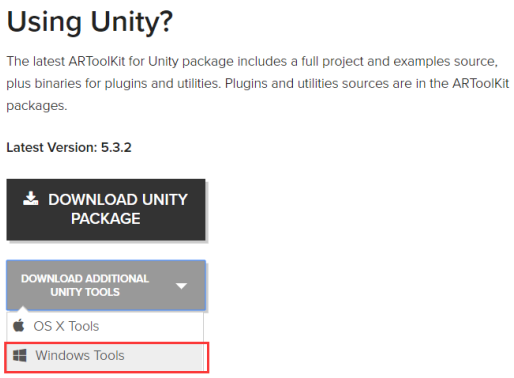
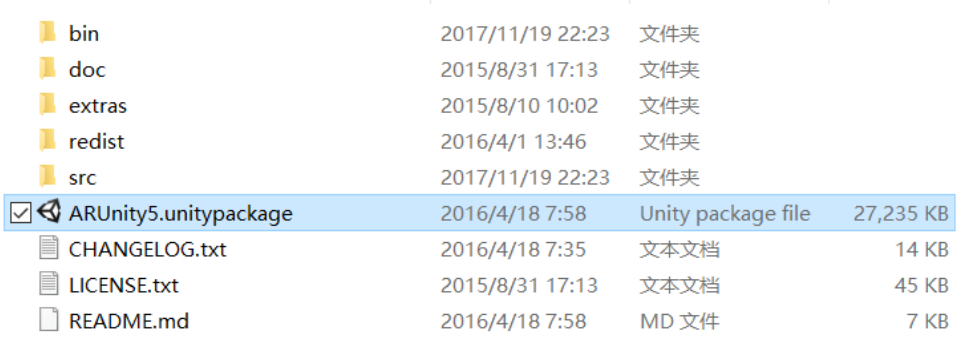
目录

一、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）

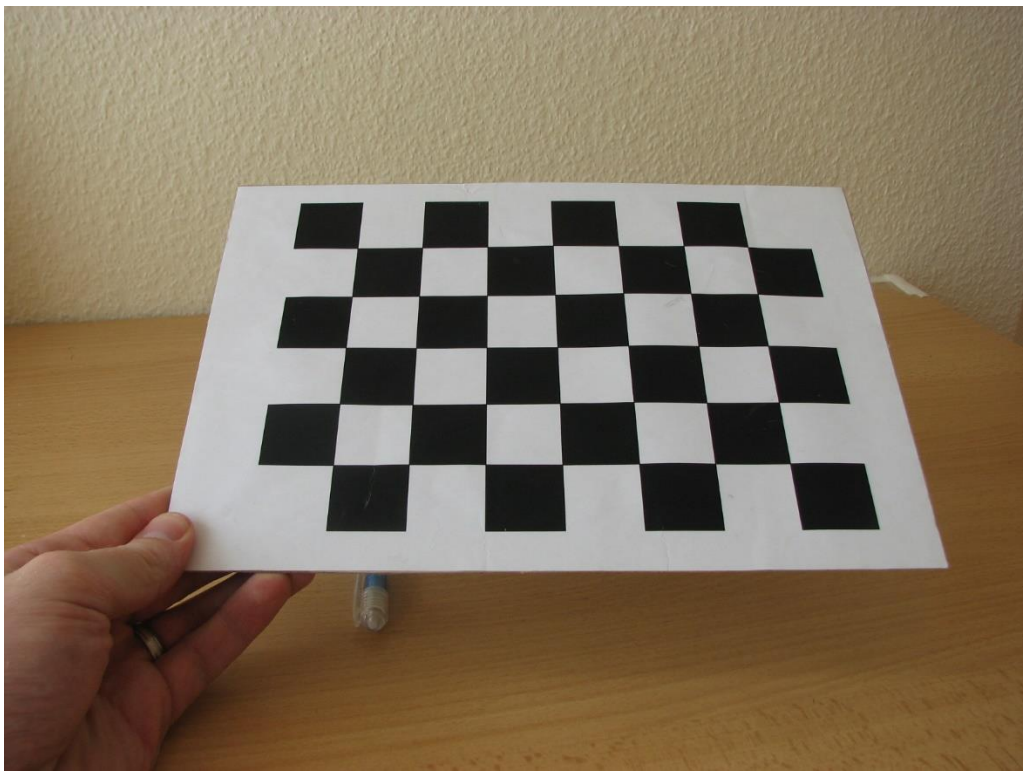


二、如何校准自己的相机（让 ARtoolkit 使用自己相机的参数）

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：

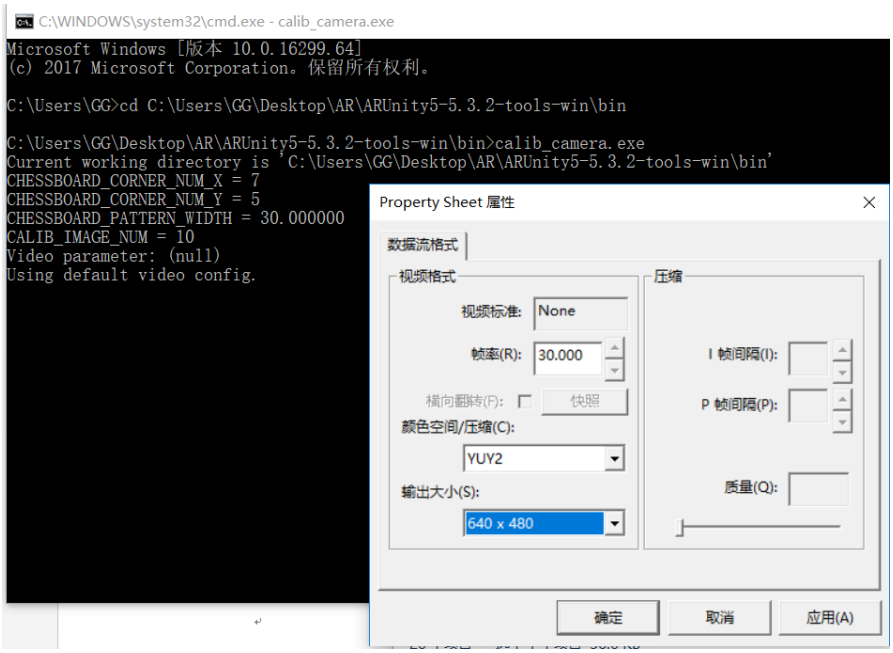


打印完成后，将该打印纸固定到一块薄板上（保持平坦），且测量打印图中一个正方形边长的大小（以毫米为单位）并记录（后面要用），如果在打印的时候没有进行缩放，那么打印出的图案中小正方形的边长刚好是 30mm，同时，查看在图纸的 X 方向（水平方向）有多少个中间角（即两个黑色正方形顶点的交界处），上图中，该值为 7，同理，上图中图纸的 Y 方向（竖直方向）有 5 个中间角。

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。

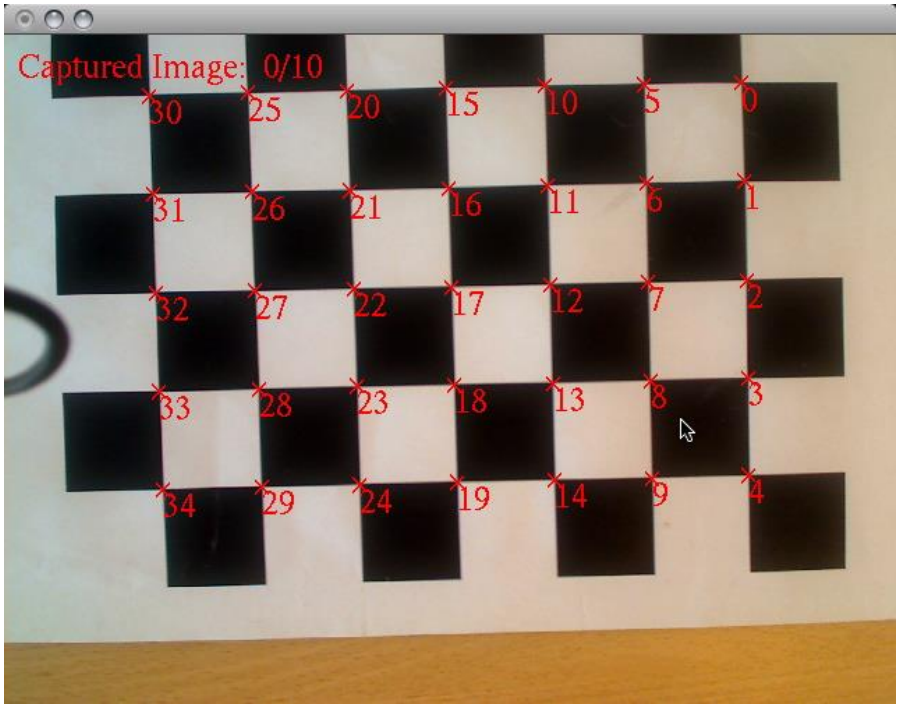
开始校准：
在 windows 命令行中输入 calib_camera.exe，会产生如下输出：



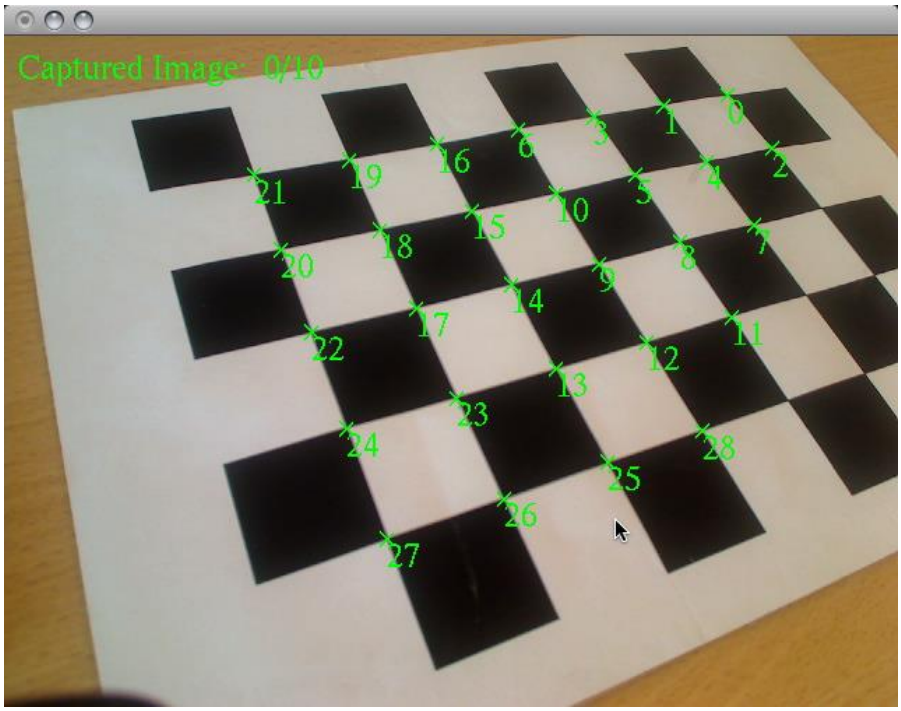
此处可以为相机选择不同的宽高比，建议选择 640×480，且在之后的图片训练和标记追踪时都使用该值。

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：

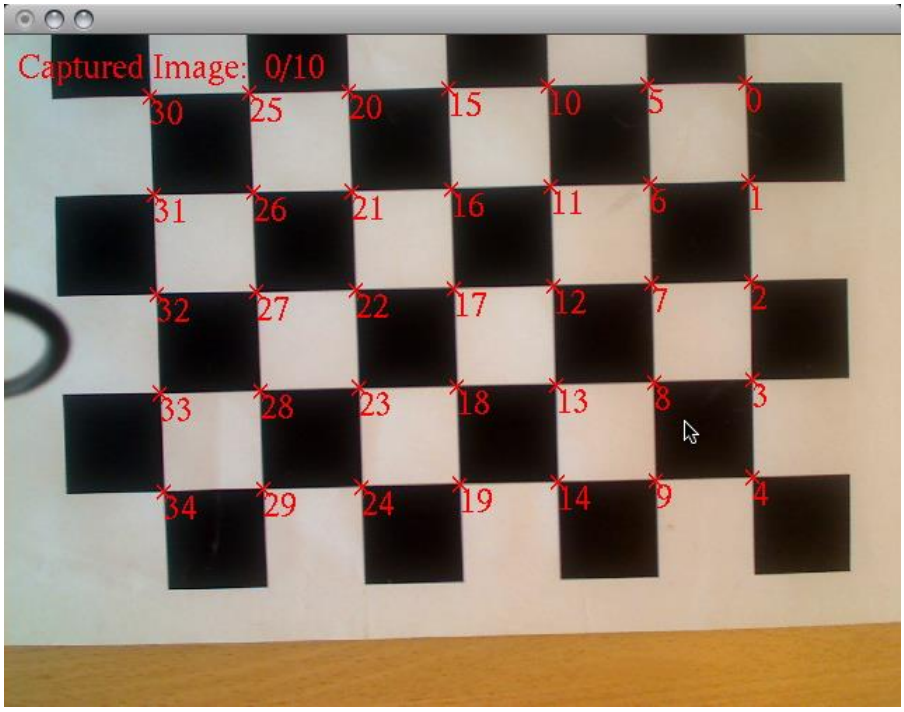


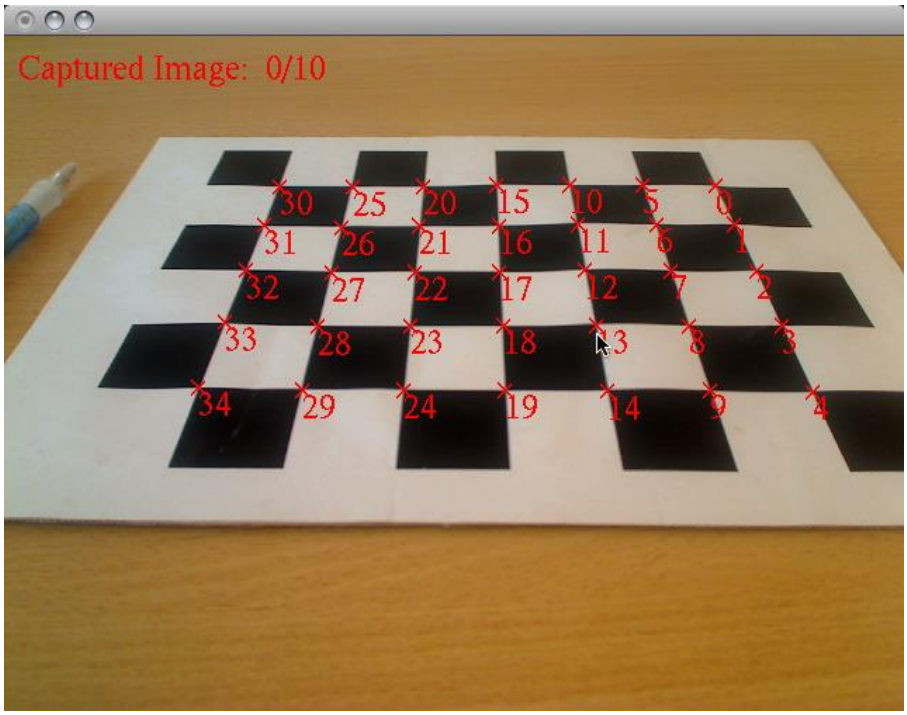
如果某些角落被相机边缘遮挡，或者照明或反射不良，则“X”将变为绿色：

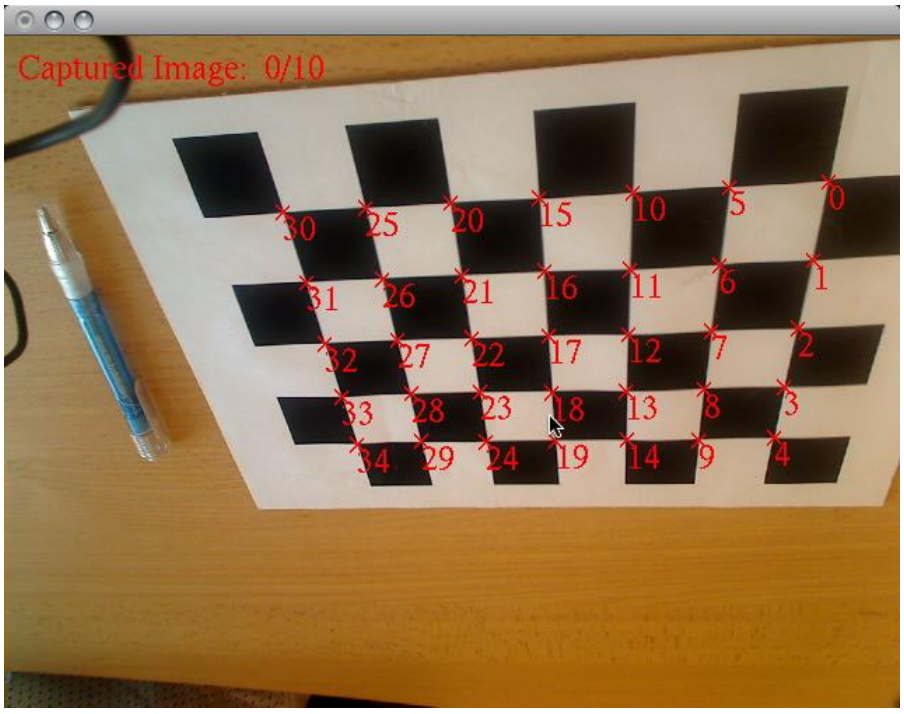


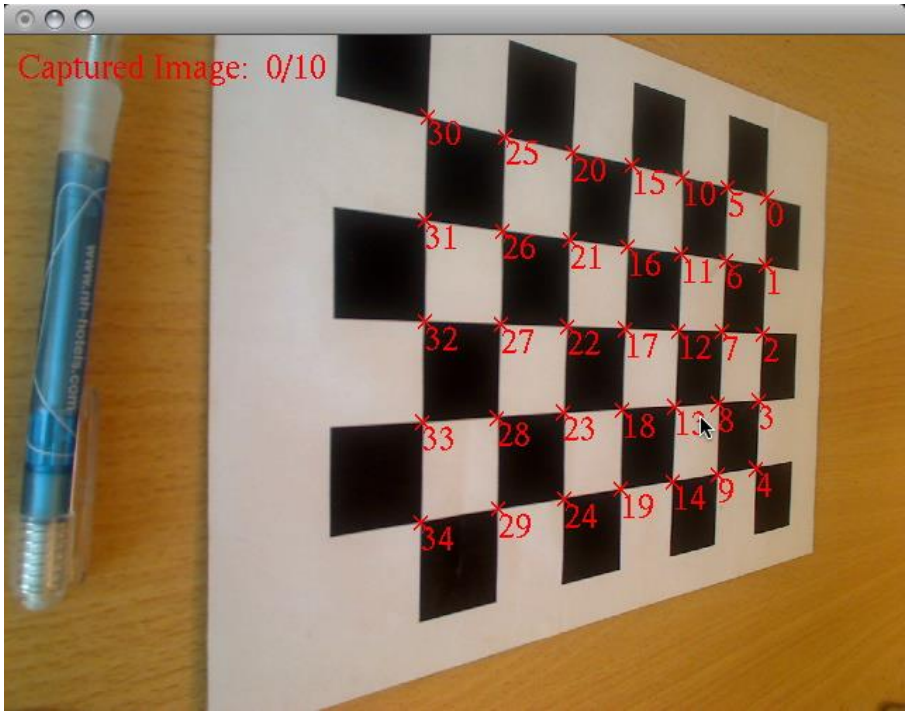
一旦有一个全红的图像，就可以按下空格键完成一次捕获。

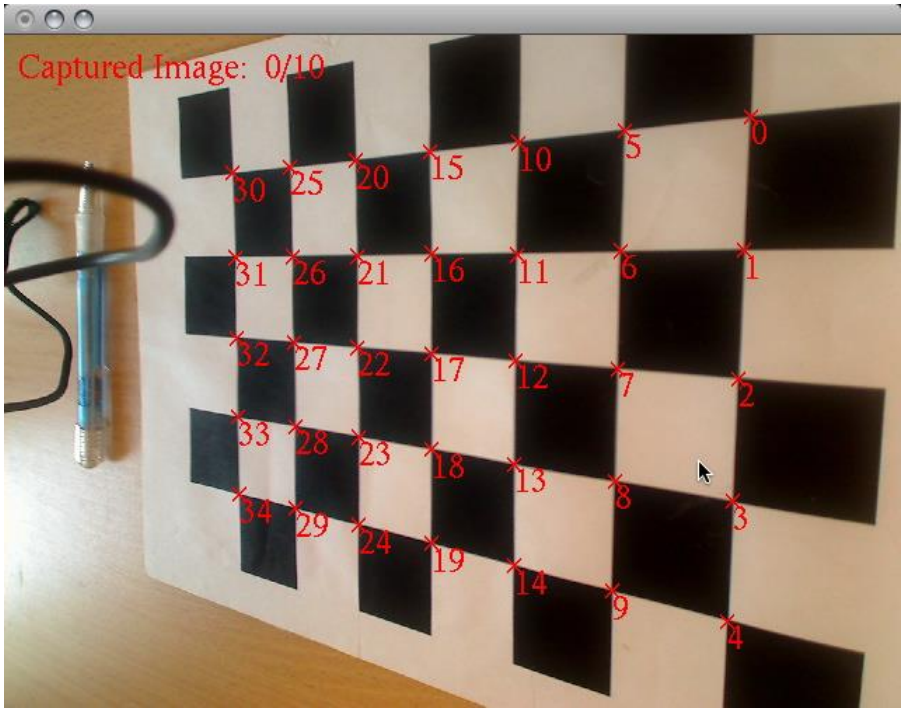
为了更好地校准相机，最好以相机镜头的各种角度获得校准的图像，如下示例角度，包括颠倒：

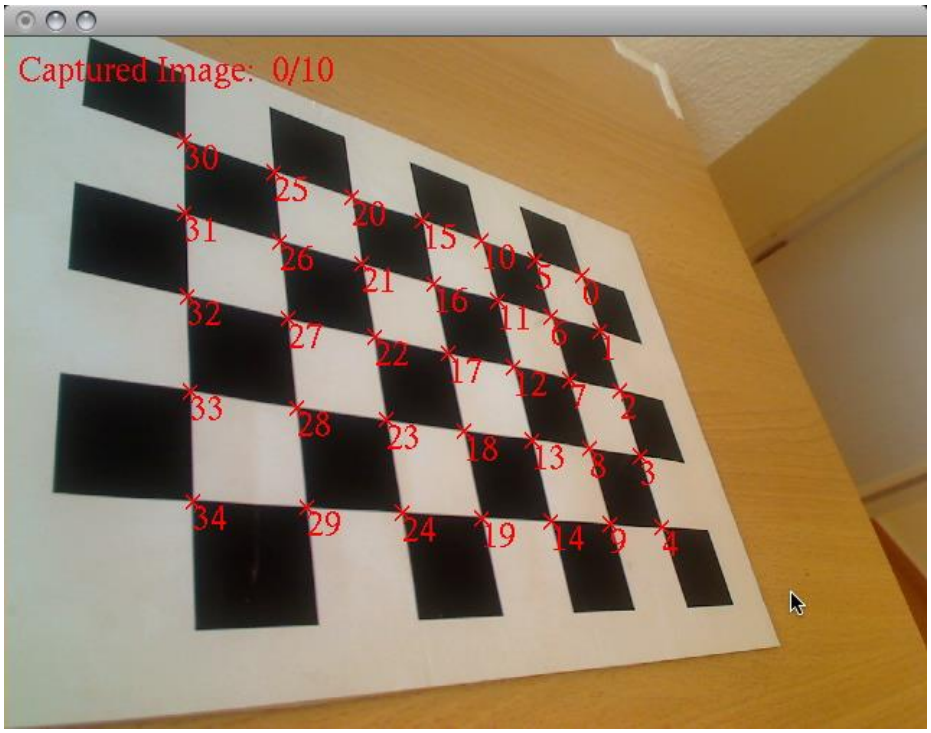



Captured Image: 0/10
30 25 20 15 10 5 0
31 26 21 16 11 6 1
32 27 22 17 12 7 2
33 28 23 18 13 8 3
34 29 24 19 14 9 4


Captured Image: 0/10
30 25 20 15 10 5 0
31 26 21 16 11 6 1
32 27 22 17 12 7 2
33 28 23 18 13 8 3
34 29 24 19 14 9 4


Captured Image: 0/10
30 25 20 15 10 5 0
31 26 21 16 11 6 1
32 27 22 17 12 7 2
33 28 23 18 13 8 3
34 29 24 19 14 9 4


Captured Image: 0/10
30 25 20 15 10 5 0
31 26 21 16 11 6 1
32 27 22 17 12 7 2
33 28 23 18 13 8 3
34 29 24 19 14 9 4



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：

```
Distortion factor: k1=-0.064153, k2=0.000674,
                   p1=-0.003693, p2=-0.011219,
                   fx=6819.694824, fy=4906.716797,
                   x0=-13355.703125, y0=1494.664429,
                   s=-0.891334
-7651.11364 -0.00000 -13355.70312 0.00000
-0.00000 -5504.91609 1494.66443 0.00000
0.00000 0.00000 1.00000 0.00000
------------------------------------
Err[ 1]: 0.384557[pixel]
Err[ 2]: 0.637637[pixel]
Err[ 3]: 0.605374[pixel]
Err[ 4]: 0.517292[pixel]
Err[ 5]: 0.730103[pixel]
Err[ 6]: 0.560028[pixel]
Err[ 7]: 0.586818[pixel]
Err[ 8]: 0.501263[pixel]
Err[ 9]: 0.437201[pixel]
Err[10]: 0.385920[pixel]
Filename[camera_para.dat]:
```

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。

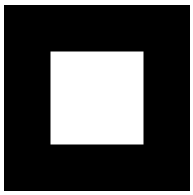
三、标记制作

1.制作传统模板方形标记

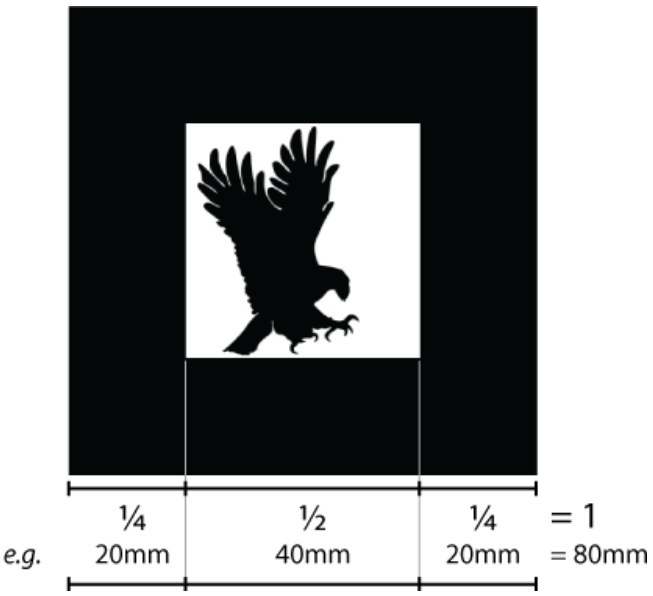
上图是传统模板方形标记的示例，想要制作出符合要求的标记，需要遵循一些规则：

- 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

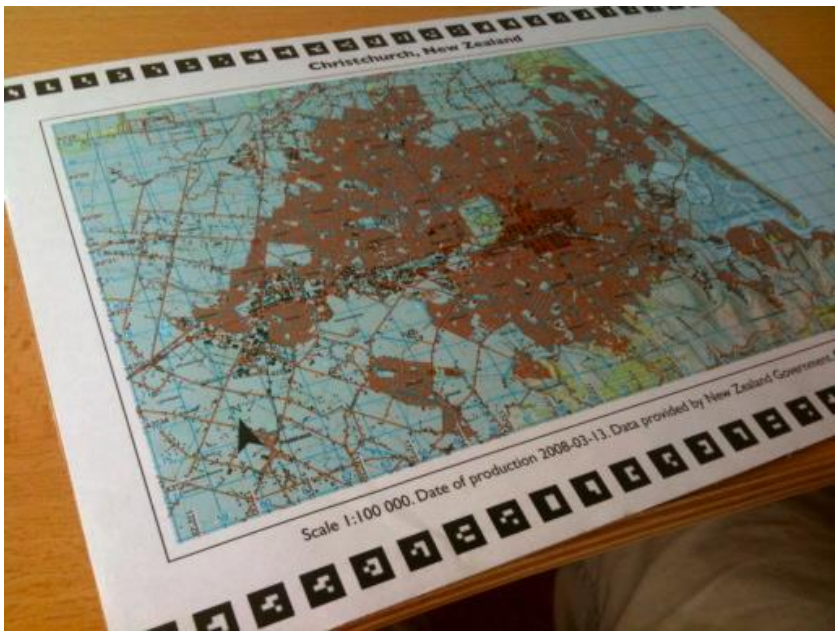
制作基于基准标记的自然特征图片与 NFT 类似，目的是为了在自然图片内部或外部周围通过嵌入标记来增强图像的识别和追踪。

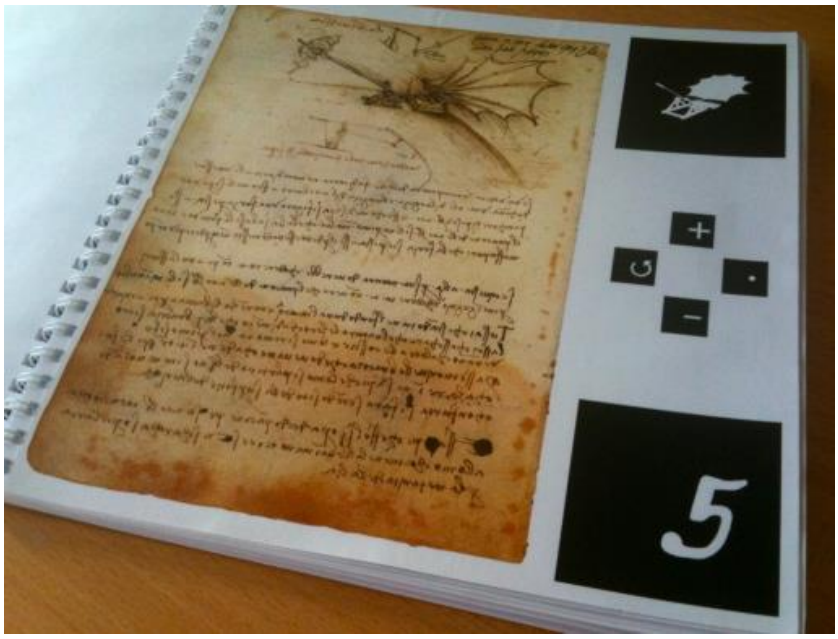
使用基于基准标记的自然特征图片追踪具有如下优点：
• 使用 NFT 1.0 跟踪器加基准标记比 NFT 1.0 + 2.0 完全无标记跟踪在计算上花费更少。
• NFT 2.0 追踪器对于任何时候都可以区分的不同标记的数量都有实际的限制。因此，如果需要跟踪大量图像，则基准标记能够有效识别意图被跟踪的多个图像。
• 基准标记跟踪增加了对跟踪的健壮性，特别是在光线不足的情况下，或者当相机远离跟踪的图像时。

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：



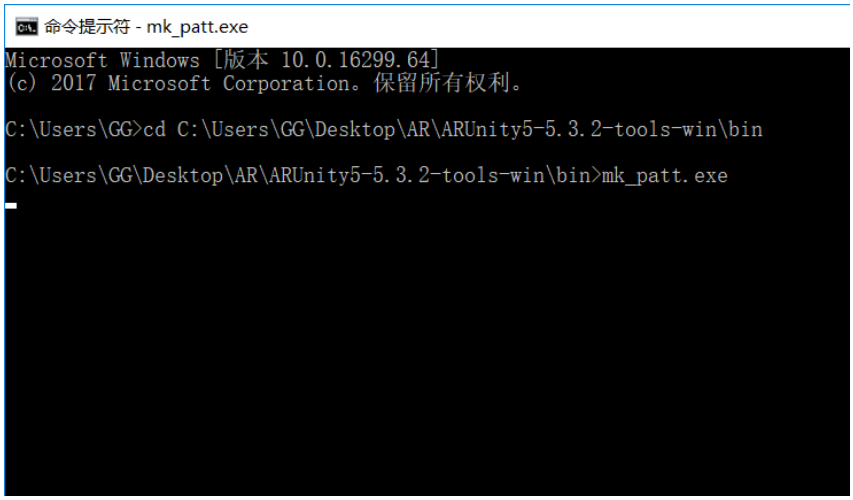


同时，如果标记内部的内容为所嵌入背景图像的一部分，那么该标记在增强识别的同时，在图像中看起来也不会那么突兀：

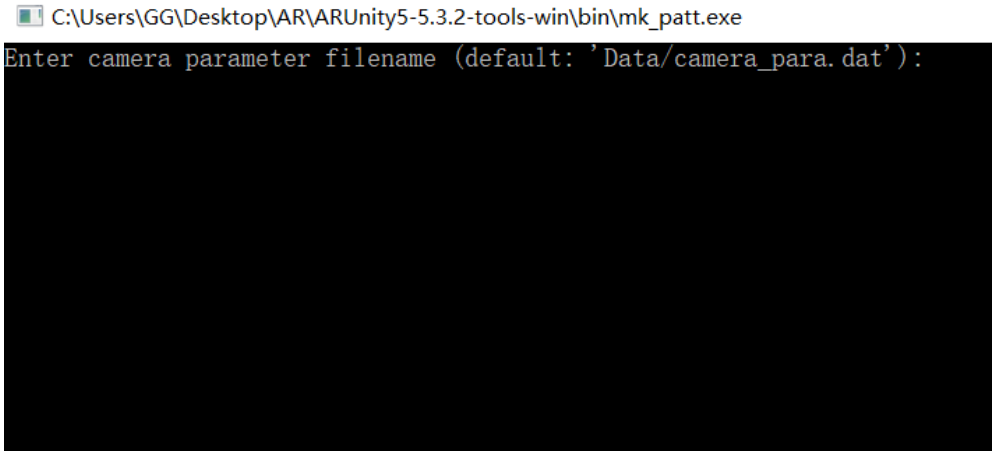
四、标记训练（Marker Training）

1. 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：



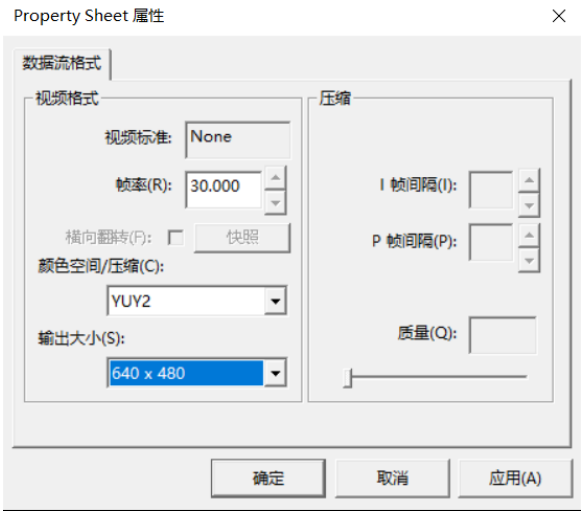
此时会自动产生另一个终端，提示输入相机校准参数（以文件的格式输入）：



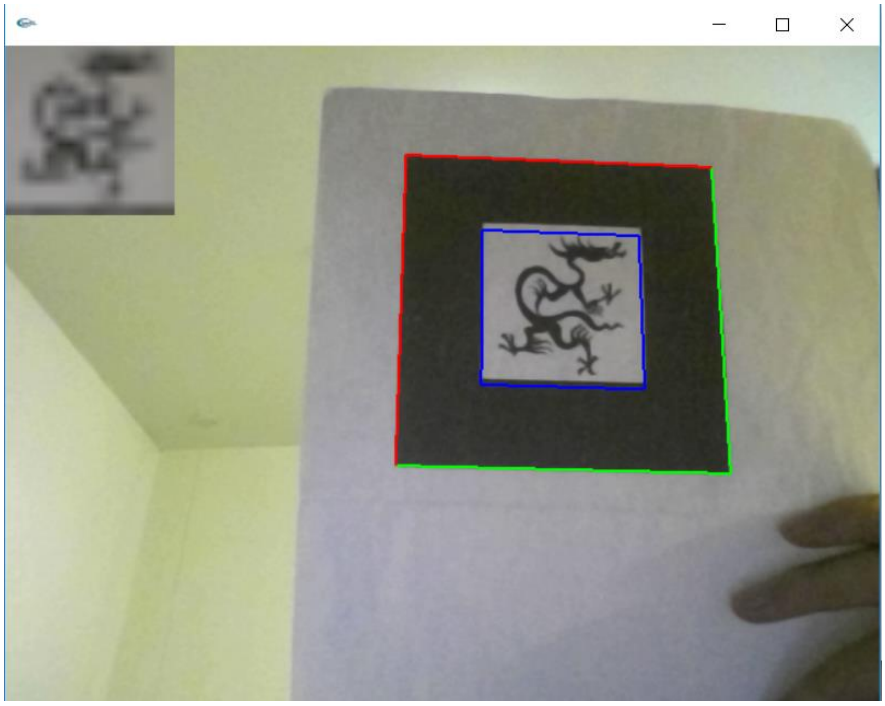
回车即可，程序默认会读取位于 ARUnity5-5.3.2-tools-win\bin\Data 文件夹下的相机校准文件

camera_para.dat

随后选择摄像头的相关配置信息：

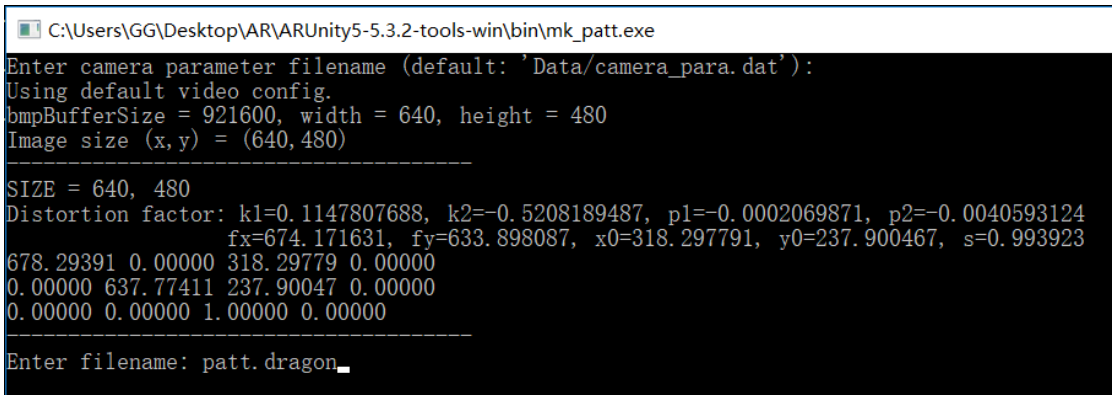


随后，摄像头被激活：

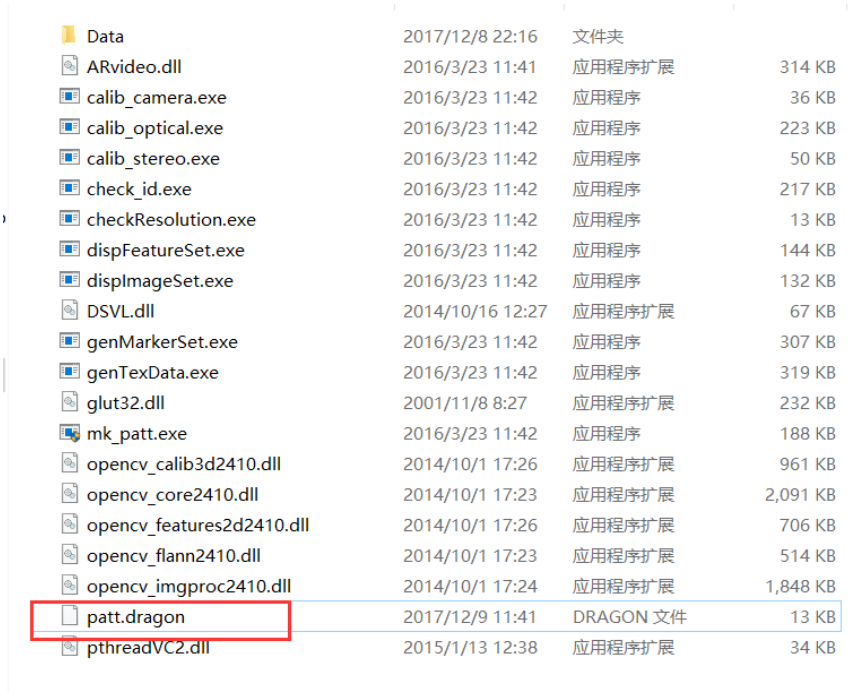


把相机对准 marker，如果 ARToolKit 已经识别标记，它会用红线和绿线勾勒出来。旋转 marker，让红线角落标记在 marker 的左上角，绿线角落标记在 marker 的右下角，随后点击鼠标左键完成图像捕获。

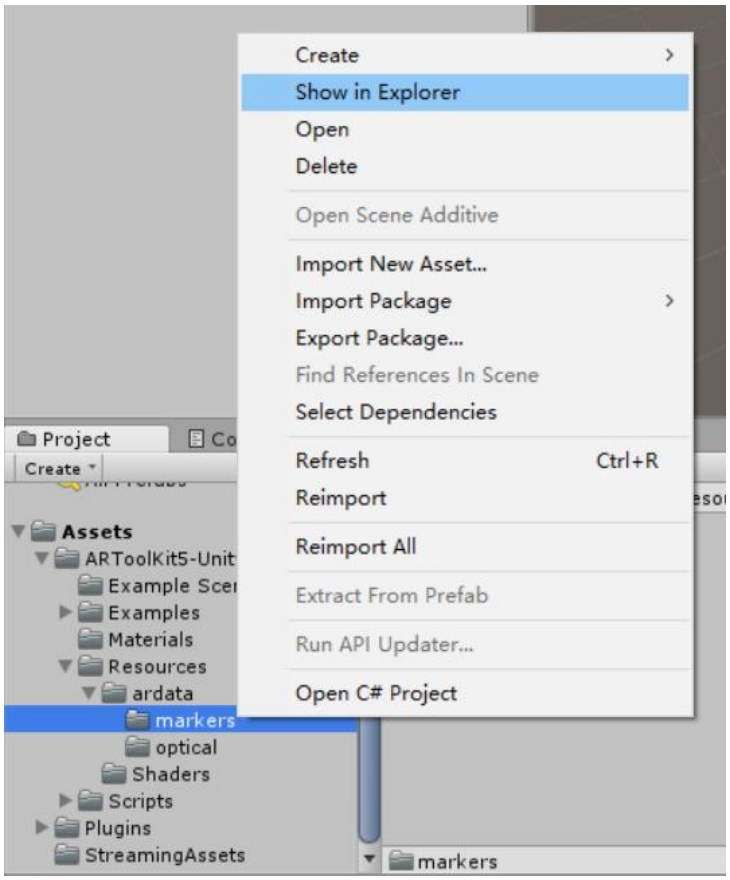
随后，终端提示要求输入文件名：



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：



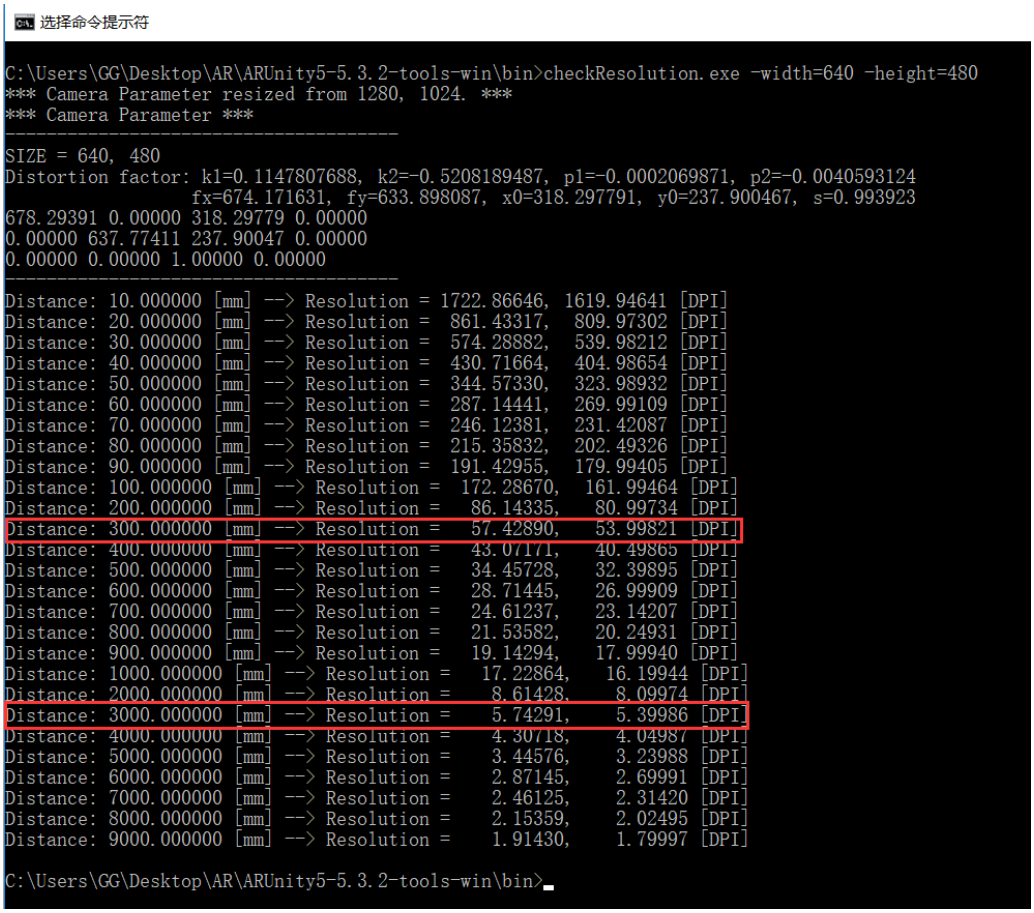
最后，将该文件更名为 patt.dragon.txt(更改成 txt 文件格式)，随后打开 unity 工程，定位到 Assets\ARToolKit5-Unity\Resources\ardata\markers 文件夹下，将 patt.dragon.txt 放到该文件夹下即可。



2. 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：

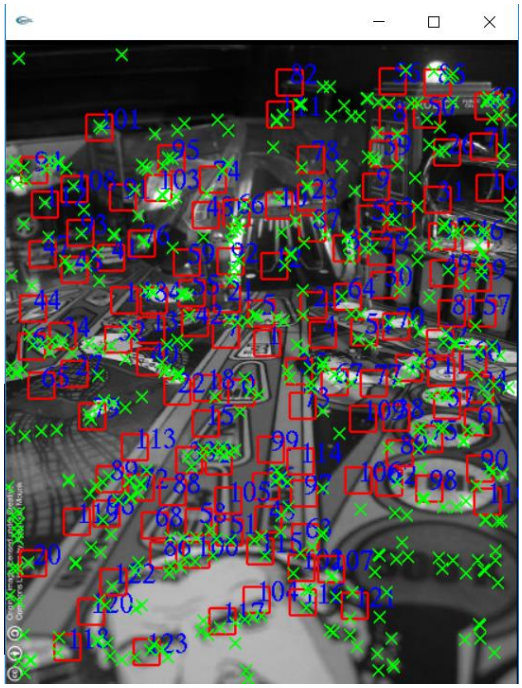
```
C:\Users\GG>cd C:\Users\GG\Desktop\AR\ARUnity5-5.3.2-tools-win\bin

C:\Users\GG\Desktop\AR\ARUnity5-5.3.2-tools-win\bin>genTexData.exe pinball.jpg
Select extraction level for tracking features, 0(few) <--> 4(many), [default=2]: 4
MAX_THRESH  = 0.980000
MIN_THRESH  = 0.450000
SD_THRESH   = 6.000000
Select extraction level for initializing features, 0(few) <--> 3(many), [default=1]: 3
SURF_FEATURE = 200
Reading JPEG file...
   Done.
JPEG image 'pinball.jpg' is 1637x2048.
JPEG image 'pinball.jpg' does not contain embedded resolution data, and no resolution specified on command-line.
Enter resolution to use (in decimal DPI):
```

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）：

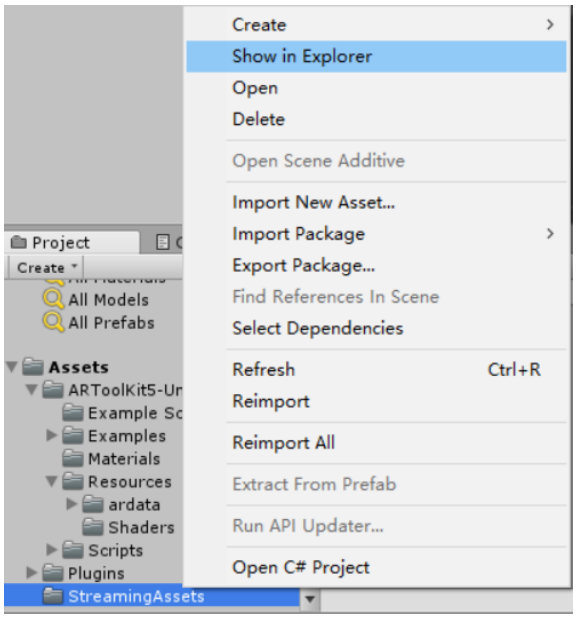
```
命令提示符
SD_THRESH   = 6.000000
Select extraction level for initializing features, 0(few) <--> 3(many), [default=1]: 3
SURF_FEATURE = 200
Reading JPEG file...
   Done.
JPEG image 'pinball.jpg' is 1637x2048.
JPEG image 'pinball.jpg' does not contain embedded resolution data, and no resolution specified on command-line.
Enter resolution to use (in decimal DPI): 220
Enter the minimum image resolution (DPI, in range [3.762, 220.000]): 5
Enter the maximum image resolution (DPI, in range [5.000, 220.000]): 58
Image DPI (1): 5.000000
Image DPI (2): 6.299605
Image DPI (3): 7.937006
Image DPI (4): 10.000001
Image DPI (5): 12.599212
Image DPI (6): 15.874012
Image DPI (7): 20.000002
Image DPI (8): 25.198423
Image DPI (9): 31.748024
Image DPI (10): 40.000004
Image DPI (11): 50.396847
Image DPI (12): 58.000000
Generating ImageSet...
   (Source image xsize=1637, ysize=2048, channels=3, dpi=220.0).
  Done.
Saving to pinball.iset...
  Done.
Generating FeatureList...
Start for 58.000000 dpi image.
        ImageSize =  233280[pixel]
```

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：

```
C:\Users\GG\Desktop\AR\ARUnity5-5.3.2-tools-win\bin>dispFeatureSet.exe pinball.fset
Read ImageSet.
Imageset contains 12 images.
  end.
Read FeatureSet.
  end.
Read FeatureSet3.
  end.
num = 3375
Size = (432,540) Zoom = 1.000000
58.000000[dpi] image. Size = (432,540)
fset:  Num of feature points: 124
fset3: Num of feature points: 610
```



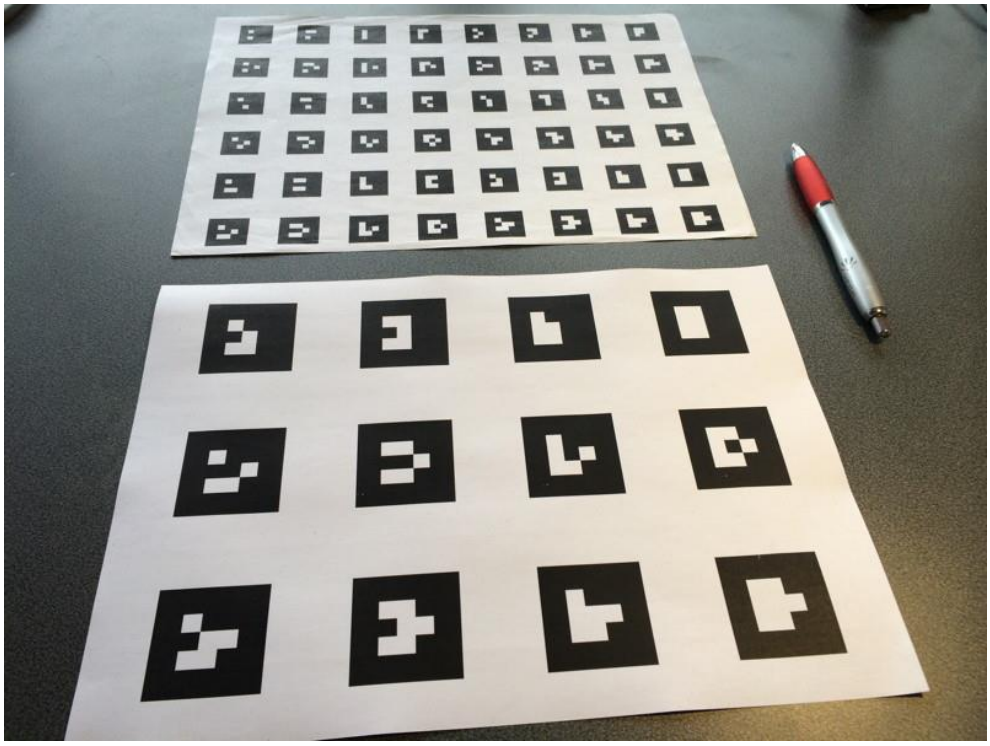
最后，将得到的 3 个数据集文件放到 Assets\StreamingAssets 文件夹下：



cubeMarkerConfig.dat	2015/8/7 12:54	DAT 文件	1 KB
cubeMarkerConfig.dat.meta	2015/8/7 12:54	META 文件	1 KB
gibraltar.fset	2015/8/31 17:13	FSET 文件	6 KB
gibraltar.fset.meta	2015/8/31 17:13	META 文件	1 KB
gibraltar.fset3	2015/8/31 17:13	FSET3 文件	354 KB
gibraltar.fset3.meta	2015/8/31 17:13	META 文件	1 KB
gibraltar.iset	2015/8/10 10:09	ISET 文件	62 KB
gibraltar.iset.meta	2015/8/31 17:13	META 文件	1 KB
Harlech_Castle_plan_colour.fset	2015/8/31 17:13	FSET 文件	21 KB
Harlech_Castle_plan_colour.fset.m...	2015/8/31 17:13	META 文件	1 KB
Harlech_Castle_plan_colour.fset3	2015/8/31 17:13	FSET3 文件	731 KB
Harlech_Castle_plan_colour.fset3....	2015/8/31 17:13	META 文件	1 KB
Harlech_Castle_plan_colour.iset	2015/8/10 10:10	ISET 文件	566 KB
Harlech_Castle_plan_colour.iset.m...	2015/8/31 17:13	META 文件	1 KB
multi-barcode-4x3.dat	2015/8/7 12:54	DAT 文件	1 KB
multi-barcode-4x3.dat.meta	2015/8/7 12:54	META 文件	1 KB
multi-barcode-8x6.dat	2015/8/7 12:54	DAT 文件	4 KB
multi-barcode-8x6.dat.meta	2015/8/7 12:54	META 文件	1 KB
pinball.fset	2017/12/9 15:03	FSET 文件	8 KB
pinball.fset3	2017/12/9 15:03	FSET3 文件	436 KB
pinball.iset	2017/12/9 15:03	ISET 文件	41 KB

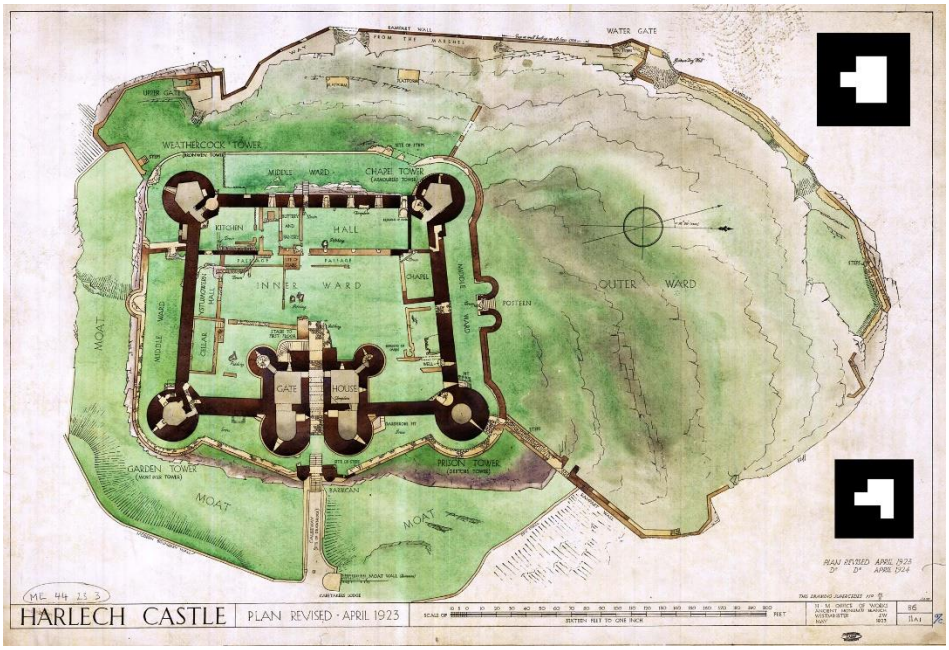
3. 多重标记训练（Multimarker）

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：



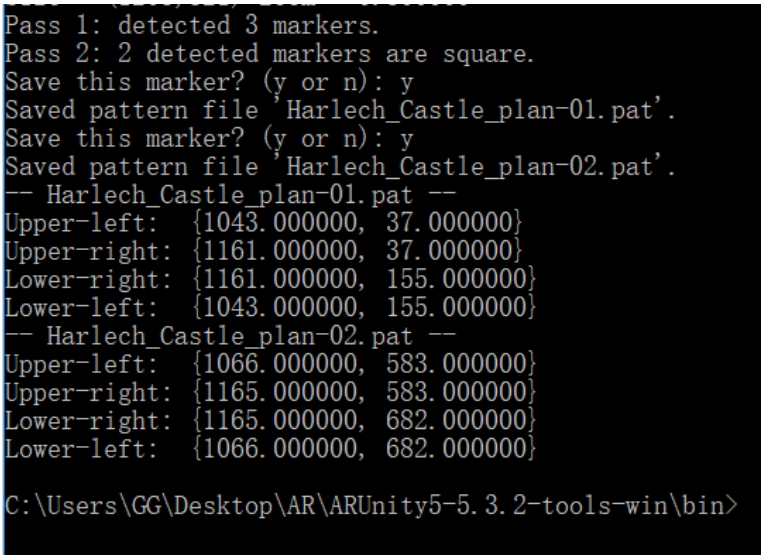
每个 barcode 具有对应的序号，这是后续使用 barcode 和 multimarker 的基础。

4. 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：

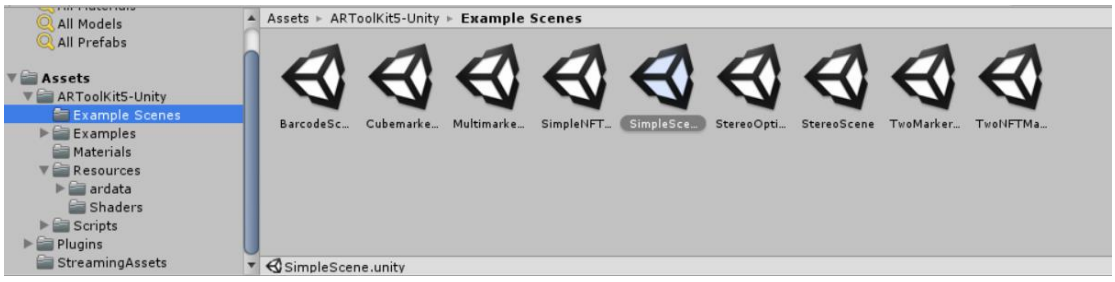


Harlech_Castle_plan.fset	2017/12/9 15:56	FSET 文件	24 KB
Harlech_Castle_plan.fset3	2017/12/9 15:56	FSET3 文件	661 KB
Harlech_Castle_plan.iset	2017/12/9 15:54	ISET 文件	231 KB
Harlech_Castle_plan.jpg	2017/11/20 11:37	JPG 文件	1,428 KB
Harlech_Castle_plan.mrk	2017/12/9 16:13	MRK 文件	1 KB
Harlech_Castle_plan-01.pat	2017/12/9 16:13	PAT 文件	13 KB
Harlech_Castle_plan-02.pat	2017/12/9 16:13	PAT 文件	13 KB

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

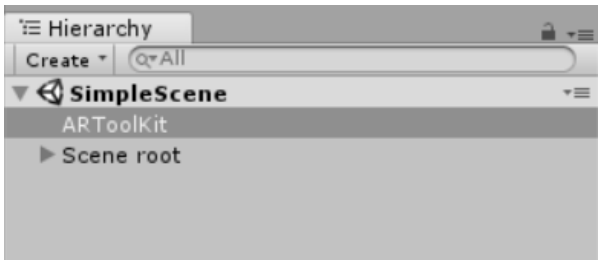
ARToolkit 插件包中提供了使用各种类型 marker 作为目标识别对象的场景示例，位于 example scenes 下：



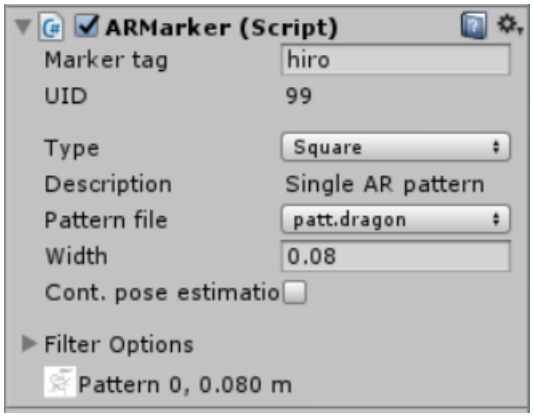
下面将针对不同类型的 marker 介绍典型的示例和用法：

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

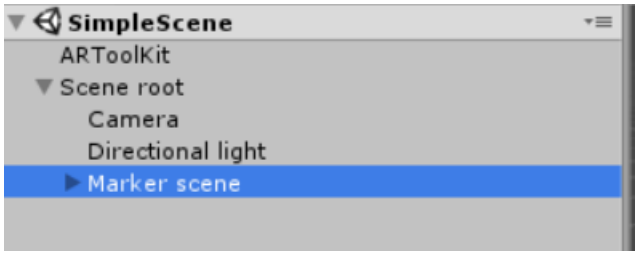


可以在 Inspector 属性面板下找到 ARMarker 组件：

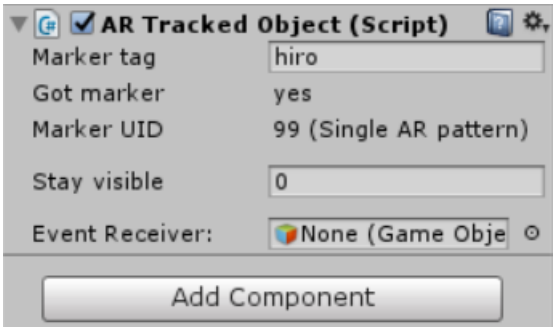


其中 Marker tag 由用户自定义，相当于给这个 marker “取名”，在 Type 中选择该 marker 的类型，包含 Square、Square Barcode、Multimarker、NFT，对于传统方形标记，选择 Square，并在 Pattern file 中指定 marker 的图形（patt.dragon 是之前自己制作好并放入 Assets\ARToolKit5-Unity\Resources\ardata\markers 文件夹下的图案）。

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



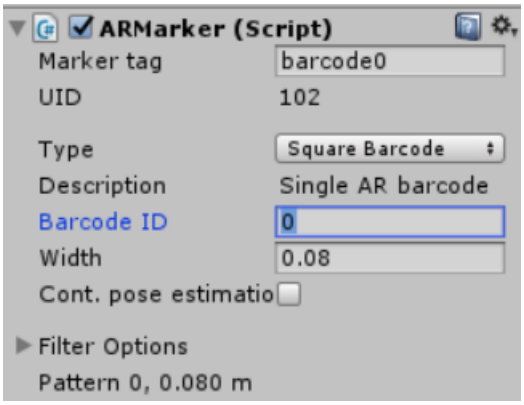
可在 Inspector 面板中查看 AR Tracked Object 组件的相关信息：



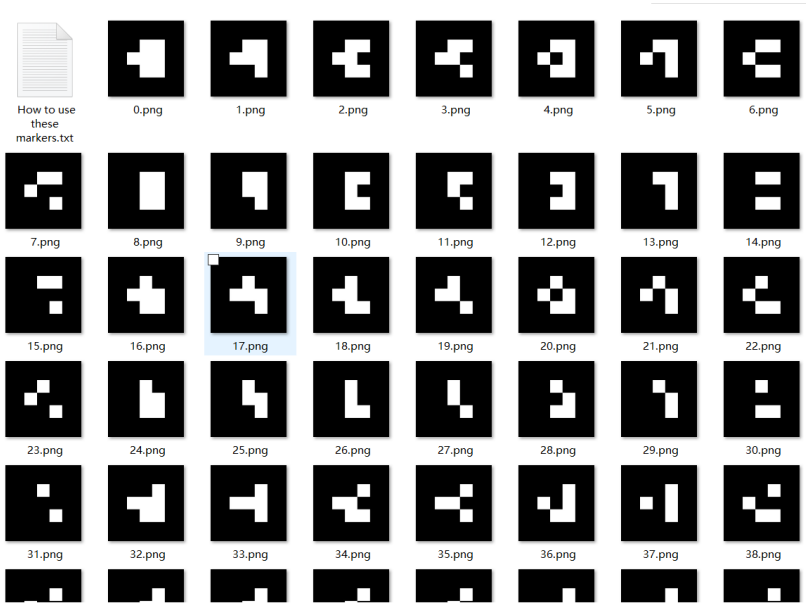
若在 Got marker 一栏显示“yes”，且下方出现具体的 Marker UID 值（99），则说明系统正确接受 marker 信息，并可以正常使用该 marker。

2.使用二维条形码标记

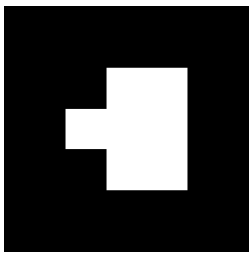
使用方法与使用“传统模板标记”类似，典型示例为 BarcodeScene，不同的是，在 Type 中选择“Square Barcode” ，且通过 Barcode ID 来选择使用何种外观的 barcode



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：

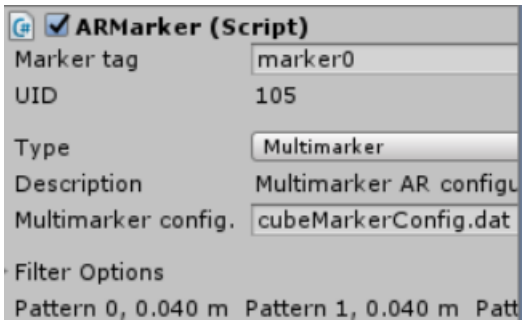


此处指定的 Barcode ID 为 0，则表示选择的 barcode 为



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。

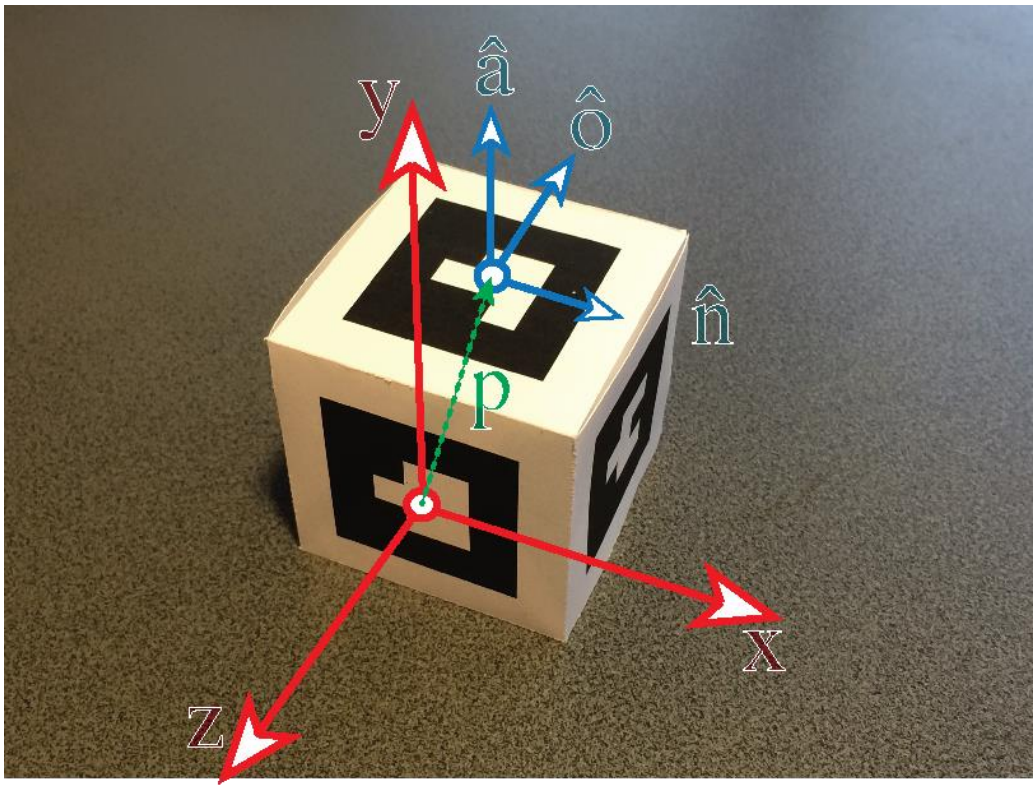
下面的 3-5 行给出了标准 4×4 均匀坐标变换矩阵的前三行（旋转矩阵，行 - 主要顺序），该变换是从组合的多标记集（整体）的坐标系原点到该标记（成员）的原点。（注：坐标系的建立满足右手原则，一般为 X 水平向右，Y 水平向上，Z 垂直向外）

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

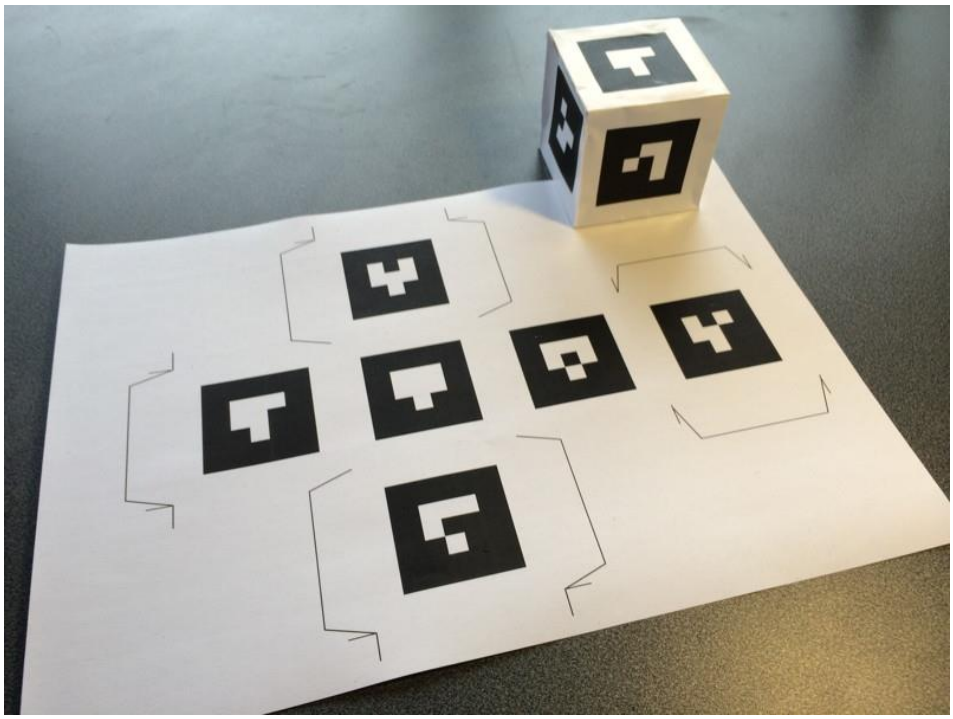


上图中，对于 marker2，图案为 barcode 01，由其自身的转向可以得出其自身的坐标系为：a 对应于 Z，o 对应于 Y，n 对应于 X
再有旋转矩阵又可以表示为：

```
n [x] o [x] a [x] p [x]
n [y] o [y] a [y] p [y]
n [z] o [z] a [z] p [z]
```

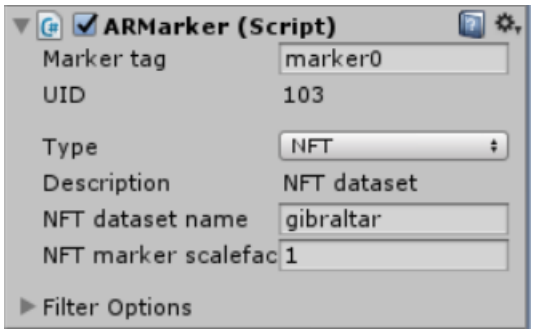
在上图中，因为 n 与 X 有相同的指向，所以 n[x]=1,n[y]=n[z]=0
o 与-Z 有相同的指向，所以 o[x]=o[y]=0，o[z]=-1
a 与 Y 有相同的指向，所以 a[x]=a[z]=0，a[z]=1
同时 marker2 原点距离集坐标系（marker1 坐标系）原点在 x 方向上的距离为 0，在 y 方向上的距离为+30mm（白色边框的宽度为 10mm），在 z 方向上的距离为-

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。

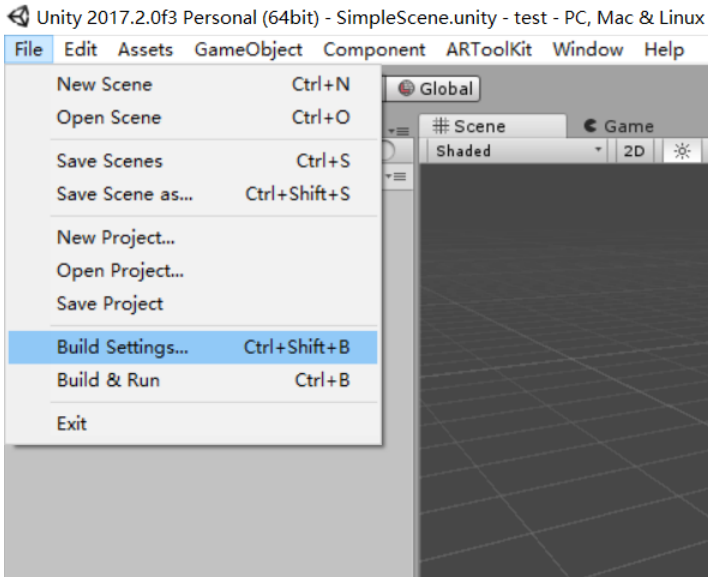
6）在 Marker scene 下创建需要显示的虚拟物体（如 cube），scale 属性可以设置为{0.08, 0.08, 0.08}，position 属性可以设置为{0, 0.04, 0}

7）创建完成，点击 play 验证。

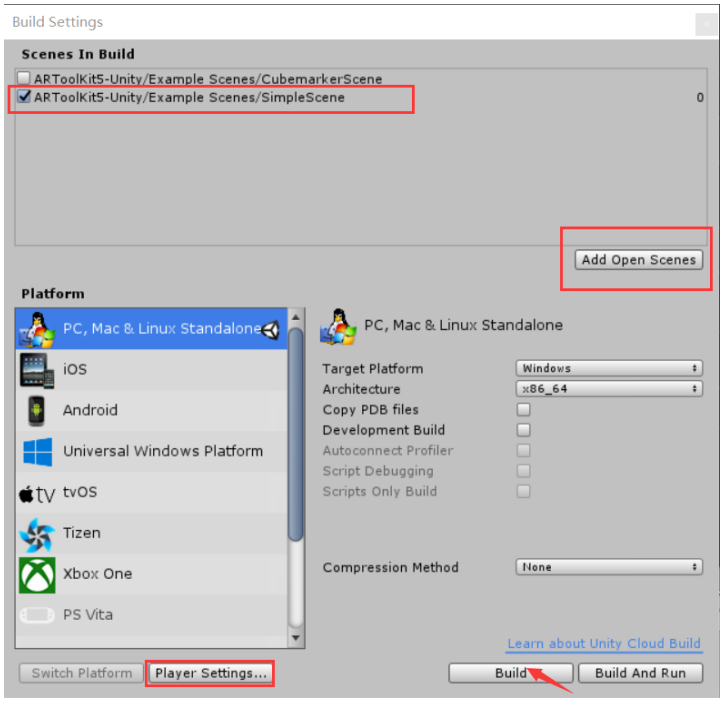
七、包含 ARToolkit 的 unity 场景发布（windows/android）步骤及一些错误处理

1.发布到 windows 平台

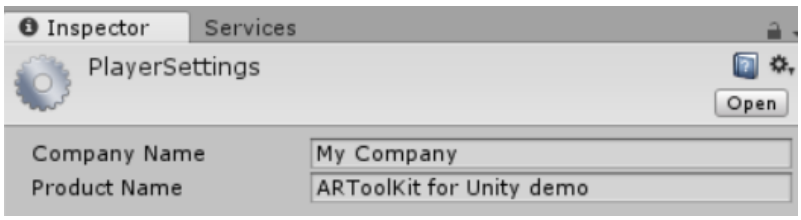
1）首先双击需要发布的场景，在 File 菜单栏下点击 Build Settings



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）

Java SE Development Kit 8u152

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

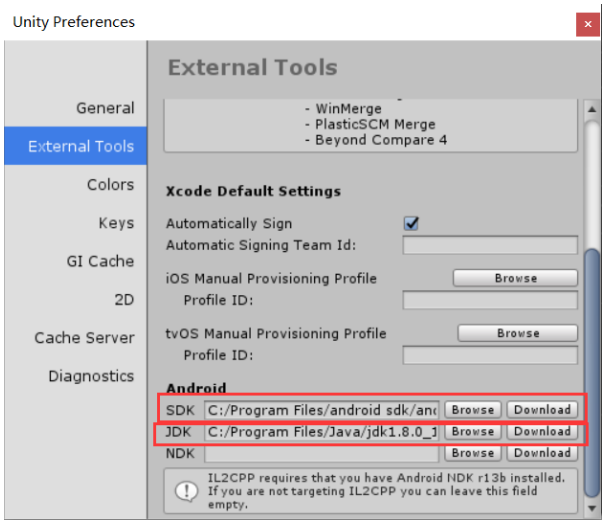
Accept License Agreement Decline License Agreement

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.94 MB	jdk-8u152-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	74.88 MB	jdk-8u152-linux-arm64-vfp-hflt.tar.gz
Linux x86	168.99 MB	jdk-8u152-linux-i586.rpm
Linux x86	183.77 MB	jdk-8u152-linux-i586.tar.gz
Linux x64	166.12 MB	jdk-8u152-linux-x64.rpm
Linux x64	180.99 MB	jdk-8u152-linux-x64.tar.gz
macOS	247.13 MB	jdk-8u152-macosx-x64.dmg
Solaris SPARC 64-bit	140.15 MB	jdk-8u152-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	99.29 MB	jdk-8u152-solaris-sparcv9.tar.gz
Solaris x64	140.6 MB	jdk-8u152-solaris-x64.tar.Z
Solaris x64	97.04 MB	jdk-8u152-solaris-x64.tar.gz
Windows x86	198.46 MB	jdk-8u152-windows-i586.exe
Windows x64	206.42 MB	jdk-8u152-windows-x64.exe

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/

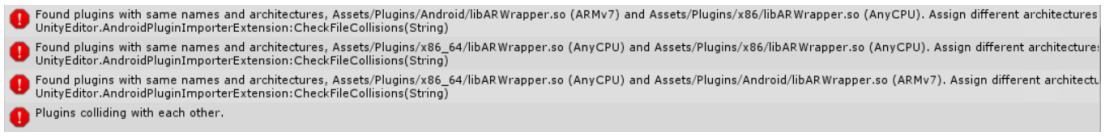
关于 Android sdk 的下载和配置可以参考这个帖子：
http://blog.csdn.net/love4399/article/details/77164500
唯一需要注意的是，在用 SDK Manager 下载 API 时，应该保证 API level≥25

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

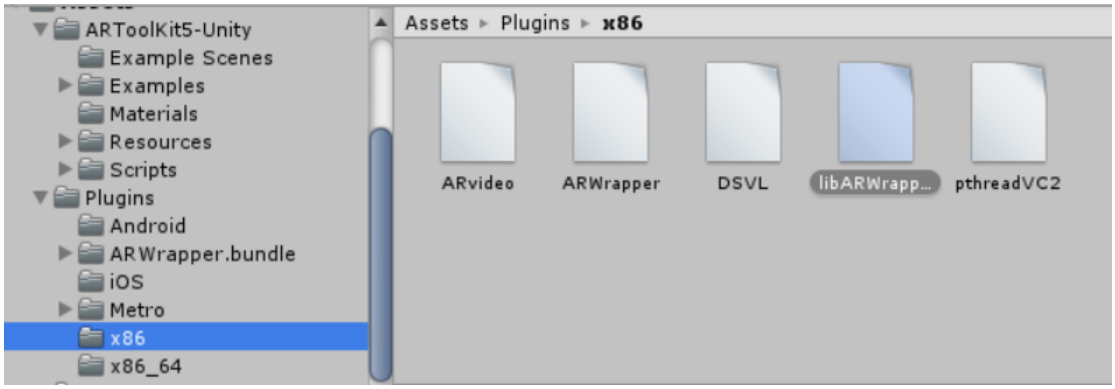


之后进入 Build Settings，选择 Android，在添加完需要发布的场景后，点击 build 即可。

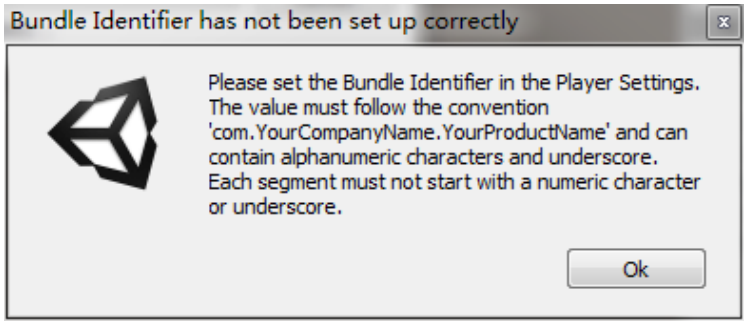
错误处理：
如果出现这样的错误：



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