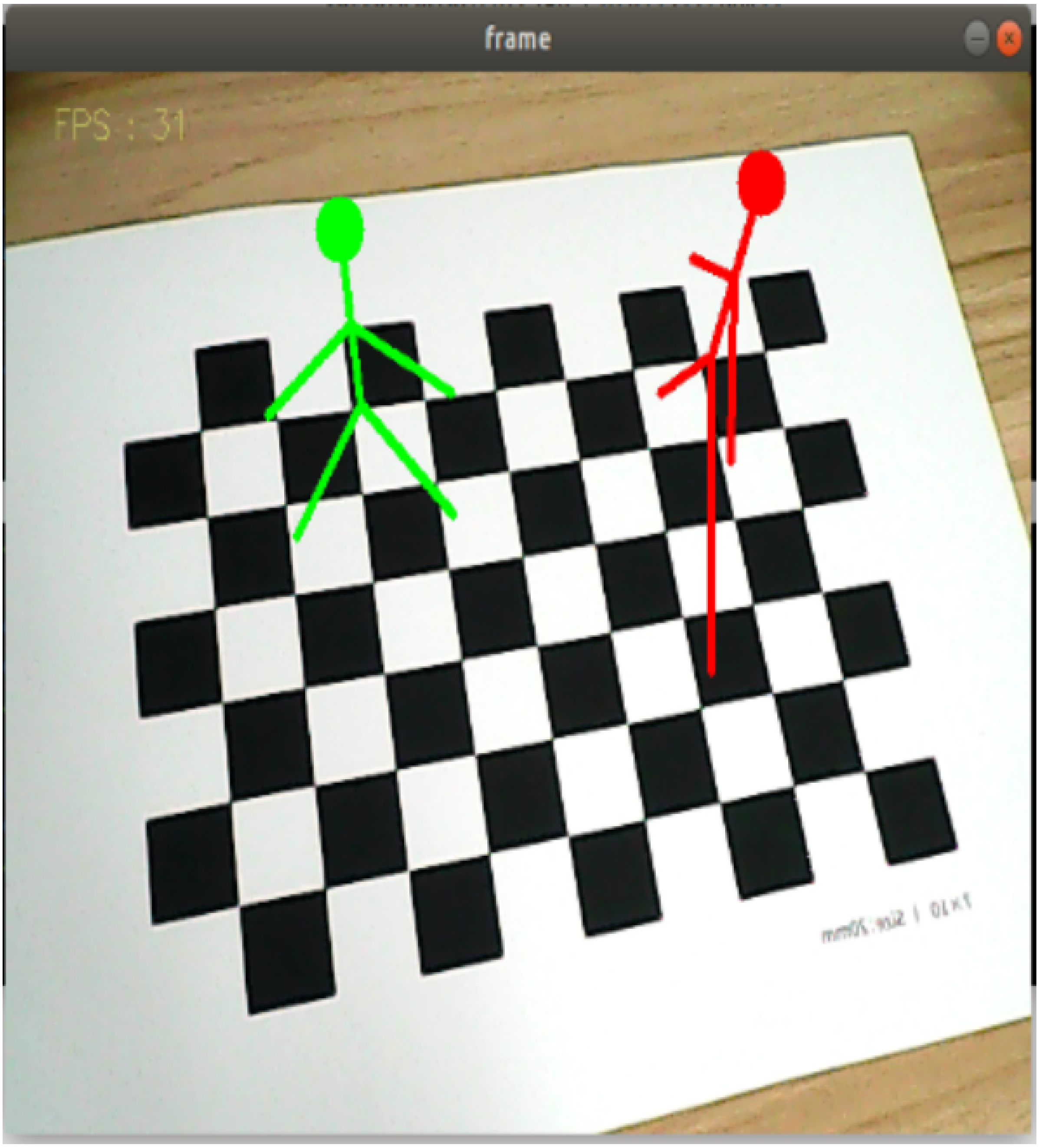
③Adding and positioning virtual objects in three-dimensional space.

Augmented reality technology includes new technologies and new means such as multimedia, three-dimensional modeling, real-time video display and control, multi-sensor fusion, real-time tracking and registration, and scene fusion.

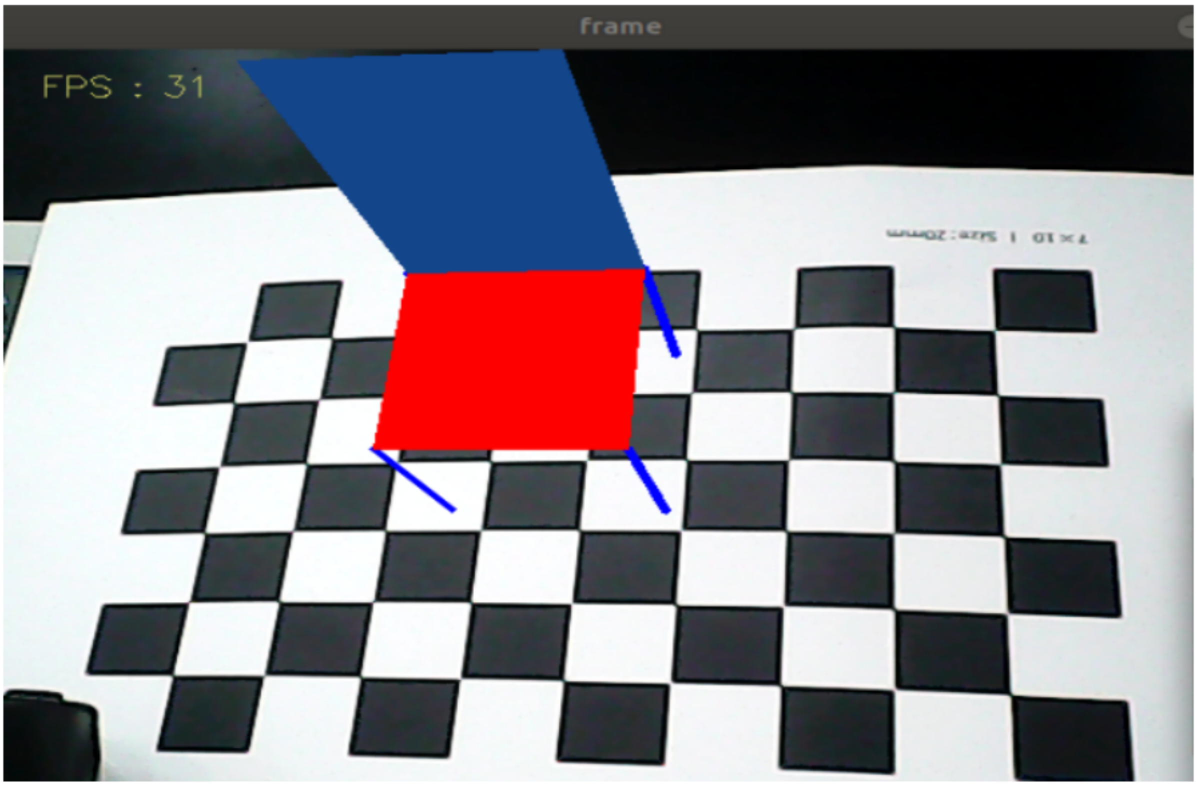
33.2, Startup

Terminal input,

```
roslaunch astra_visual astra_AR.launch VideoSwitch:=True
```



Use the [f] or [F] key to switch between different effects.

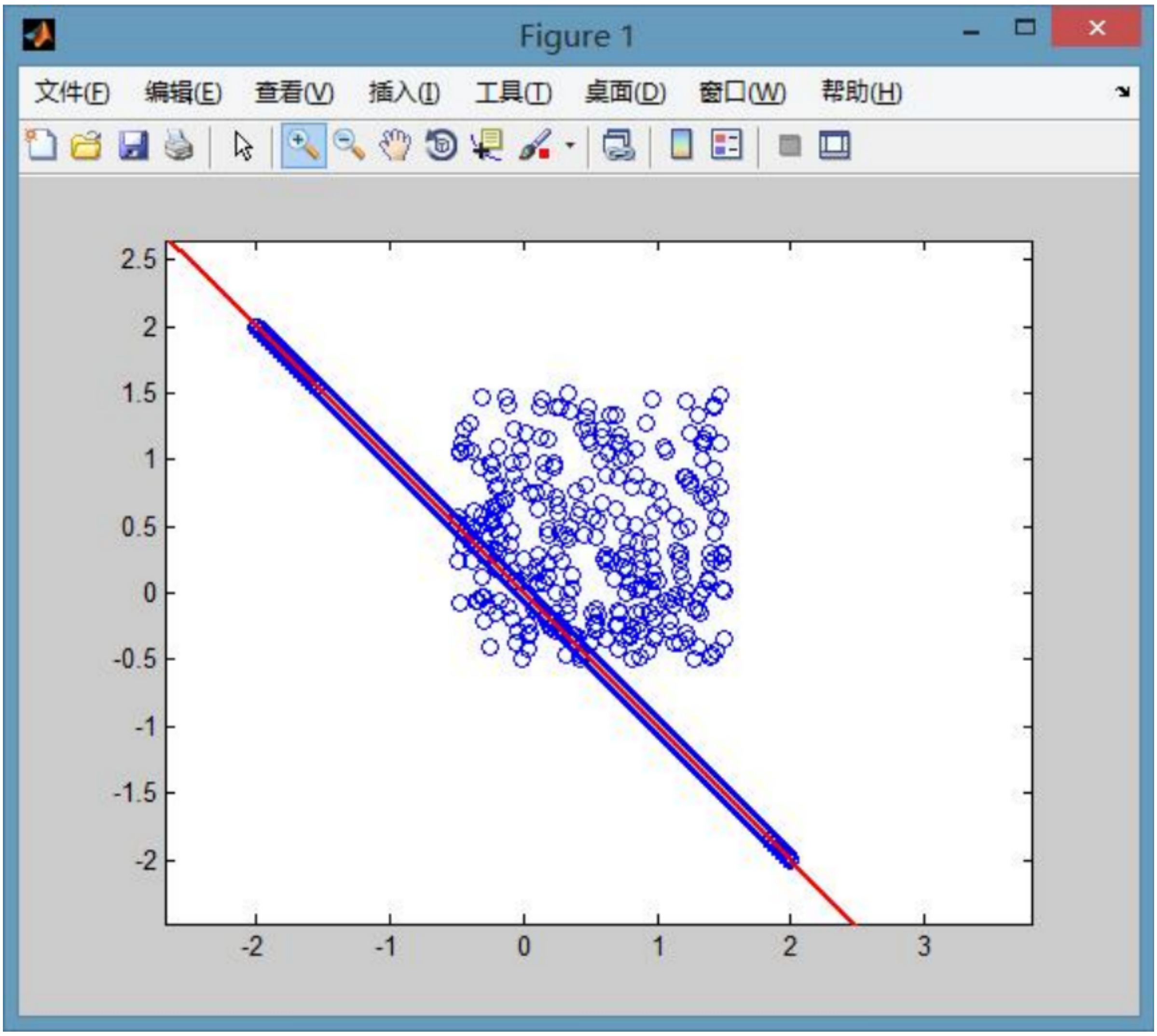


33.3, Source code analysis

33.3.1, Algorithm principle

Use the RANSAC scheme to find the object posture from the 3D-2D point correspondence.

The RanSaC algorithm (random sampling consensus) was originally a classic algorithm for data processing. Its function is to extract specific components in an object under a large amount of noise. The following figure illustrates the effect of the RanSAC algorithm. Some points in the figure clearly satisfy a certain straight line, and another group of points is pure noise. The purpose is to find the equation of the straight line in the case of a large amount of noise. At this time, the amount of noise data is three times that of the straight line.



If the least squares method is used, such an effect cannot be obtained. The straight line will be slightly above the straight line in the figure.

The basic assumptions of RANSAC are:

(1) The data consists of "inside points", for example: the distribution of data can be explained by some model parameters;

(2) "Outside points" are data that cannot adapt to the model;

(3) Data other than these are noise.

The reasons for the generation of outliers include: extreme values of noise; incorrect measurement methods; incorrect assumptions about the data.

RANSAC also makes the following assumptions: given a set of (usually small) inliers, there exists a process that can estimate the model parameters; and the model can explain or apply to the inliers.

33.3.2, core code

Code location: /home/yahboom/orbbec_ws/src/astra_visual/AR

```
def process(self, img, action):
if action == ord('f') or action == ord('F'):
self.index += 1
if self.index >= len(self.graphics): self.index = 0
self.Graphics = self.graphics[self.index]
```

```
if self.flip: img = cv.flip(img, 1)
gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
# Find the corners of each image
retval, corners = cv.findChessboardCorners(
gray, self.patternSize, None,
flags=cv.CALIB_CB_ADAPTIVE_THRESH + cv.CALIB_CB_NORMALIZE_IMAGE +
cv.CALIB_CB_FAST_CHECK)
# Find corner subpixels
if retval:
corners = cv.cornerSubPix(
gray, corners, (11, 11), (-1, -1),
(cv.TERM_CRITERIA_EPS + cv.TERM_CRITERIA_MAX_ITER, 30, 0.001))
# Calculate object pose solvePnPRansac
retval, rvec, tvec, inliers = cv.solvePnPRansac(
self.objectPoints, corners, self.cameraMatrix, self.distCoeffs)
# Output image points and Jacobian matrix
image_Points, jacobian = cv.projectPoints(
self.__axis, rvec, tvec, self.cameraMatrix, self.distCoeffs, )
img = self.draw(img, corners, image_Points)
return img
```

Key functions

- findChessboardCorners()

```
def findChessboardCorners(image, patternSize, corners=None, flags=None):
'''
Find image corners
:param image: Input the original chessboard image. The image must be an 8-bit
grayscale or color image.
:param patternSize: (w,h), the number of inner corners of each row and column on
the chessboard. w=the number of black and white blocks on a row of the chessboard
-1, h=the number of black and white blocks on a column of the chessboard -1.
For example: for a 10x6 chessboard, (w,h)=(9,5)
:param corners: array, the output array of detected corners.
:param flags: int, various operation flags, can be 0 or a combination of the
following values:
CALIB_CB_ADAPTIVE_THRESH Convert the image to black and white using adaptive
thresholding instead of using a fixed threshold.
CALIB_CB_NORMALIZE_IMAGE Histogram equalize the image before binarizing it using
fixed or adaptive thresholding.
CALIB_CB_FILTER_QUADS Use additional criteria (such as contour area, perimeter,
square shape) to filter out false quadrilaterals extracted during the contour
retrieval phase.
CALIB_CB_FAST_CHECK Runs a fast check on the image to find the corners of a
chessboard, returning a quick reminder if no corners are found.
This can greatly speed up calls in degenerate conditions when no chessboard is
observed.
:return: retval, corners
'''
pass
```

- cornerSubPix()

```
def cornerSubPix(image, corners, winSize, zeroZone, criteria):
'''
Sub-pixel corner detection function
:param image: input image
:param corners: pixel corners (both input and output)
:param winSize: region size is NXN; N=(winSize*2+1)
:param zeroZone: similar to winSize, but always has a smaller range, Size(-1,-1)
means ignore
:param criteria: criteria for stopping optimization
:return: sub-pixel corners
'''
pass
```

- solvePnPRansac()

```
def solvePnPRansac(objectPoints, imagePoints, cameraMatrix, distCoeffs,
rvec=None, tvec=None, useExtrinsicGuess=None,
iterationsCount=None,
reprojectionError=None, confidence=None, inliers=None,
flags=None):
'''
Calculate object pose
:param objectPoints: object point list
:param imagePoints: corner point list
:param cameraMatrix: camera matrix
:param distCoeffs: distortion coefficients
:param rvec:
:param tvec:
:param useExtrinsicGuess:
:param iterationsCount:
:param reprojectionError:
:param confidence:
:param inliers:
:param flags:
:return: retval, rvec, tvec, inliers
'''
pass
```