目录

[一、ARToolkit下载与导入 2](#_Toc500669897)

[二、如何校准自己的相机（让ARtoolkit使用自己相机的参数） 2](#_Toc500669898)

[三、标记制作 9](#_Toc500669899)

[1.制作传统模板方形标记 9](#_Toc500669900)

[2.制作符合需求的自然特征图片 10](#_Toc500669901)

[3.制作基于基准标记的自然特征图片 11](#_Toc500669902)

[四、标记训练（Marker Training） 13](#_Toc500669903)

[1. 训练传统模板方形标 13](#_Toc500669904)

[2. 训练自然特征图片（NFT） 17](#_Toc500669905)

[3. 多重标记训练（Multimarker） 21](#_Toc500669906)

[4. 训练基于基准标记的自然特征图片 22](#_Toc500669907)

[五、如何使用四种不同的Marker type作为标记进行识别 23](#_Toc500669908)

[1.使用传统模板标记 24](#_Toc500669909)

[2.使用二维条形码标记 25](#_Toc500669910)

[3.使用多重标记 26](#_Toc500669911)

[4.使用自然特征图片标记（包括基于基准标记的自然特征图片） 29](#_Toc500669912)

[六、使用ARToolkit制作实例的简易流程 30](#_Toc500669913)

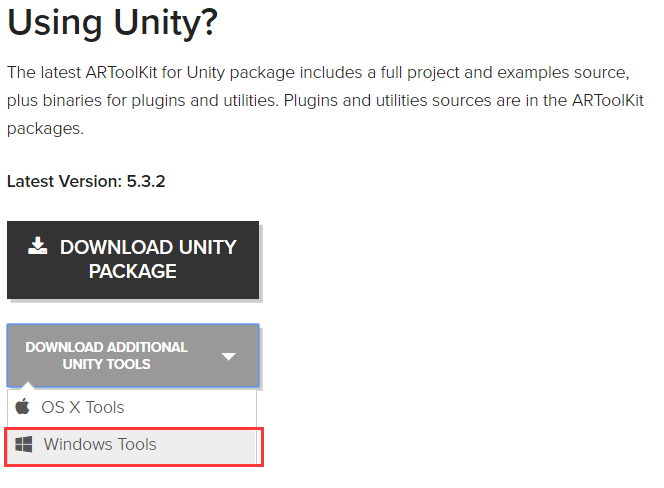
[七、包含ARToolkit的unity场景发布（windows/android）步骤及一些错误处理 30](#_Toc500669914)

[1.发布到windows平台 30](#_Toc500669915)

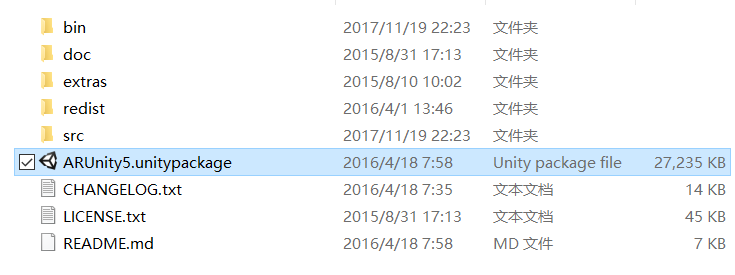
[2.发布到android平台 32](#_Toc500669916)

# ARToolkit下载与导入

下载：登陆ARToolkit官网<https://www.artoolkit.org/download-artoolkit-sdk>，下载ARToolkit package（直接点击DOWNLOAD ADDITIONAL UNITY TOOLS即可，内部已经包含了UNITY PACKAGE）：



导入：上一部下载完成后得到一个压缩文件，解压该文件并打开，双击ARUnity5.unitypackage即可自动导入（或在unity中使用import pakage）



# 如何校准自己的相机（让ARtoolkit使用自己相机的参数）

在ARToolkit的使用中，相机参数文件camera\_para.dat（位于ARUnity5-5.3.2-tools-win\bin\Data下）包含了默认的相机属性，每次启动应用程序的时候都会去读取它，如果不去设置该文件，那么ARToolkit将使用默认值，虽然默认参数足够用于不同相机的基本跟踪需求，但是若想获得最佳的跟踪精度，建议对特定使用的相机进行校准。

相机校准需要用到一个称为Calibration chessboard（校准棋盘）的图案，该图案位于ARUnity5-5.3.2-tools-win\doc\patterns文件夹下，格式为PDF，需要将该PDF文件进行打印，其中有A4和US Letter两种类型，这两种类型是针对各自的打印介质而言的，将该图案打印到A4纸上即可，如下图：



打印完成后，将该打印纸固定到一块薄板上（保持平坦），且测量打印图中一个正方形边长的大小（以毫米为单位）并记录（后面要用），如果在打印的时候没有进行缩放，那么打印出的图案中小正方形的边长刚好是30mm，同时，查看在图纸的X方向（水平方向）有多少个中间角（即两个黑色正方形顶点的交界处），上图中，该值为7，同理，上图中图纸的Y方向（竖直方向）有5个中间角。

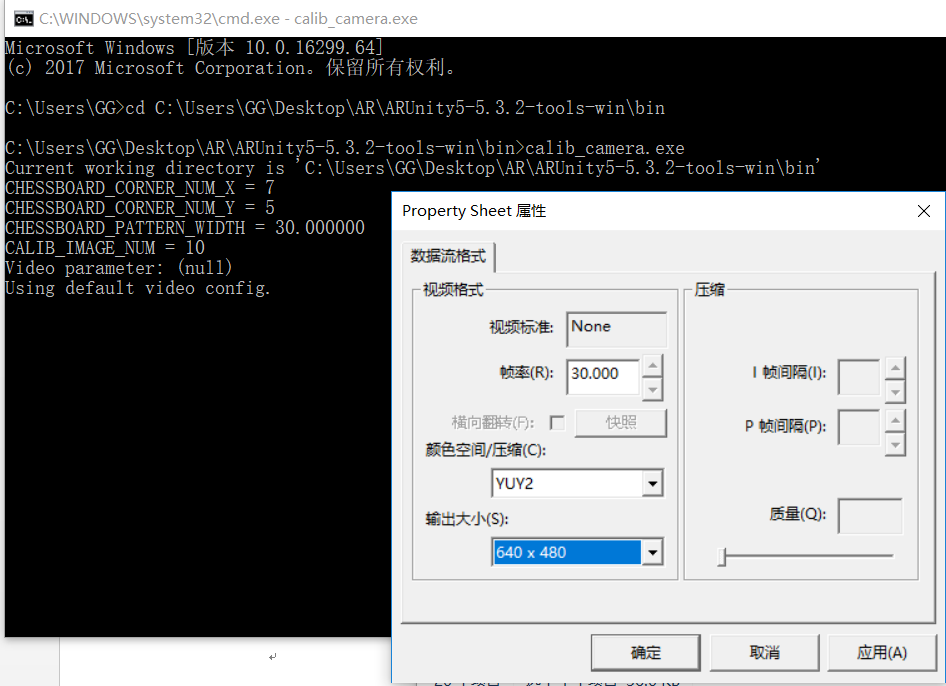
最后对相机进行对焦设置，因为校准文件仅适用于相机的一个对焦设置，如果在校准时和标记追踪时分别使用不同的对焦设置（被允许），将影响标记的追踪效果。最好保证在对相机进行校准和之后的标记追踪时，使用相同的对焦设置。

相机校准的原理：

由于供相机校准使用的图案是由黑白正方体交替构成的网格。所以当通过相机镜头观看时，镜头失真会导致方块边缘的直线出现弯曲。calib\_camera.exe（稍后会使用到，位于ARUnity5-5.3.2-tools-win\bin下）程序使用Opencv库来定位正方形的角点，然后测量角点之间的间距计算镜头的失真。同时，捕获的图像次数越多，捕获的角度越多，失真测量中的误差就越低。

开始校准：

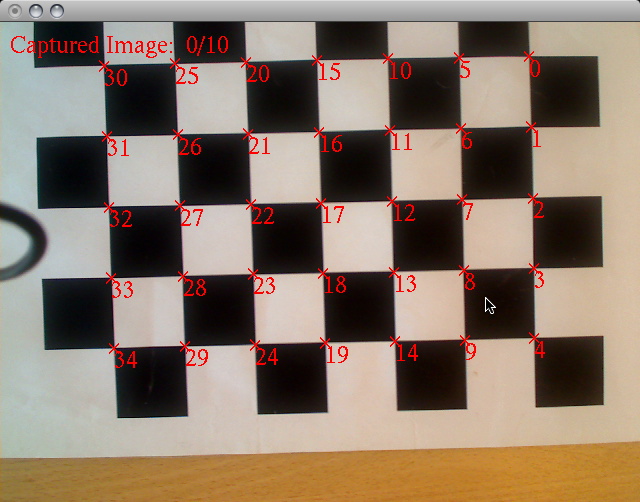
在windows命令行中输入calib\_camera.exe，会产生如下输出：



此处可以为相机选择不同的宽高比，建议选择640×480，且在之后的图片训练和标记追踪时都使用该值。

随后相机需要捕获一系列图像，在捕获窗口的左上角显示目前已捕获的图像数量。把相机指向棋盘格，方块的中间角将用“X”标记和数字标出。

如果相机可以清晰地看到所有的中间角时，“X”标记会变为红色，并且可以捕获校准图像：



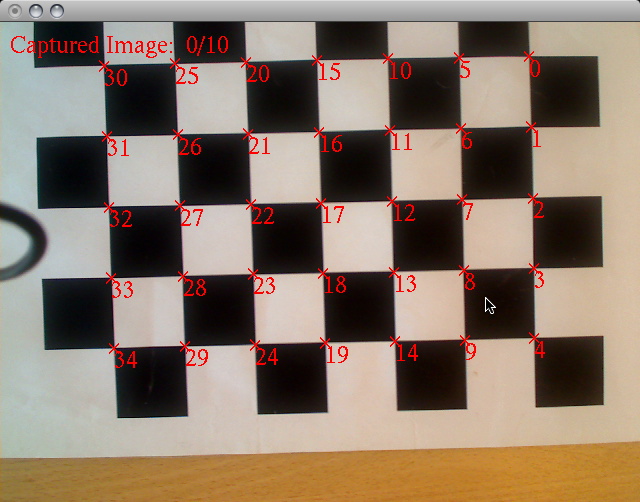
如果某些角落被相机边缘遮挡，或者照明或反射不良，则“X”将

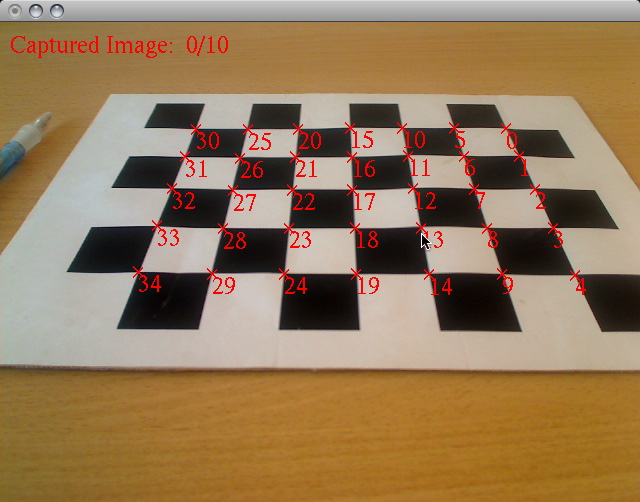
变为绿色：

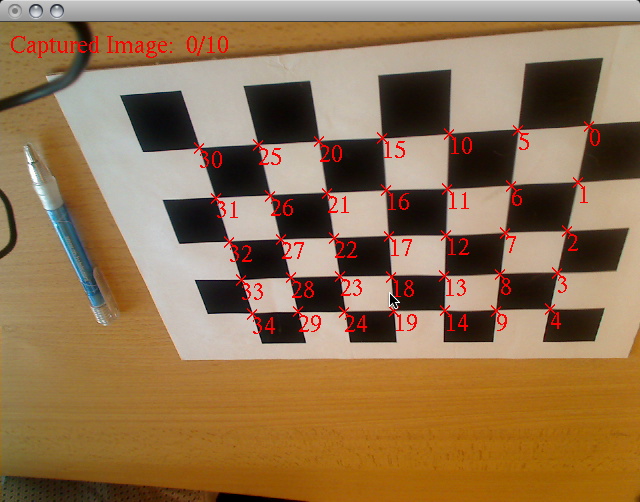


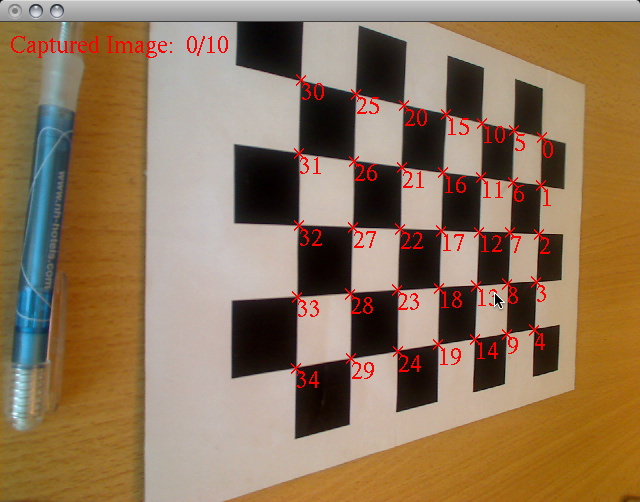
一旦有一个全红的图像，就可以按下空格键完成一次捕获。

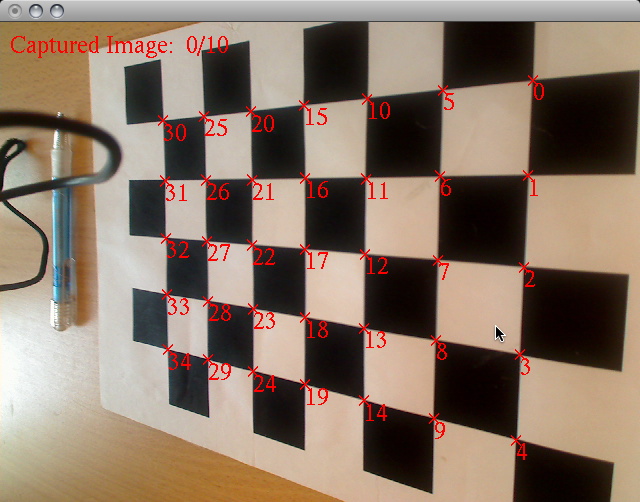
为了更好地校准相机，最好以相机镜头的各种角度获得校准的图像，如下示例角度，包括颠倒：

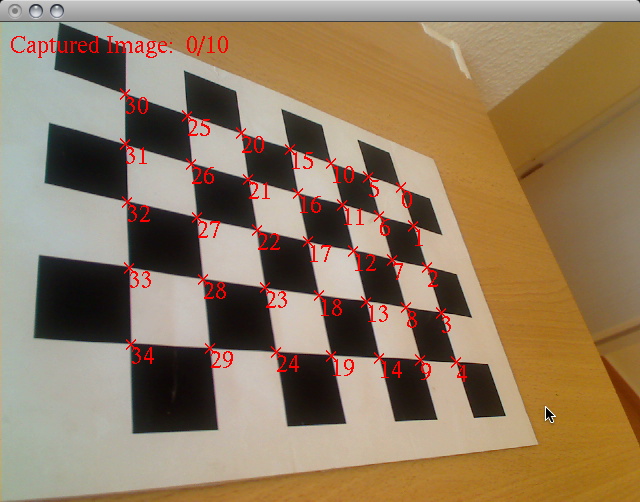




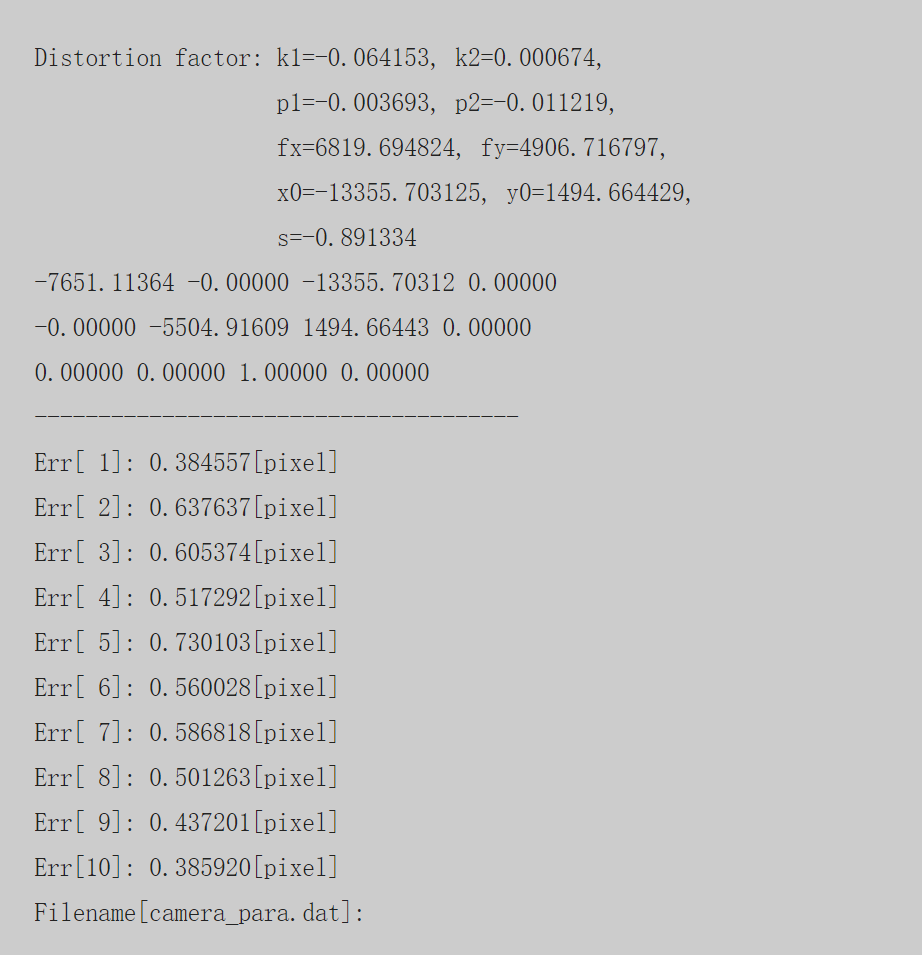








一旦所有的校准图像被捕获完成（10次），校准的数据会被以列表的形式给出，并且相机的失真系数会被自动计算并且输出到标准输出，最后会提示你输入生成的校准文件名，如：



如果校准数据良好，则捕获的每张图像所计算得到的估算误差会比较小，理想值为<1 pixel，

如果误差>2 pixel则说明校准不理想，此时应该进行重新校准。

如果所有的图像校准数据都为理想值，则此时可以输入camara\_para.dat来为该校准文件命名。

最后，将新生成的camara\_para.dat文件覆盖掉ARUnity5-5.3.2-tools-win\bin\Data路径下的同名文件即可。

# 三、标记制作

## 1.制作传统模板方形标记



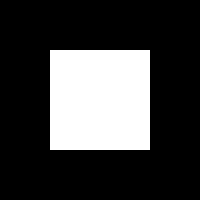
上图是传统模板方形标记的示例，想要制作出符合要求的标记，需要遵循一些规则：

·标记必须是方形。

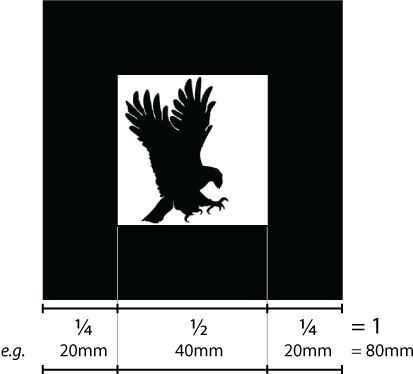
·标记必须有一个连续的边界（通常是全黑或纯白）。并且，位于前景位置的标记，其背景必须是对比色（通常是暗色或浅色）。默认情况下，边框的厚度是标记边缘长度的25%，在上图中，边框即为黑色的部分。

·边界内的区域，称为“pattern”，在上图中即为黑色边框的内部部分，该部分必须具有旋转不对称性，pattern既可以是黑白的也可以是彩色的（且如果为彩色的话ARToolkit能更加准确地跟踪）。

为了帮助制作传统模板方形标记，ARToolkit提供了位于ARUnity5-5.3.2-tools-win\doc\patterns文件夹下的Blank pattern.png标记模板图像文件，如下图所示：



通过编辑该模板图像文件，可以设计和创建新的标记，且标记可以缩放到任意大小，下图是一个设计的例子：



标记的内部50%被解释为ARToolkit的标记图像，正如之前所述，图像既可以是黑白的，又可以是彩色的，并且可以延伸到“边界”区域。但是内部50%以外的图像部分将被ARToolkit忽略，不要过度将pattern内的内容延伸到边界，否则可能造成无法识别标记的后果。

## 2.制作符合需求的自然特征图片

自然特征追踪NFT（natural feature tracking）是基于自然图片作为标记来进行的识别和跟踪。由于自然图片的选择具有任意性，所以为了使ARToolkit能正常工作，需要让ARToolkit预先知道该图片的一些特征数据信息，这称为图像的训练，因此，制作出符合需求的高质量图片是后续训练的基础。

制作ARToolkit的自然图片同样需要符合一些要求：

·图片的外形必须是矩形的，即矩形图片

·图像必须以jepg的格式提供

·图片最好具有较低的自相似性和高的空间频率。大面积的单一平面色彩，模糊或具有柔和细节的图像将不能很好地跟踪。在这样的图像中，很难定位不同的特征点。

·更大或更高分辨率的图像（更高像素）将允许ARToolkit以更高的级别提取特征点细节，这样当相机靠近图像时能同样保持较好的跟踪效果。

针对NFT最常见的情况是：在网上寻找或制作一张jepg格式的图片用作稍后的图片训练，并将该图片进行打印出来用作之后的标记追踪，如下图：



## 3.制作基于基准标记的自然特征图片

制作基于基准标记的自然特征图片与NFT类似，目的是为了在自然图片内部或外部周围通过嵌入标记来增强图像的识别和追踪。

使用基于基准标记的自然特征图片追踪具有如下优点：

·使用NFT 1.0跟踪器加基准标记比NFT 1.0 + 2.0完全无标记跟踪在计算上花费更少。

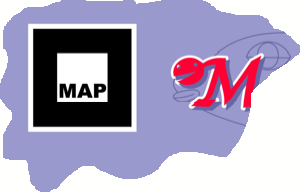
·NFT 2.0追踪器对于任何时候都可以区分的不同标记的数量都有实际的限制。因此，如果需要跟踪大量图像，则基准标记能够有效识别意图被跟踪的多个图像。

·基准标记跟踪增加了对跟踪的健壮性，特别是在光线不足的情况下，或者当相机远离跟踪的图像时。

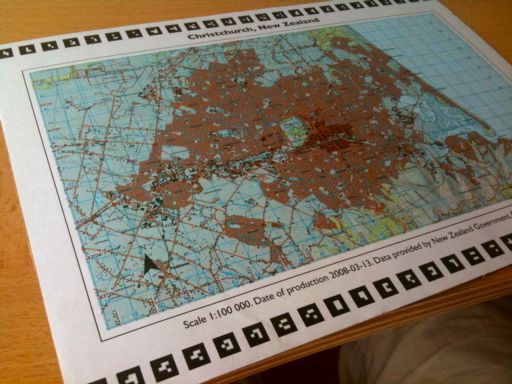
基准标记的外观和放置：

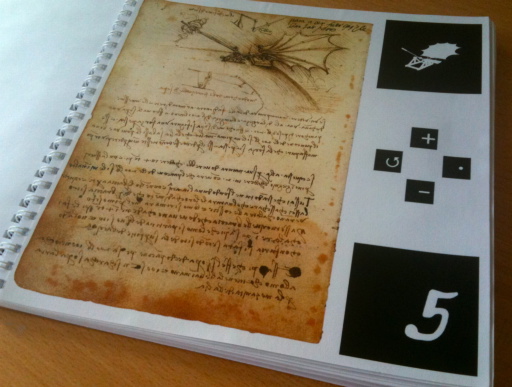
对于嵌入的标记，必须是方形的，且嵌入的标记和背景之间需要有明显的对比色，标记可以是随意大小，并且标记的内部可以嵌入与标记所在图像背景混合的彩色图案。

标记可以使用黑色或白色的边框，如果使用黑色边框，则标记必须位于白色或者浅色的背景区域（如果有必要甚至可以在黑色边框外部添加一层额外的白色边框），如果使用白色边框标记必须位于黑色或深色背景区域。标记的内半部分形成独特的部分，即如果标记有80mm宽的标记，则内部40mm的垂直和水平尺寸以内都是独特的部分。



除了标记可以位于NFT表面的图像部分内，它也可以位于图像外部周围：





同时，如果标记内部的内容为所嵌入背景图像的一部分，那么该标记在增强识别的同时，在图像中看起来也不会那么突兀：

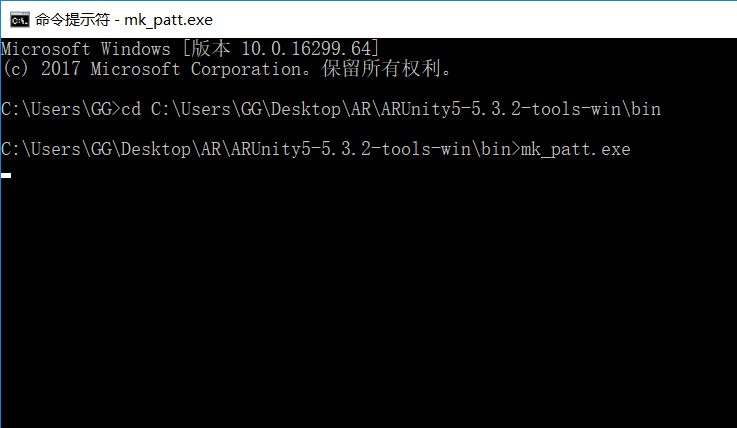


# 标记训练（Marker Training）

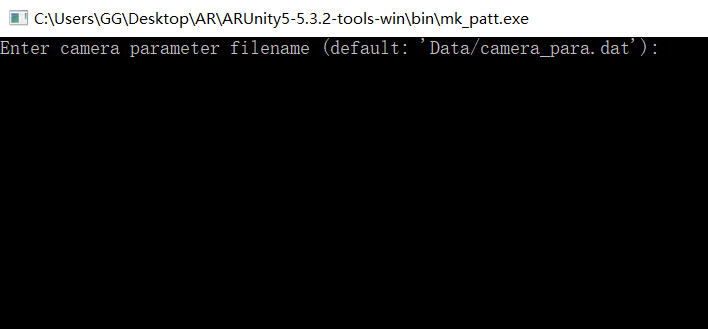
## 训练传统模板方形标

首先将制作好的marker打印到A4纸上。

使用路径ARUnity5-5.3.2-tools-win\bin文件夹下的mk\_patt.exe文件对marker进行训练，在命令行（win+cmd）中执行mk\_patt.exe：



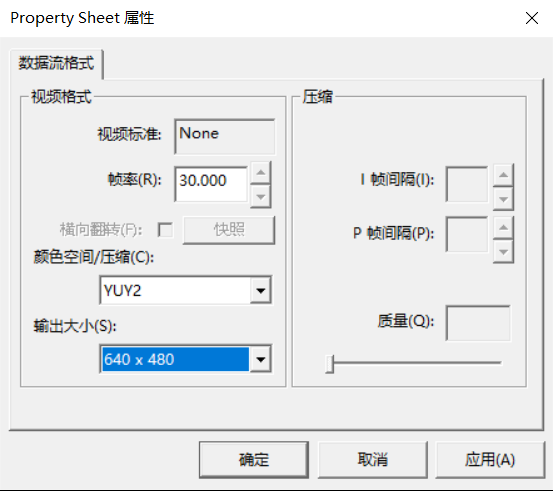
此时会自动产生另一个终端，提示输入相机校准参数（以文件的格式输入）：

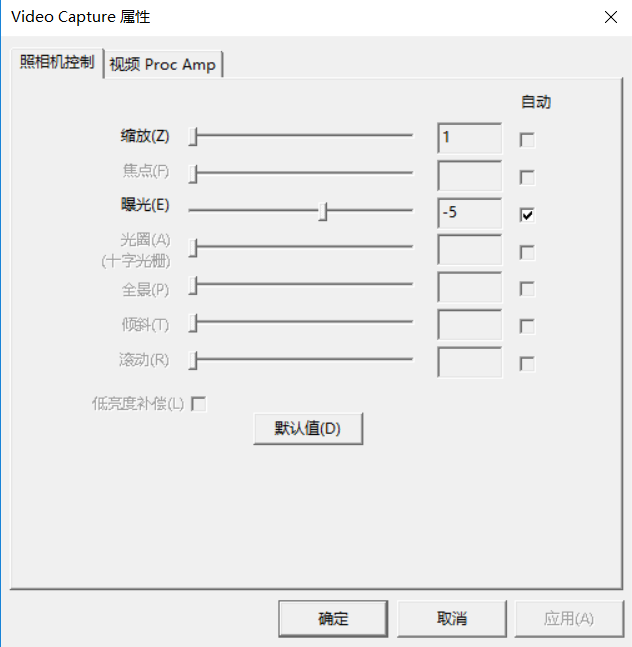


回车即可，程序默认会读取位于ARUnity5-5.3.2-tools-win\bin\Data文件夹下的相机校准文件

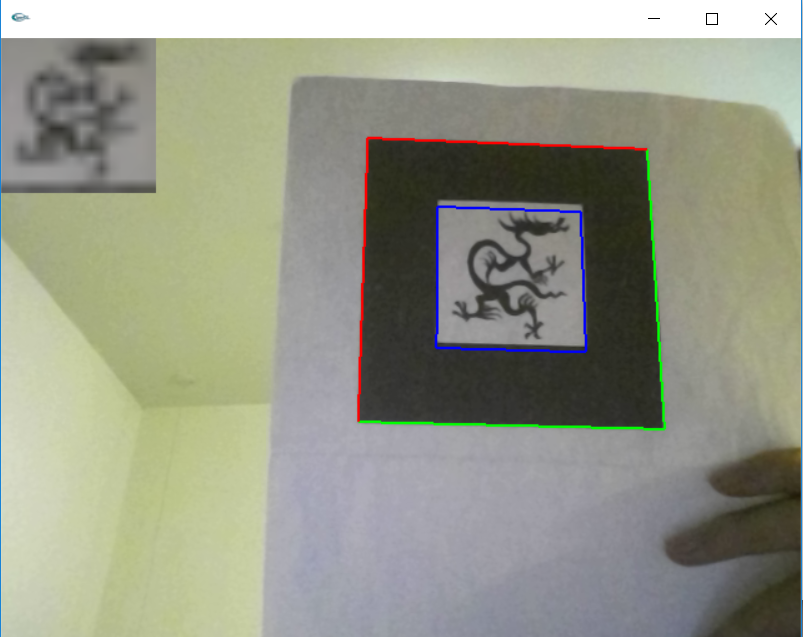
camera\_para.dat

随后选择摄像头的相关配置信息：





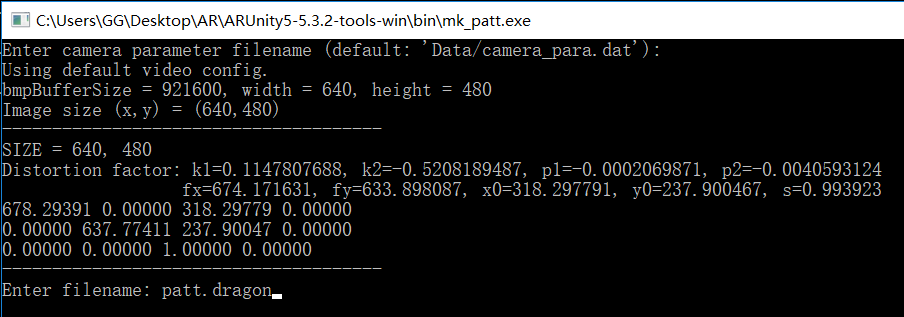
随后，摄像头被激活：



把相机对准marker，如果ARToolKit已经识别标记，它会用红线和绿线勾勒出来。

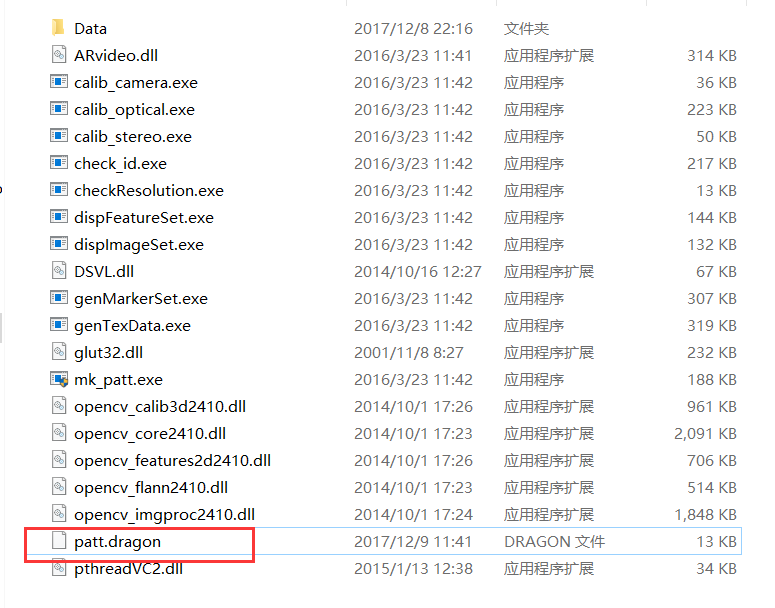
旋转marker，让红线角落标记在marker的左上角，绿线角落标记在marker的右下角，随后点击鼠标左键完成图像捕获。

随后，终端提示要求输入文件名：

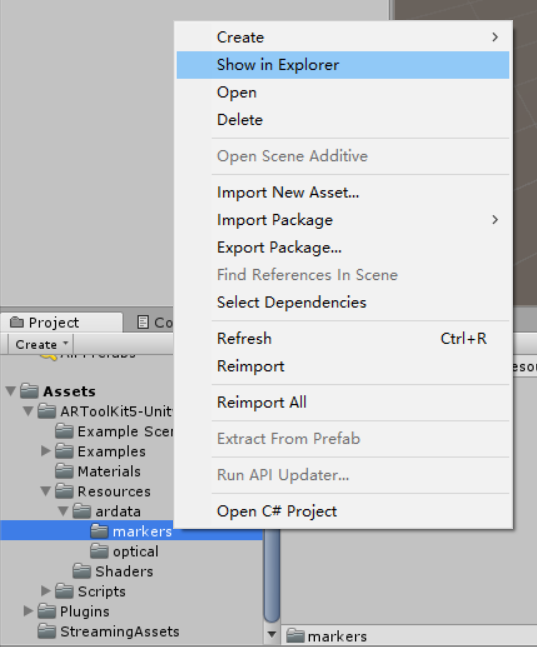


输入一个以“patt.”为前缀的目标文件名，如上图patt.dragon并回车

通过以上步骤，便在bin文件下生成了相应文件：



最后，将该文件更名为patt.dragon.txt(更改成txt文件格式)，随后打开unity工程，定位到Assets\ARToolKit5-Unity\Resources\ardata\markers文件夹下，将patt.dragon.txt放到该文件夹下即可。

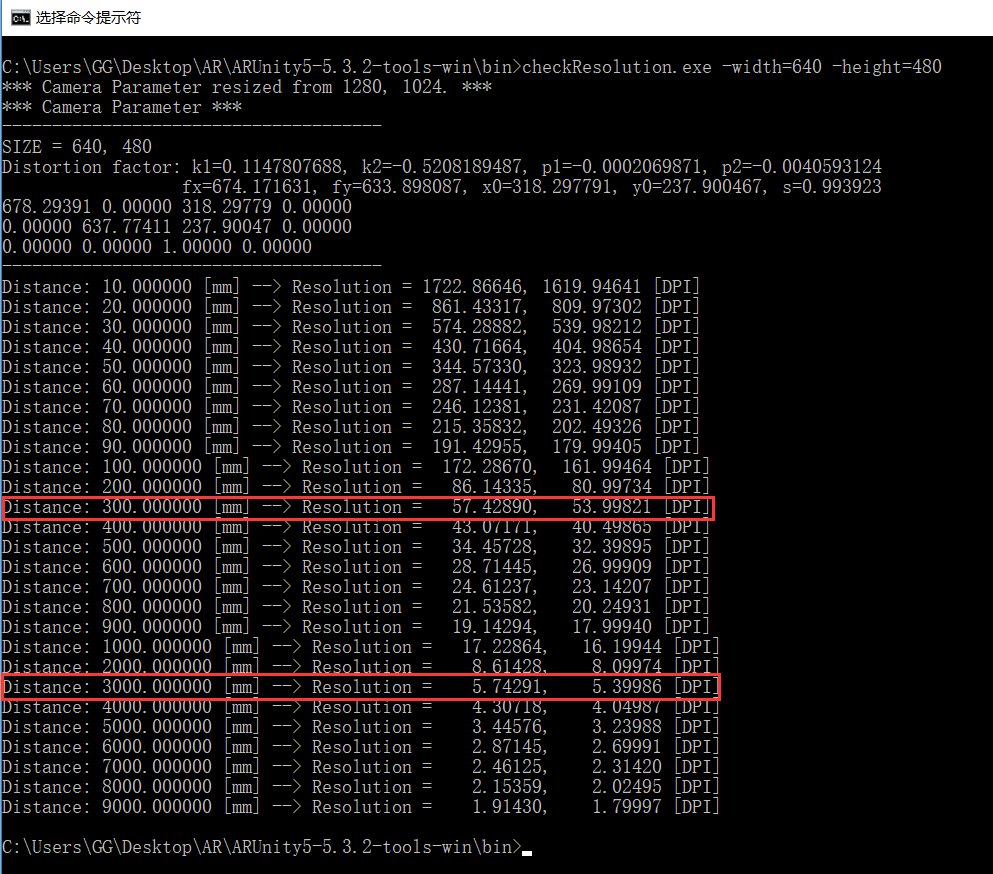


## 训练自然特征图片（NFT）

要想得到较好的NFT训练结果，需要首先确定两项数据：

1. 打印出来的图片实际尺寸为多大
2. 在对该图片进行识别和追踪时，摄像头与该图片之间的距离大约处于一个什么样的范围

首先在ARUnity5-5.3.2-tools-win\bin文件夹下找到checkResolution.exe，在命令行中执行，并且附带一个摄像头输出画面宽高比的参数640×480（如果不带该参数，默认为1280×1024）



可以在输出的不同距离对应的DPI参考值之间选择适合的数值，比如实际识别和追踪NFT时，摄像机与图片的距离大约是30cm-3m，故可以得到对应的DPI范围为5-58。

确定距离范围后，第二步需要确定打印出来的图片尺寸大小，之所以需要确定实际尺寸，是为了求出图片的打印DPI为多大，用作后续步骤的输入。虽然一般的彩色打印机的打印DPI为150，但是由打印出的实际图片经尺寸测量并由公式求解得出往往打印DPI并不是这个值。

关于打印DPI，有：图片分辨率=实际尺寸（单位：英寸）×打印DPI

举个例子，比如有如下图片：



经测量，其实际尺寸为宽度：18.9cm=18.9/2.54=7.441英寸，高度：23.65cm=23.65/2.54=9.331英寸 （注：1英寸=2.54cm）

图片的分辨率可以通过查看图片的属性得到：



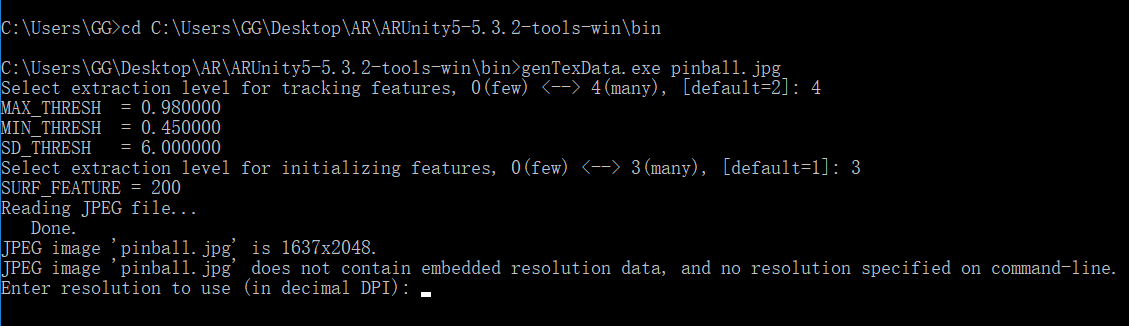
由上图可以看到，该图片的分辨率为1637×2048，故可以计算出打印DPI为

1637/7.441=220

2048/9.331=219.5，故打印DPI为220×220，与图片的参考DPI一致。

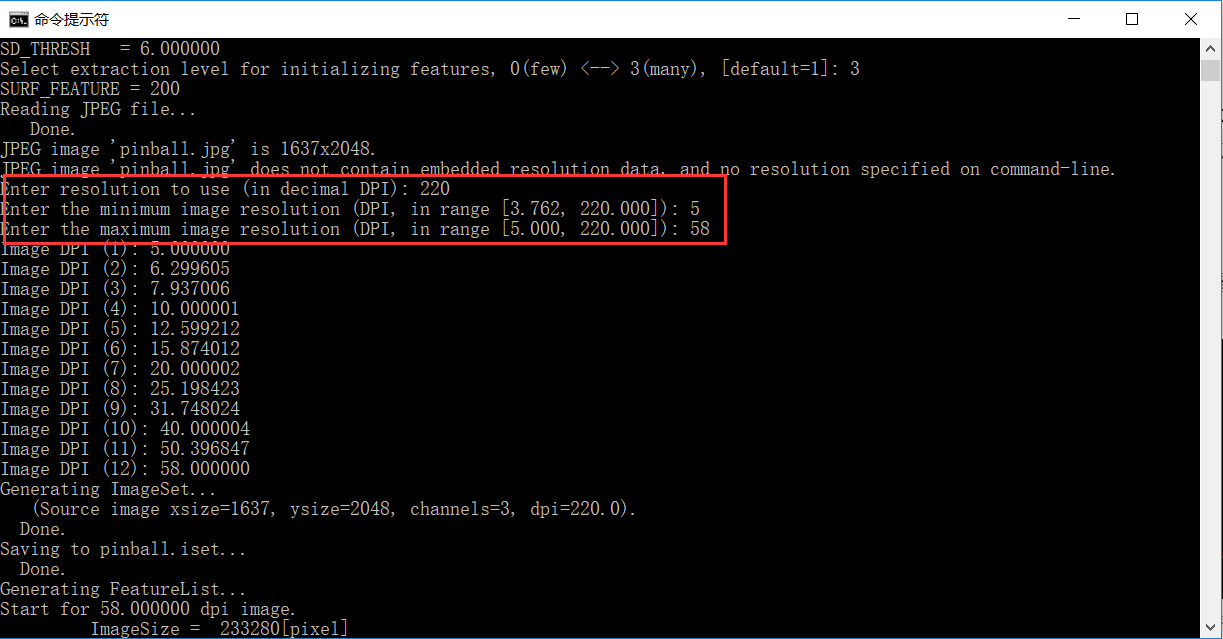
虽然计算得到的打印DPI与参考DPI一致，不过这两者不是恒等的关系，打印DPI主要还是取决于对图片的打印要求，在分辨率不变的情况下，打印的DPI越大，打印出的图片尺寸越小，反之亦然。

得到了图片的打印DPI之后，在命令行中执行ARUnity5-5.3.2-tools-win\bin文件夹下的genTexData.exe，并且以需要训练的NFT图像作为输入（需要事先将图片放到与genTexData.exe同一文件夹下），如下图：

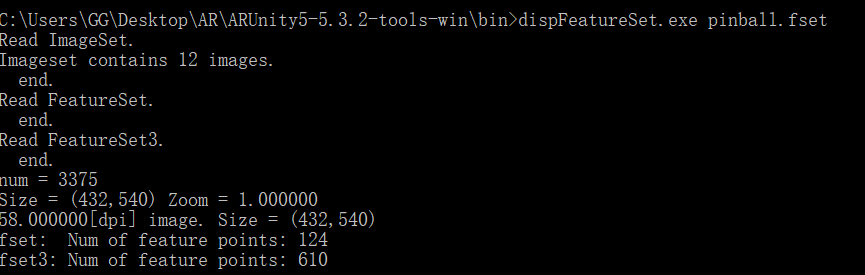


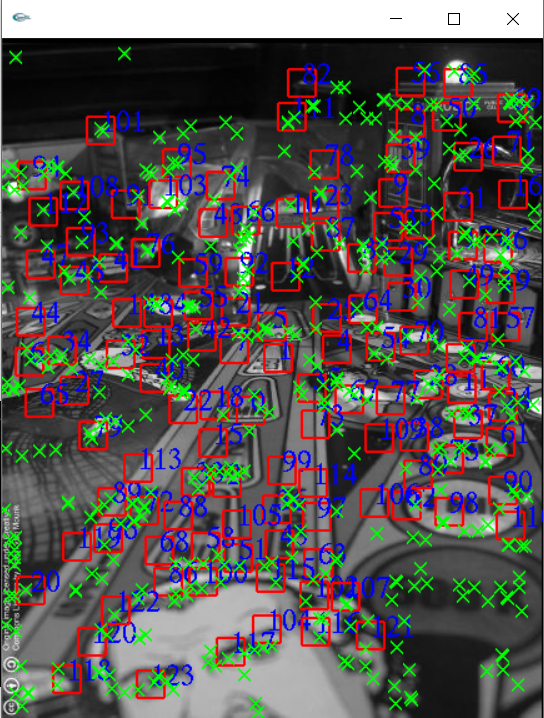
由于NFT跟踪是基于图片的特征点信息，所以在执行genTexData.exe后会提示你输入提取该图片特征点信息的等级和初始化特征点信息的等级，分别选择4和3（代表many）以达到较好的提取效果。

紧接着，要求你输入对应图片的打印DPI及之前由checkResolution.exe产生的与识别距离相关的DPI信息（以pinball.jpg为例）：

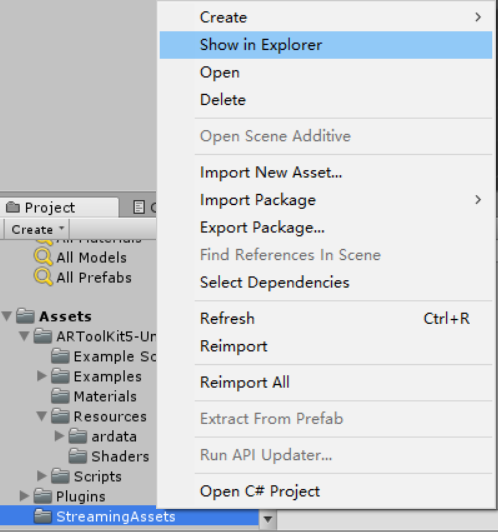


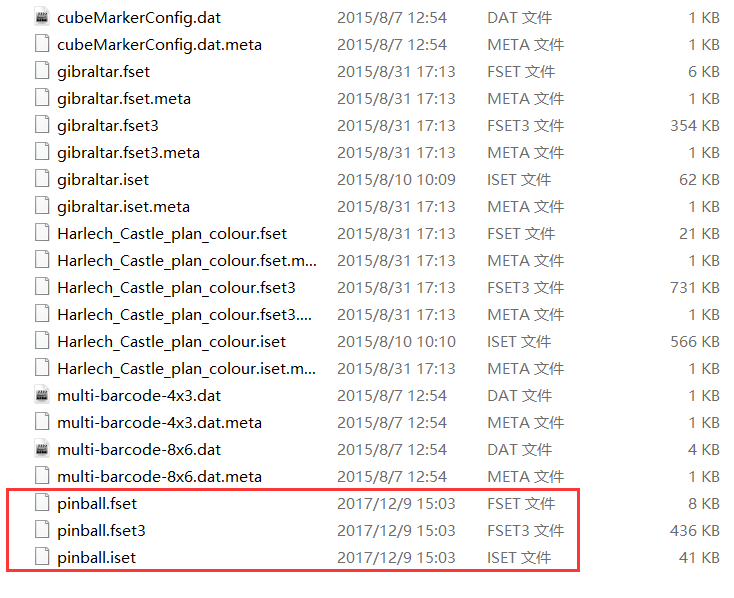
经过上述步骤，便可以在bin文件夹下获得3个与图片对应的数据集文件：pinball.fset、pinball.fset3、pinball.iset，另外可以使用dispFeatureSet工具显示特征点提取情况：





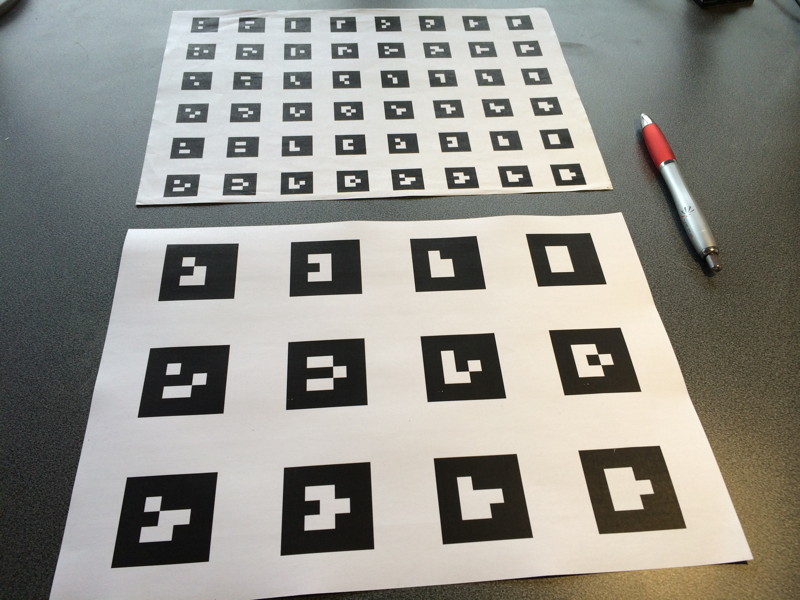
最后，将得到的3个数据集文件放到Assets\StreamingAssets文件夹下：





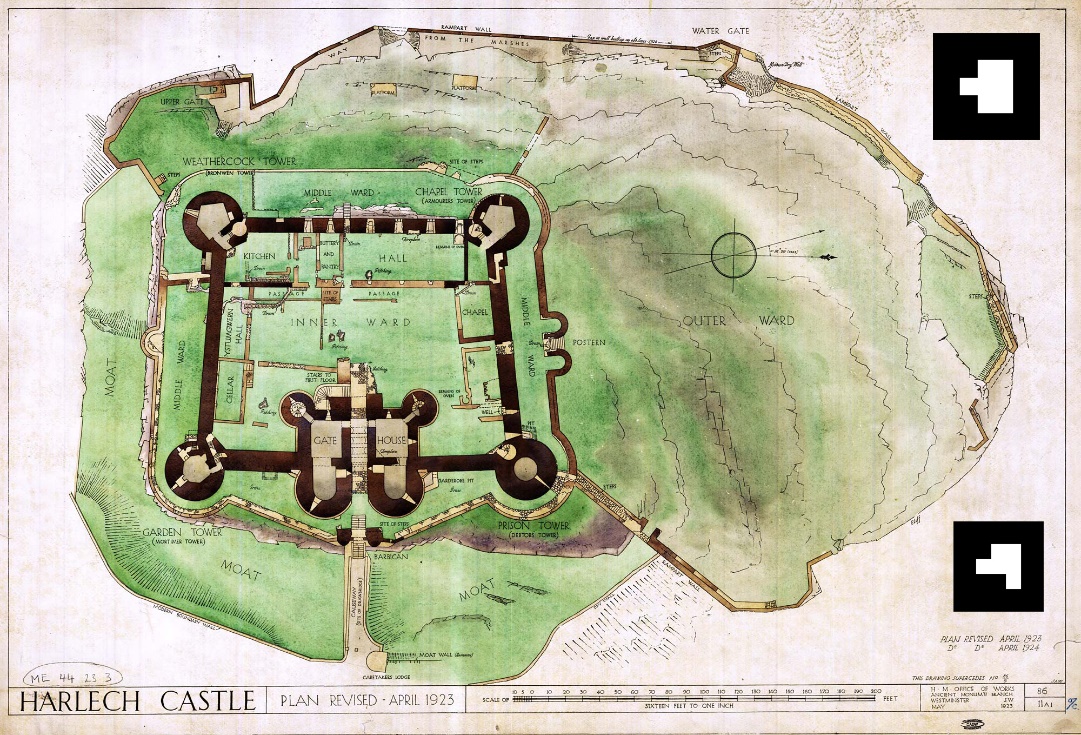
## 多重标记训练（Multimarker）

多重标记其实并不需要训练，这里加入多重标记主要是为了使分类完整，因为ARToolkit默认使用一套内置的具有序号的marker（称为barcode）集合来使用mutimarker，这个barcode集合在ARUnity5-5.3.2-tools-win\doc\patterns\Matrix code 3x3 (72dpi)文件夹下可以看到，如下：



每个barcode具有对应的序号，这是后续使用barcode和multimarker的基础。

## 训练基于基准标记的自然特征图片

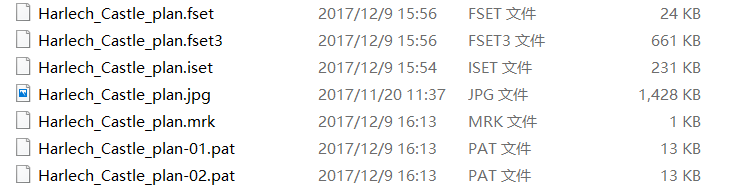


基于基准标记的自然特征图片训练与NFT训练类似，在按照训练NFT的方法训练以上图片得到.iset、.fset、.fset3文件后，此时需要用到genMarkerSet.exe工具，以图片对应的.iset文件作为输入如下：



此时你会看到2个数字，第一个是从图片中识别到的候选标记数量，第二个是满足成为基准标记条件的标记数量，此时按下空格键，并且输入“y”+回车表示接受标记，会得到.mrk文件和.pat文件：

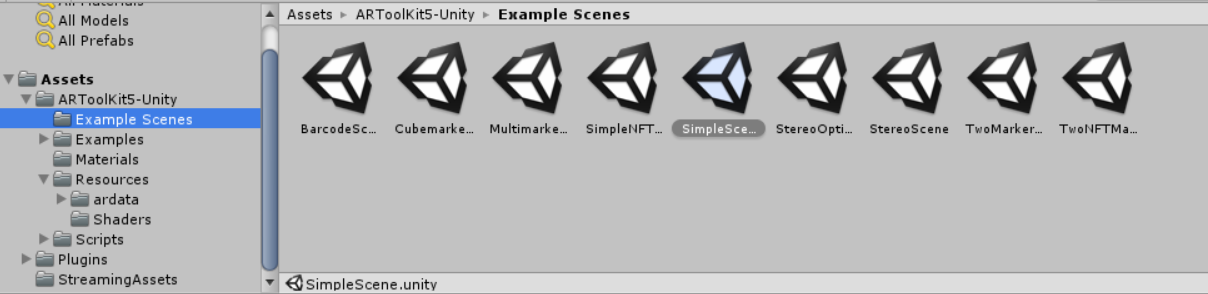




最后将生成的.mrk和.pat文件与数据集文件一起放到Assets\StreamingAssets文件夹下

# 五、如何使用四种不同的Marker type作为标记进行识别

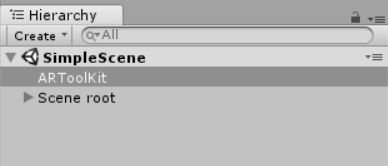
ARToolkit插件包中提供了使用各种类型marker作为目标识别对象的场景示例，位于example scenes下：



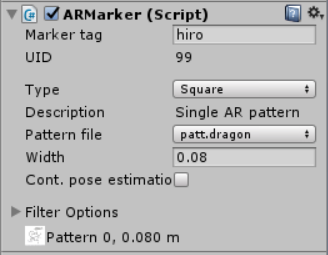
下面将针对不同类型的marker介绍典型的示例和用法：

## 1.使用传统模板标记

在SimpleScene示例中，使用了类型为”square”的marker用于对象的标记，通过在包含ARMarker脚本组件的物体上设置，在本例中即ARToolkit物体

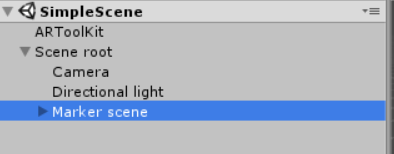


可以在Inspector属性面板下找到ARMarker组件：

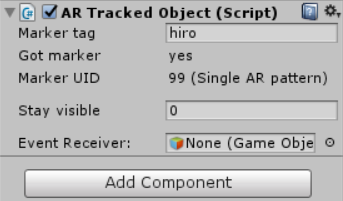


其中Marker tag由用户自定义，相当于给这个marker“取名”，在Type中选择该marker的类型，包含Square、Square Barcode、Multimarker、NFT，对于传统方形标记，选择Square，并在Pattern file中指定marker的图形（patt.dragon是之前自己制作好并放入Assets\ARToolKit5-Unity\Resources\ardata\markers文件夹下的图案）。

要检查Pattern file是否被ARToolkit接受并能在后续识别中正常使用，可以点击包含AR Tracked Object脚本组件的物体，在本例中为Marker scene



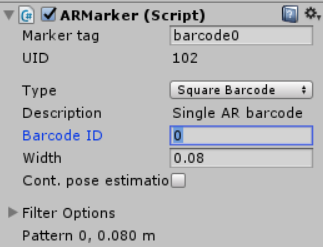
可在Inspector面板中查看AR Tracked Object组件的相关信息：



若在Got marker一栏显示“yes”，且下方出现具体的Marker UID值（99），则说明系统正确接受marker信息，并可以正常使用该marker。

## 2.使用二维条形码标记

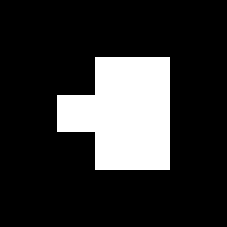
使用方法与使用“传统模板标记”类似，典型示例为BarcodeScene，不同的是，在Type中选择“Square Barcode” ，且通过Barcode ID来选择使用何种外观的barcode



可供选择的barcode有64个，位于ARUnity5-5.3.2-tools-win\doc\patterns\Matrix code 3x3 (72dpi)文件夹下：

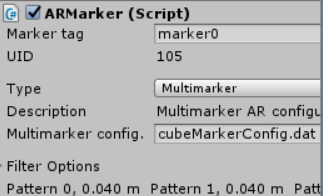


此处指定的Barcode ID为0，则表示选择的barcode为



## 3.使用多重标记

Multimarker的使用较其它三种标记的使用复杂一些，典型的示例为CubeMarkerScene，用法为在Type处选择“Multimarker”，随后在Multimarker config处添加cubeMarkerConfig.dat文件，该文件位于Assets\StreamingAssets下：



Multimarker的使用是基于Barcode的，打开cubeMarkerConfig.dat文件可以看到如下信息：

|  |
| --- |
| #the number of patterns to be recognized  6  #marker 1  00  40.0  1.0000 0.0000 0.0000 0.0000  0.0000 1.0000 0.0000 0.0000  0.0000 0.0000 1.0000 0.0000  #marker 2  01  40.0  1.0000 0.0000 0.0000 0.0000  0.0000 0.0000 1.0000 30.0000  0.0000 -1.0000 0.0000 -30.0000  #marker 3  02  40.0  0.0000 0.0000 1.0000 30.0000  0.0000 1.0000 0.0000 0.0000  -1.0000 0.0000 0.0000 -30.0000  #marker 4  03  40.0  1.0000 0.0000 0.0000 0.0000  0.0000 -1.0000 0.0000 0.0000  0.0000 0.0000 -1.0000 -60.0000  #marker 5  04  40.0  1.0000 0.0000 0.0000 0.0000  0.0000 0.0000 -1.0000 -30.0000  0.0000 1.0000 0.0000 -30.0000  #marker 6  05  40.0  0.0000 0.0000 -1.0000 -30.0000  0.0000 1.0000 0.0000 0.0000  1.0000 0.0000 0.0000 -30.0000 |

其中#开头的行代表注释

.dat文件的格式如下：

文件中的第一个非注释行必须是一个整数，表示该Multimarker使用的barcode数量

随后的每5个非注释行一起用来表示一个barcode的信息（包括位置和图案信息）：

其中第1行为大于或等于0的整数，为选择的barcode的序号。

第2行为打印成实际图案时，barcode的外边框长度，单位为mm

以上两行给出了barcode的图案信息。

下面的3-5行给出了标准4×4均匀坐标变换矩阵的前三行（旋转矩阵，行 - 主要顺序），该变换是从组合的多标记集（整体）的坐标系原点到该标记（成员）的原点。（注：坐标系的建立满足右手原则，一般为X水平向右，Y水平向上，Z垂直向外）

考虑marker1:

#marker 1

00

40.0

1.0000 0.0000 0.0000 0.0000

0.0000 1.0000 0.0000 0.0000

0.0000 0.0000 1.0000 0.0000

可以得出，marker1的图案为barcode 0，且打印出的标记外边框长度为40mm，且由3-5行的前三列可以得出，marker1相对于组合的多标记集的坐标系在各个轴方向上的旋转偏移量都为0，且由最后一列可以看出，marker1的坐标系原点在各个方向（X、Y、Z）上距离集坐标系原点的距离都为0，这就可以说明，marker1的坐标系原点就是集坐标系的原点，且marker1的坐标系各个轴的指向与集坐标系完全一致，概括之，marker1的坐标系就是集坐标系。

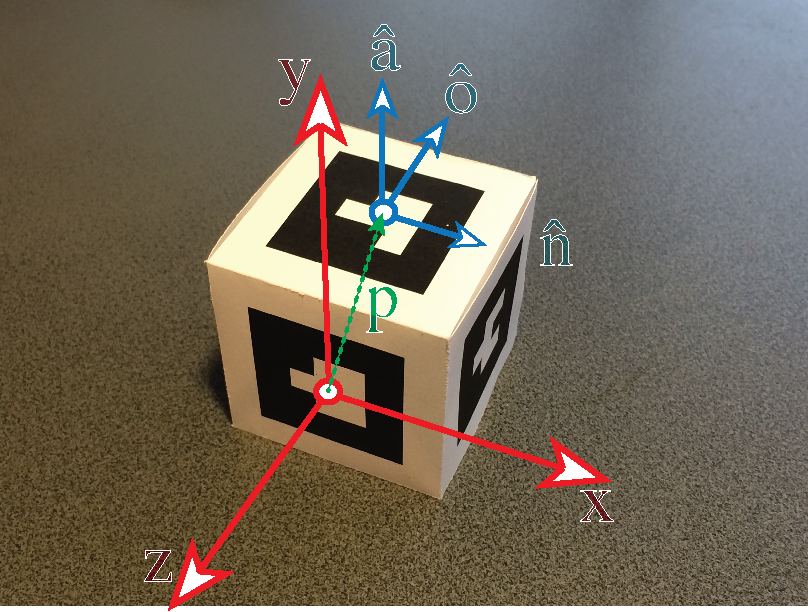
再考虑marker2:

第3-5行为：

1.0000 0.0000 0.0000 0.0000

0.0000 0.0000 1.0000 30.0000

0.0000 -1.0000 0.0000 -30.0000



上图中，对于marker2，图案为barcode 01，由其自身的转向可以得出其自身的坐标系为：a对应于Z，o对应于Y，n对应于X

再有旋转矩阵又可以表示为：

n [x] o [x] a [x] p [x]

n [y] o [y] a [y] p [y]

n [z] o [z] a [z] p [z]

在上图中，因为n与X有相同的指向，所以n[x]=1,n[y]=n[z]=0

o与-Z有相同的指向，所以o[x]=o[y]=0，o[z]=-1

a与Y有相同的指向，所以a[x]=a[z]=0，a[z]=1

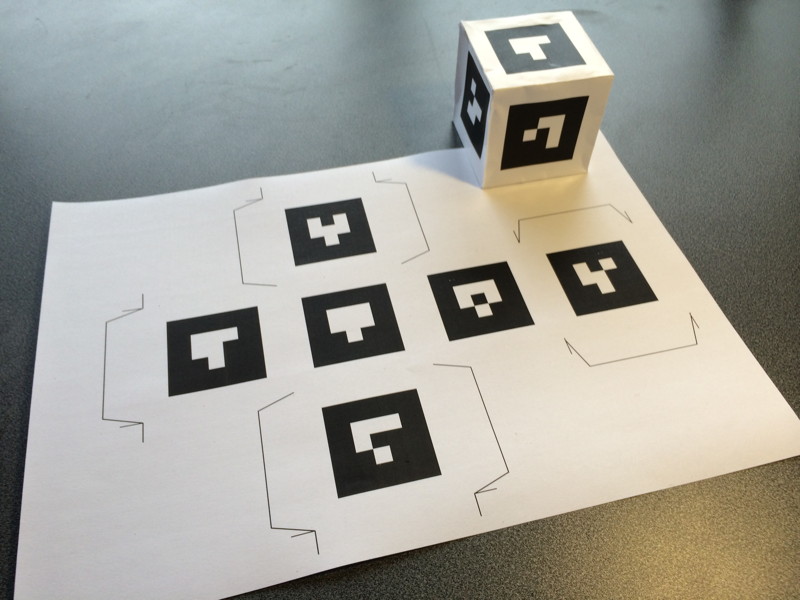
同时marker2原点距离集坐标系（marker1坐标系）原点在x方向上的距离为0，在y方向上的距离为+30mm（白色边框的宽度为10mm），在z方向上的距离为-30mm，所以p[x]=0，p[y]=30mm，p[z]=-30mm

这样一来就推导出了marker2的旋转矩阵信息，反过来，也可以根据.dat中的旋转矩阵信息，确定marker2与集坐标系之间的空间位置关系。

你可以按照同样的方法接着验证其他的marker。

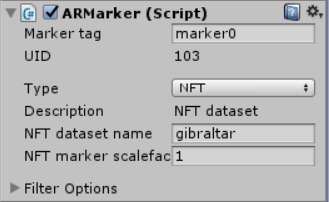
最后给一张CubeMarker的实物图：

（貌似在官网下的插件包里没有下面的打印图，实际验证时还是从网上其他地方找的，有点坑。。。）



## 4.使用自然特征图片标记（包括基于基准标记的自然特征图片）

使用方法与使用“传统模板标记”类似，典型示例为SimpleNFTScene，不同的是，在Type中选择“NFT”，并且输入该NFT图片的名字（位于\Assets\StreamingAssets下）：



# 六、使用ARToolkit制作实例的简易流程

1）创建一个Gameobject物体，命名为ARToolkit，将ARController和ARMarker脚本拖到其上（首先需要思考这个Scene需要几个Marker，需要几个就拖几个ARMarker到物体上），设置Marker tag（不同的Marker需要有不同的Marker tag值用于区分），然后选择Maker type(Square、Square Barcode、Multimarker、NFT)，最后选择具体的Marker图案。

2）在layers中定义一个“AR Foreground”的layer，这将是放置所有Marker对应虚拟物体的layer。

3）创建一个Gameobject物体，命名为scene root，将AROrigin脚本拖到上面，并把该物体位置放到0,0,0 ,AR场景中所有的动态内容都将作为这个基点的子物体（child）存在, 且它在你的场景中处于最高层级（在Unity中的hierarchy中不存在任何Parent），并把其layer设置为AR foreground。

4）将camera、Light移到scene root下，且将ARcamera拖到camera上，将culling mask 属性设置为AR foreground。

5）在scene root下创建一个Gameobject物体，命名为Marker scene，layer设置为AR foreground，将ARTrackedObject拖到其上，将其Marker tag属性与之前ARToolkit中的Marker tag相匹配，如果匹配成功，则Got Marker属性会显示“yes”，且会显示对应的Marker UID，最后将Marker scene的rotation X设置为90。

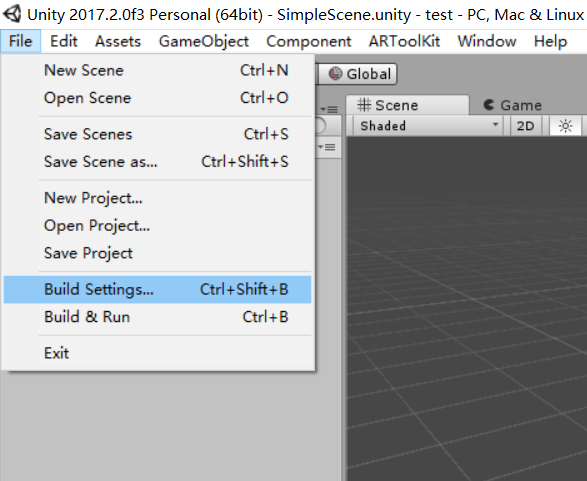
6）在Marker scene下创建需要显示的虚拟物体（如cube），scale 属性可以设置为{0.08, 0.08, 0.08}，position属性可以设置为{0, 0.04, 0}

7）创建完成，点击play验证。

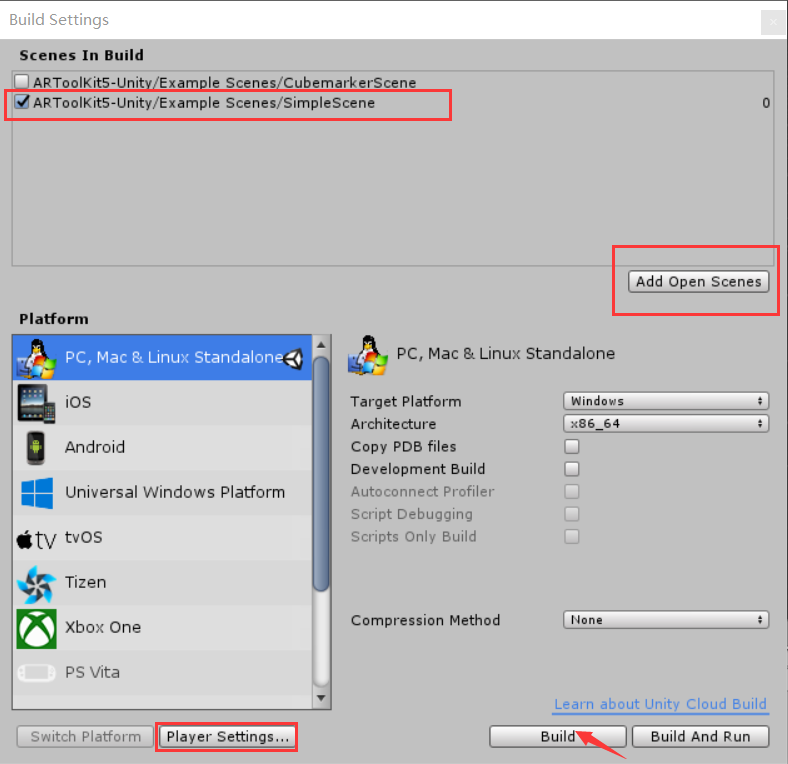
# 七、包含ARToolkit的unity场景发布（windows/android）步骤及一些错误处理

## 1.发布到windows平台

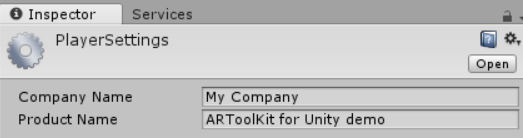
1）首先双击需要发布的场景，在File菜单栏下点击Build Settings



2）在Build Setting中选中PC，Mac & Linux Standalone，点击Add Open Scenes，选择要发布的场景。



3）点击左下方的Player Settings，在Inspector面板中输入相应的campany name和product name，最后在Build Setting中点击build



4）最后选择.exe文件的保存路径，单击保存即可完成发布

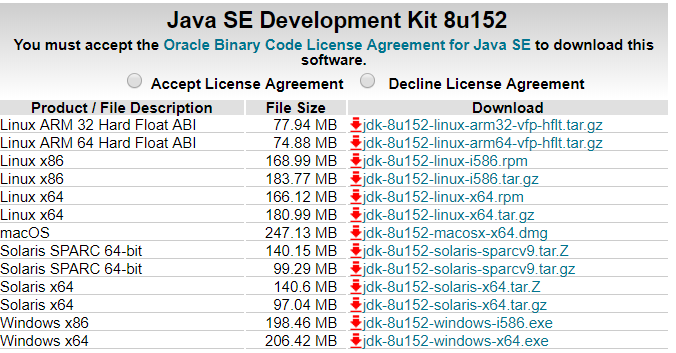
## 2.发布到android平台

发布到android平台需要jdk和android sdk的支持，首先安装jdk和android sdk

Jdk官网：

<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

建议下载jdk1.8.0\_152（开始用jdk1.8.0\_151发现会报错，换成52后就成功了）



配置环境变量可以百度

Android sdk下载：

官网：<https://developer.android.com/studio/index.html?hl=zh-cn>

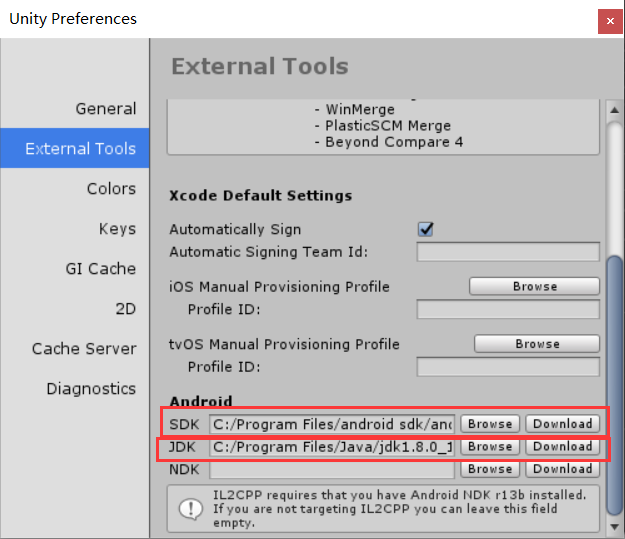
国内：<http://www.androiddevtools.cn/>

关于Android sdk的下载和配置可以参考这个帖子：

<http://blog.csdn.net/love4399/article/details/77164500>

唯一需要注意的是，在用SDK Manager下载API时，应该保证API level≥25

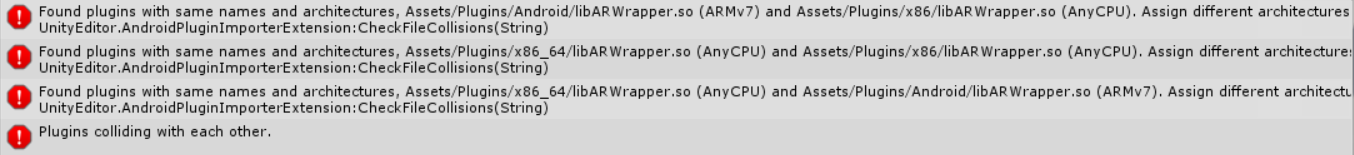
配置完jdk和android sdk后，在unity中点击Edit->Preferences，在External Tools中配置sdk和jdk的路径，选择两者所在的根目录即可



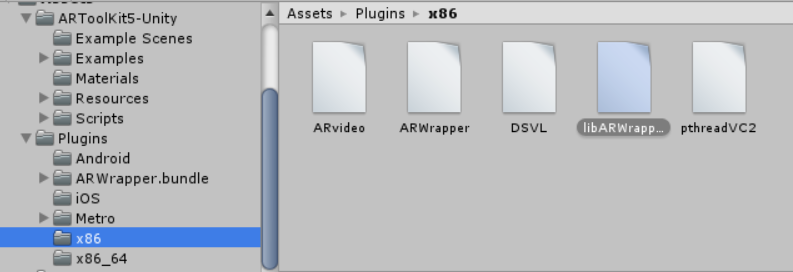
之后进入Build Settings，选择Android，在添加完需要发布的场景后，点击build即可。

错误处理：

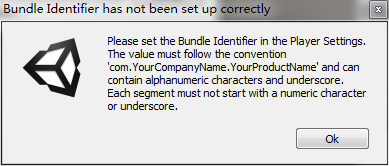
如果出现这样的错误：



则删除x86和x86\_64目录下的libARWrapper.so文件，再重新build即可



如果出现这样的错误：



可参考<http://blog.csdn.net/love4399/article/details/77164500>解决

以上即为我本人学习和使用ARToolkit for unity过程中的一些总结和经验，学习资料几乎都是基于ARToolkit官方文档，某些过程的介绍有所省略，若要了解详细情况，登陆ARToolkit官网查询即可：<https://www.artoolkit.org/documentation/>

By：李涛涛