

This forum is disabled, please visit <https://forum.opencv.org>



ALL

UNANSWERED

Search or ask your question 

ASK YOUR QUESTION



OpenCV + OpenGL: proper camera pose using solvePnP

edit



solvePnP

camera

rotation

translation

ios

glkit

I've got problem with obtaining proper camera pose from iPad camera using OpenCV.

I'm using custom made 2D marker (based on [AruCo library](#)) - I want to render 3D cube over that marker using OpenGL.

In order to receive camera pose I'm using solvePnP function from OpenCV.

According to [THIS LINK](#) I'm doing it like this:

```
cv::solvePnP(markerObjectPoints, imagePoints, [self currentCameraMatrix], _userDefaultsManager
.distCoeffs, rvec, tvec);

tvec.at<double>(0, 0) *= -1; // I don't know why I have to do it, but translation in X axis is
inverted

cv::Mat R;
cv::Rodrigues(rvec, R); // R is 3x3

R = R.t(); // rotation of inverse
tvec = -R * tvec; // translation of inverse

cv::Mat T(4, 4, R.type()); // T is 4x4
T(cv::Range(0, 3), cv::Range(0, 3)) = R * 1; // copies R into T
T(cv::Range(0, 3), cv::Range(3, 4)) = tvec * 1; // copies tvec into T
double *p = T.ptr<double>(3);
p[0] = p[1] = p[2] = 0;
p[3] = 1;
```

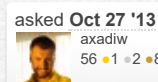
camera matrix & dist coefficients are coming from *findChessboardCorners* function, *imagePoints* are manually detected corners of marker (you can see them as green square in the video posted below), and *markerObjectPoints* are manually hardcoded points that represents marker corners:

```
markerObjectPoints.push_back(cv::Point3d(-6, -6, 0));
markerObjectPoints.push_back(cv::Point3d(6, -6, 0));
markerObjectPoints.push_back(cv::Point3d(6, 6, 0));
markerObjectPoints.push_back(cv::Point3d(-6, 6, 0));
```

Because marker is 12 cm long in real world, I've chosen the same size in the for easier debugging.

As a result I'm receiving 4x4 matrix T, that I'll use as ModelView matrix in OpenCV. Using GLKit drawing function looks more or less like this:

```
- (void)glkView:(GLKView *)view drawInRect:(CGRect)rect {
    // preparations
    glClearColor(0.0, 0.0, 0.0, 0.0);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    float aspect = fabsf(self.bounds.size.width / self.bounds.size.height);
    effect.transform.projectionMatrix = GLKMatrix4MakePerspective(GLKMathDegreesToRadians(39),
    aspect, 0.1f, 1000.0f);
```



asked Oct 27 '13
axadiw
56 1 2 8

Links

- [Official site](#)
- [GitHub](#)
- [Wiki](#)
- [Documentation](#)

Question Tools

Follow

4 followers

[subscribe to rss feed](#)

Stats

Asked:	Oct 27 '13
Seen:	25,896 times
Last updated:	Jan 04 '18

Related questions

[Android camera image rotation](#)[How to decrease the number of processed frames from a live video camera?](#)[Capture Properties](#)[Compiling OpenCV for iOS - no product type error](#)[videofacerec.py example help](#)[opencv2 ios framework error](#)[Collision Avoidance using OpenCV on iPad](#)[Face lifting on iOS](#)[How to limit the number of FPS from camera](#)[Is it possible to use front camera with 2.4.2 for android? \[closed\]](#)

```
// set modelViewMatrix
float mat[16] = generateOpenGLMatFromFromOpenCVMat(T);
currentModelMatrix = GLKMatrix4MakeWithArrayAndTranspose(mat);
effect.transform.modelviewMatrix = currentModelMatrix;

[effect prepareToDraw];

glDrawArrays(GL_TRIANGLES, 0, 36); // draw previously prepared cube
}
```

I'm not rotating everything for 180 degrees around X axis (as it was mentioned in previously linked article), because I doesn't look as necessary.

The problem is that it doesn't work! Translation vector looks OK, but X and Y rotations are messed up :(

I've recorded a video presenting that issue:

<http://www.youtube.com/watch?v=EMNBT5H7-os>

I've tried almost everything (including inverting all axes one by one), but nothing actually works.

What should I do? How should I properly display that 3D cube? Translation / rotation vectors that come from solvePnP are looking reasonable, so I guess that I can't correctly map these vectors to OpenGL matrices.

 add a comment

2 answers

Sort by » [oldest](#) [newest](#) [most voted](#)

Hi axadiw,

I had the same problem and all the samples i found on the net were crappy hacks!

The 3 most important things to know are that :

- solvePnP gives you the transfer matrix from the model's frame (ie the cube) to the camera's frame (it's called view matrix).
- The camera's frame are not the same in opencv and opengl. Y axis and Z axis are inverted.
- How matrixes are stored is not the same neither. Opengl matrixes are column major order whereas they are row major order in Opencv.

So to compute the view matrix (transfer matrix from the model's frame to the camera's frame) that will be used in OpenGL, you have to:

- Use same coordinates to draw the cube in Opengl and to compute the camera's pose with solvePnP (markerObjectPoints)
- build the view matrix like this:

```
cv::Mat rvec, tvec;
cv::solvePnP(objectPoints, imagePoints, intrinsics, distortion, rvec, tvec, ...);
cv::Mat rotation, viewMatrix(4, 4, CV_64F);
cv::Rodrigues(rvec, rotation);

for(unsigned int row=0; row<3; ++row)
{
    for(unsigned int col=0; col<3; ++col)
    {
        viewMatrix.at<double>(row, col) = rotation.at<double>(row, col);
    }
    viewMatrix.at<double>(row, 3) = tvec.at<double>(row, 0);
}
viewMatrix.at<double>(3, 3) = 1.0f;
```

- Multiply the view matrix by the transfer matrix between OpenCV and OpenGL:

answered **Oct 28 '13**

Djo1509
104 • 2 • 6

updated **Jan 5 '18**

Eduardo
3589 • 1 • 15 • 41

```
cv::Mat cvToGl = cv::Mat::zeros(4, 4, CV_64F);
cvToGl.at<double>(0, 0) = 1.0f;
cvToGl.at<double>(1, 1) = -1.0f; // Invert the y axis
cvToGl.at<double>(2, 2) = -1.0f; // invert the z axis
cvToGl.at<double>(3, 3) = 1.0f;
viewMatrix = cvToGl * viewMatrix;
```

- Because OpenCV's matrixes are stored by row you have to transpose the matrix in order that OpenGL can read it by column:

```
cv::Mat glViewMatrix = cv::Mat::zeros(4, 4, CV_64F);
cv::transpose(viewMatrix, glViewMatrix);
glMatrixMode(GL_MODELVIEW);
glLoadMatrixd(&glViewMatrix.at<double>(0, 0));
```


And after that it should work fine ;) Moreover by watching your video, i can notice a shift between the cube and the marker, so i think you probably have calibration problems. Try with default values to see if it's better..

I hope it will be useful ;)

Comments

- 1 Thanks man, it works. Actaully with your answer I've found out that main problem with my solution was that rotation matrix wasn't transposed when it should be. Also whole process of inverting matrices ($R = R.t()$; $tvec = -R * tvec$) wasn't necessary.

I've spend more than a week on that issue, thanks again :)

 [axadiw](#) (Oct 29 '13) [edit](#)


To clarify. Does the transformation matrix produced from the solvePnP function describe where the camera is positioned and facing in my model's coordinate system?

 [ozzyoli](#) (Dec 13 '13) [edit](#)


I would like to point out that when building the viewMatrix, is better to initialize it like

```
cv::Mat viewMatrix = cv::Mat::zeros(4, 4, CV_64FC1);
```

instead of simply `cv::Mat(---)`. That will cause problem especially when going from-to debug release modes. Thanks a lot for the hint anyway!

 [alvisedt](#) (Mar 31 '15) [edit](#)

is the step of multiplying by `cvToGl` not zeroing out all but 4 values?


 [dakom](#) (Nov 22 '18) [edit](#)

Thank you. Which coordination is used to compute solvePnP about ObjectPoints?

 [tjuxh](#) (Jun 11 '0) [edit](#)

I resove the question by transferring the 3d points to opencv coordination (right x down y back z). Then, we get the rotation and translation vector by solvepnp. Like the above, we get the viewMatrix. The difference is `cvToGl.at<double>(2, 2) = 1.0f`. We can render the object in the same view like image.

 [tjuxh](#) (Jun 19 '0) [edit](#)

 [add a comment](#)

Hey , Thank you for the wonderful answer. I have one more query.I have 3x3 camera calibration Matrix, how can i use camera calibration matrix with the openGL projection matrix. I used above glViewMatrix and now I am able see 3d object on my camera screen. but i think there is some issue with size. it may be because i am not doing anything with OpenGL projection matrix

answered Jun 1 '18

 Gurjap
1

```
@Override public void onSurfaceChanged(GL10 unused, int width, int height) { // Set the OpenGL
viewport to the same size as the surface. GLES30.glViewport(0, 0, width, height);
```

```
// Create a new perspective projection matrix. The height will stay the same
// while the width will vary as per aspect ratio.
final float ratio = (float) width / height;
final float left = -ratio;
final float right = ratio;
```

```

final float bottom = -1.0f;
final float top = 1.0f;
final float near = 1.0f;
final float far = 10.0f;

Matrix.frustumM(mProjectionMatrix, 0, left, right, bottom, top, near, far);
}

```

I am calculating glViewMatrix same as you said and saving that in mViewMatrix. by doing this i am able to see my 3d object on the screen. but it has some size issues.

```
Mat glViewMatrix = Mat.zeros(new Size(4,4),CV_64FC1);
```

```

getglViewMatrix(glViewMatrix.getNativeObjAddr());
int index=0;
for(int rows=0;rows<glViewMatrix.rows();rows++){
    for(int cols=0;cols<glViewMatrix.cols();cols++){
        mViewMatrix[index]= (float) glViewMatrix.get(rows,cols)[0];
        index++;
    }
}

```

So my question is how we should use camera calibration matrix to correct the size?

 add a comment