```cpp
#include "Vision3D.h"
#include <cmath>

#include "ros/ros.h"

using namespace std;
using namespace cv;

//Old robot
//const cv::Matx33d Vision3D::cameraMatrix = {512.983, 0.0, 307.130,
    //0, 513.019, 253.954,
    //0, 0, 1};
//const cv::Matx<double, 1, 5> Vision3D::distortionCoefficients = {.20803238, -.352899, .012604, -.0033081, .090205};

//New robot
const cv::Matx33d Vision3D::cameraMatrix = {502.564, 0.0, 309.1057,
    0.0, 502.666, 248.452,
    0.0, 0.0, 1.0};
const cv::Matx<double, 1, 5> Vision3D::distortionCoefficients = {.1402, -.2818, .01328, .00042, .1597};

Point2d Vision3D::getPosIfHeight(Point imagePos, double height) {
  vector<Point2d> initial{{(double) imagePos.x, (double) imagePos.y}};
  vector<Point2d> undistorted;
  undistortPoints(initial, undistorted, cameraMatrix, distortionCoefficients, noArray(), cameraMatrix);   //I was unsure how to use this without it normalizing and denormalizing the values using the camera matrix
  //ROS_INFO("Before: %f, %f, Undistorted: %f, %f", initial[0].x, initial[0].y, undistorted[0].x, undistorted[0].y);

  Matx33d rotationMat = {1, 0, 0,
             0, cos(CameraAngle), sin(CameraAngle),
             0, -sin(CameraAngle), cos(CameraAngle)};    //Rotate clockwise about the x if the positive x axis is facing you
  Point3d dir = rotationMat * (cameraMatrix.inv() * Point3d(undistorted[0].x, undistorted[0].y, 1));
  double t = (-height + CameraHeight) / dir.y;
  Point2d output;
  output.x = t * dir.z + 0.1143;
  output.y = - t * dir.x;
  return output;
}
```