

目 录

| | |
|-------------------------|------------|
| 概述 | 1 |
| 实验要求 | 1 |
| 主要工作 | 1 |
| 背景 | 1 |
| Grasping&6D Pose | 1 |
| 6D Pose 算法简介 | 2 |
| 从 6D Pose 到瓶盖姿态检测 | 2 |
| 检测-匹配算法 | 3 |
| 数据集拍摄 | 3 |
| 瓶盖选择 | 4 |
| 拍摄环境与设置 | 4 |
| 标准集拍摄 | 4 |
| 测试集拍摄 | 4 |
| 检测 | 4 |
| Selective Search 产生建议区域 | 4 |
| 利用饱和度和面积去除干扰物和复杂背景的影响 | 5 |
| NMS 消除冗余的区域建议 | 5 |
| 检测中的旋转算法 | 6 |
| 匹配 | 6 |
| 单个瓶盖的匹配 | 7 |
| 局部特征 vs 全局特征 | 7 |
| 实验结果 | 8 |
| 匹配结果 | 8 |
| UI 界面设计 | 10 |
| 感想 | 错误! 未定义书签。 |

1 概述

1.1 实验要求

设计一个瓶盖检测算法，布置一个自定义背景，在背景上随机放置 10 个瓶盖，拍摄 10 张以上不同瓶盖分布的图片，检测算法能够把这 10 张图片中各个瓶盖的位置和姿态（正、反、侧）都检测出来。

输入：一张有 10 个瓶盖的图片

输出：一张新图片，图片上用不同颜色把姿态不同的瓶盖标记出来，并显示各瓶盖上中心点的坐标。

1.2 主要工作

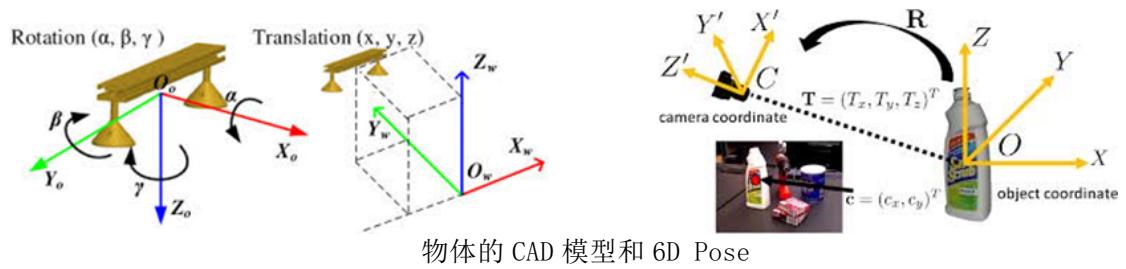
针对本次实验要求，以及内容的划分，我们所做的主要工作为：

1. 选取合适的瓶盖，拍摄标准集与测试集
2. 检测出图片中的所有瓶盖，旋转到同一角度
3. 使用局部特征与全局特征匹配瓶盖
4. 将匹配结果可视化，用 UI 界面展示

2 背景

2.1 Grasping & 6D Pose

在 6D Pose 问题中，输入往往含有深度信息，一般通过深度相机或者多视角摄像机得到，即 RGBD 或者 Multiview-RGB 图片。求解的解空间一般是 6D 解空间，难点是怎么在有遮挡和形变的情况下成功匹配图片中的物体和 CAD 模型。



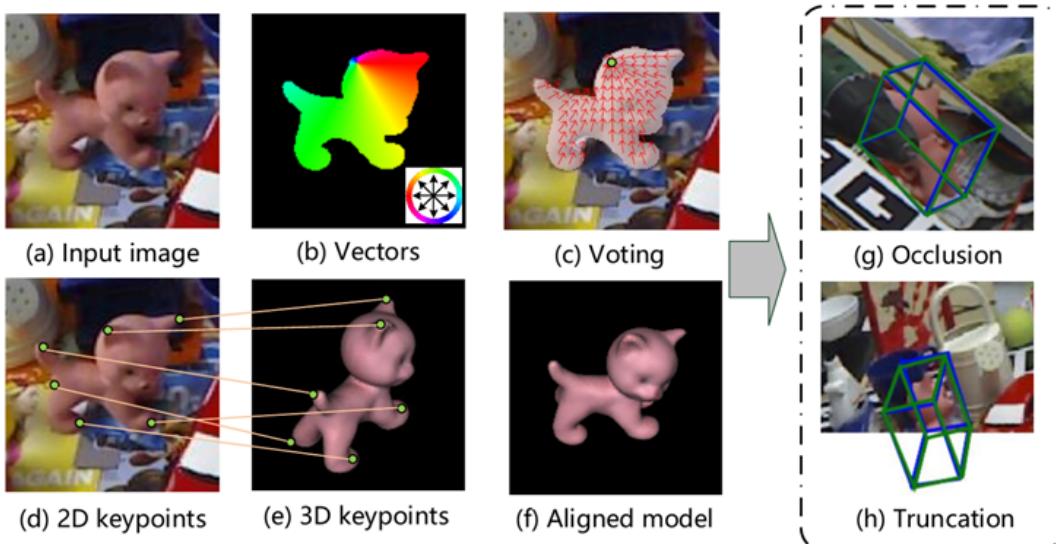
2.2 6D Pose 算法简介

传统算法：

以 Stefan Hinterstoisser 的 LINEMOD 为代表。方法是在可能的 SE3 空间通过渲染对要检测的物体作充分的采样，提取足够鲁棒的模板，再对模板进行匹配就可以预测 6D Pose，最后通过 ICP 精化结果。或者在点云上面用少量的点对构成描述子来做的，以描述子为 key，在从 scene 点晕中匹配的时候，在模型 hash 表中查询，这样可以得到所有可能匹配点法对。如果成功匹配，可以计算出其变换的刚体变换矩阵，也就是要求的 6D Pose。

基于深度学习的算法：

学习一个端到端的深度神经网络直接预测物体的 6D Pose。



神经网络端到端的预测算法

2.3 从 6D Pose 到瓶盖姿态检测

从 6D Pose 问题到瓶盖姿态检测问题，输入和要求解的空间都发生了变化。首先是输入的信息变少了，我们所用的输入是单目 RGB 图片，没有了深度信息，所以我们需要用先验知识来限制解空间，减轻因为缺少深度信息而导致的不适用的问题。在拍摄中我们可以限制瓶盖的位置空间（在画布上），以及瓶盖可能具有的三个姿态（正、反、侧），从而把解空间从 6D 降

维到了 3D 空间加上一个三分类的问题。

因为现在我们并没有得到瓶盖的 CAD 模型，所以为了得到瓶盖的上每个点的精确坐标，我们需要构造一个 2D 的标准集代替 3D 的 CAD 模型，用来得到测试图片上的像素与瓶盖实体的对应关系。

最后我们要考虑到干扰物和复杂背景的影响，所以我们要将算法分为两个部分，检测阶段和匹配阶段，两个阶段相对独立，方便开发、维护和升级。

3 检测-匹配算法

3.1 数据集拍摄

3.1.1 瓶盖选择

我们选取了三种不同颜色与特征的瓶盖：农夫山泉的红色瓶盖，正面有凹进的农夫山泉标志；芬达的蓝色瓶盖，正面有白色的芬达字样；阿华田的橙色瓶盖，正面无明显 logo，背面有较特殊的纹样。

3.1.2 拍摄环境与设置

拍摄数据集包括拍摄标准集和测试集两个部分。为了方便检测与匹配，要求拍摄中尽量保持拍摄条件一致，减少阴影与视差。

具体的拍摄环境为：在具有充足自然光的室内，使用 50mm 镜头的微单相机，固定快门与光圈参数（分别为 1/200, f/3.5），保证对于标准集和测试集的拍摄条件是一样的。使用三脚架固定相机进行拍摄，镜头方向垂直于地面。瓶盖布置在白色卡纸上，放置于镜头下方。

3.1.3 标准集拍摄

拍摄标准集时，我们希望收集到瓶盖不同角度的样本，以在匹配时进一步减少视差的影响，因此我们选择了九宫格采样法。我们把九个瓶盖按九宫格均匀布置在背景卡纸上，从背景中心点进行拍摄，并对瓶盖的正面、侧面、背面分别拍摄一次，得到了三个姿态下的九宫格标准集。

3.1.4 测试集拍摄

在测试集的拍摄中，我们希望尽量保持瓶盖分布的随机性，并且控制了瓶盖不同姿态（正面、反面、侧面）的比例。同时，我们也拍摄了多种不同难度的测试集，包括白色背景下的单色瓶盖、2-3 种不同颜色瓶盖混合、含有干扰物的多色瓶盖、杂色背景下的瓶盖等。

3.2 检测

我们首先通过通过饱和度将图片的背景滤去，然后用 Selective Search 算法给出多尺度的区域建议，再通过饱和度和面积筛选出可能的瓶盖区域，用 NMS 合并冗余的建议区域。

3.2.1 Selective Search 产生建议区域

Algorithm 1: Hierarchical Grouping Algorithm

Input: (colour) image

Output: Set of object location hypotheses L

Obtain initial regions $R = \{r_1, \dots, r_n\}$ using [13]

Initialise similarity set $S = \emptyset$

foreach Neighbouring region pair (r_i, r_j) **do**

 Calculate similarity $s(r_i, r_j)$

$S = S \cup s(r_i, r_j)$

while $S \neq \emptyset$ **do**

 Get highest similarity $s(r_i, r_j) = \max(S)$

 Merge corresponding regions $r_t = r_i \cup r_j$

 Remove similarities regarding r_i : $S = S \setminus s(r_i, r_*)$

 Remove similarities regarding r_j : $S = S \setminus s(r_*, r_j)$

 Calculate similarity set S_t between r_t and its neighbours

$S = S \cup S_t$

$R = R \cup r_t$

Extract object location boxes L from all regions in R

Selective Search 的策略是，先得到小尺度的区域，然后合并得到更大的尺寸。在合并过程中用到了颜色距离、纹理距离等多种特征来保持特征的多样性。

首先通过基于图的图像分割方法初始化原始区域，就是将图像分割成很多很多的小块。然后使用贪心策略，计算每两个相邻的区域的相似度，然后每次合并最相似的两块，知道最后得到最大的建议区域-原图。其中每次产生的图像块包括合并的图像块都保存下来，作为图像的分层表示。

3.2.2 利用饱和度和面积去除干扰物和复杂背景的影响

因为我们有瓶盖颜色和大小的先验知识，所以我们能根据颜色的饱和度和面积大小判断出哪些建议区域可能包括瓶盖，哪些建议区域可能包括干扰物。这里我们设置了建议区域内饱和度和面积大小的阈值，筛选出最可能包含瓶盖的区域。

3.2.3 NMS 消除冗余的区域建议

```
1 i←2;
2 while i<=W-1 do
3     if I[i]>I[i+1] then
4         if I[i]>=I[i-1] then
5             MaximumAt(i);
6         else
7             i←i+1;
8         while i<=W-1 AND I[i]<=I[i+1] do
9             i←i+1;
10        if i<=W-1 then
11            MaximumAt(i);
12        i←i+2;
Algorithm:1D NMS for 3-Neighborhood
```

首先分别计算每个建议区域的面积，把面积作为置信度进行排序，每次选取置信度最大的建议区域，计算该建议区域和其余建议区域的交集，计算置信度最高的建议区域和其他建议区域交集与它们的面积比，如果大于预设的阈值，那么该建议区域不作为最后保留的区域。重复直到建议区域不再发生变化，最后保留的建议区域进行并集操作。



左图含有冗余的建议区域，右图是用 NMS 消除冗余区域之后的效果。

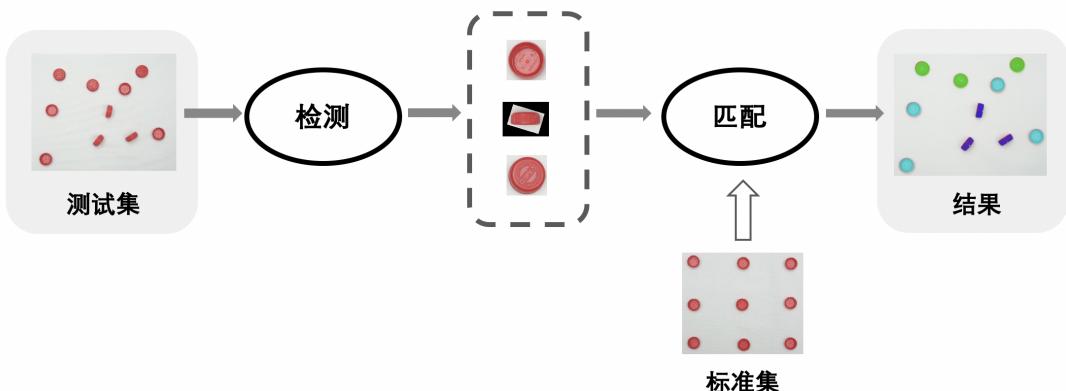
3.2.4 检测中的旋转算法

为了方便后续的匹配，我们需要把检测到瓶盖的每个区域旋转到统一的方向。我们计算瓶盖边缘与图片边界距离的方差，选取方差最小的图片旋转角度，用作匹配。在边缘距离的方差最小时，瓶盖通常旋转到了水平角度。

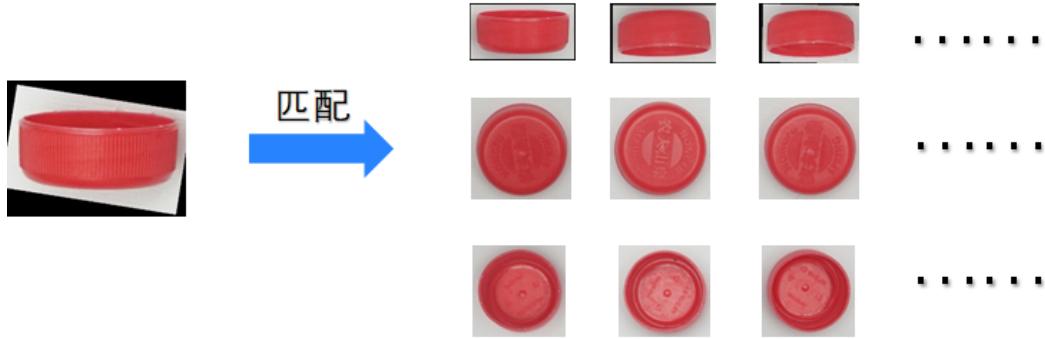


3.3 匹配

在检测阶段我们把测试图片中的瓶盖切分了出来，在匹配阶段我们要在标准集里面寻找和测试瓶盖最相近的图片，也就是我们要设计一个距离来找到距离，然后要最小化这个距离。

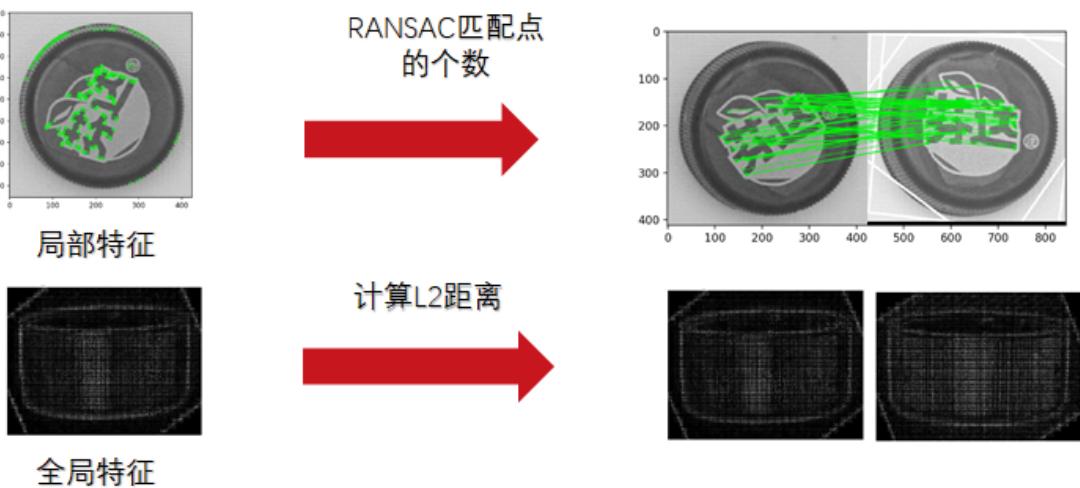


3.3.1 单个瓶盖的匹配



因为我们要对不同种类的瓶盖用一个统一的算法进行匹配，所以在匹配过程中我们必须要考虑不同瓶盖具有不同的颜色、花纹、图案、形状等特征。所以我们需要设计一个可以综合不同特征的距离来描述测试图片和标准集里图片的相似程度。

3.3.2 局部特征 vs 全局特征



我们使用 SIFT 提取局部特征，然后使用 RANSAC 实现局部特征之间的匹配。我们使用 HOG 提取全局特征，然后计算测试图片和标准集里图片 HOG 的 L2 距离。

$$S = \underset{s}{\operatorname{argmin}} \{ \hat{1}^{SIFT} \cdot dis^{SIFT}(F_{test}, F_s) + \hat{1}^{HOG} \cdot dis^{HOG}(F_{test}, F_s) \}$$

我们将全局特征和局部特征放在一起考虑，当某个特征匹配失败后，对应的示性函数值为 0，表示不考虑这个特征的结果。如果某个特征匹配成功，对应的示性函数值为 1，表示将这个特征匹配的结果容易考虑。我们用 softmax 加权组合不同特征匹配的结果，最后得到最优匹配的标准集图片。

匹配算法的本质是建立起了测试图片和标准图片一一对应的关系，我们只要标记处标准集图片

里面的任一点，我么就可以得到测试集图片里面这一点的精确位置。



用 labelme 标出瓶盖的轮廓和原点。

4 实验结果

4.1 匹配结果

由于我们在 labelme 中标注了每个标准集中瓶盖的边缘与圆心坐标，只要匹配到正确的瓶盖，即可一一对应地得到测试集中瓶盖的边缘与圆心坐标。为了清晰地显示匹配结果，我们在输入的瓶盖图片上，使用三种颜色的半透明遮盖来标记三种不同姿态的瓶盖，并且使用白色原点标记出瓶盖的中心坐标，如图所示。



匹配完成的输出图片

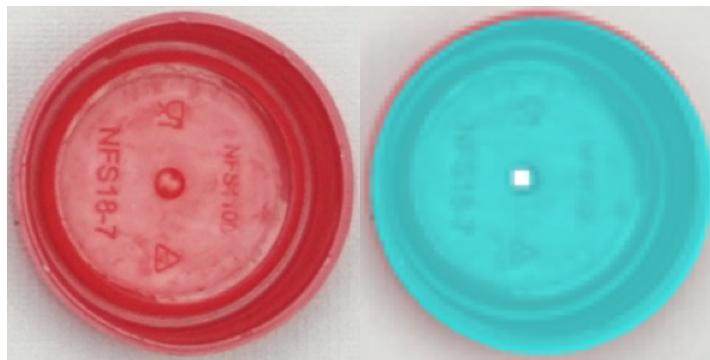
以下分别选取了正面、侧面、背面瓶盖的典型匹配结果。在一定的视差下，我们的算法依然能准确匹配到瓶盖的姿态与圆心。

对于正面瓶盖的匹配，所选图片的观察角度有一定向左上方的倾斜，但是由于标准集中拍摄了相同角度的瓶盖，且测试集与标准集正确匹配，因此我们的算法能够准确地标出瓶盖的中心点坐标。



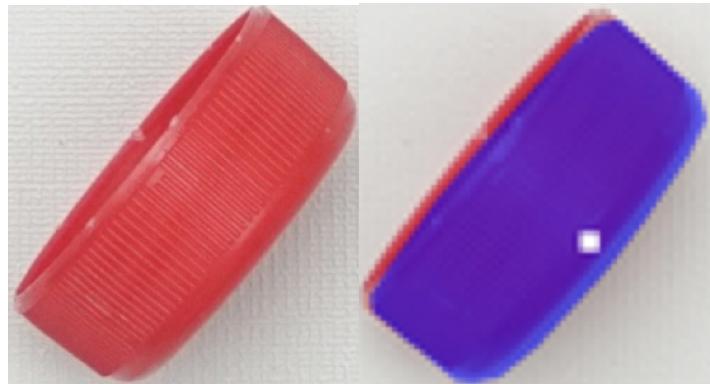
正面瓶盖的匹配示例

对于背面瓶盖的匹配，所选图片的观察角度有一定向右侧的倾斜，同样地，标准集中拍摄了相同角度的瓶盖，且测试集与标准集通过背面的特征相匹配，准确地标出瓶盖的中心点坐标。



背面瓶盖的匹配示例

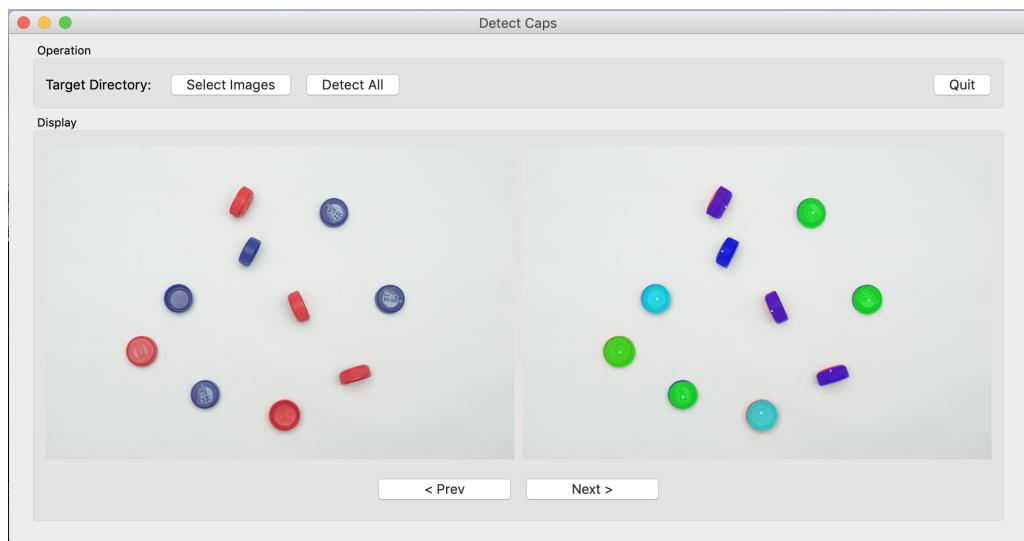
对于侧面瓶盖的匹配，所选图片有一定的视差，并且朝向左上方。我们能够通过旋转瓶盖朝向后再进行匹配，同样能避免视差的影响，准确匹配到瓶盖的中心点与朝向。



侧面瓶盖的匹配示例

4.2 UI 界面设计

为了让用户能够方便地使用我们的系统，我们设计了友好的用户界面。用户界面封装了检测与匹配算法，主要功能为：选择多张图片，检测所有图片中的瓶盖并返回匹配结果，并可使用 prev 和 next 按钮在匹配完成的图片间切换查看。



UI 界面截图

5 程序使用说明

5.1 目录

```
/bottle_cap_detection  
    detect_caps.exec  
    match.dat  
    check_cap_open_direction.py  
    color.py  
    detection.py  
    gui.py  
    hog.py  
    rotation.py
```

```
sift.py  
sift_display.py
```

5.2 环境

```
python>=3.6  
scikit_image==0.16.2  
selectivesearch==0.4  
opencv_python==4.1.2  
numpy==1.14.2  
labelme==3.18.0  
Pillow==7.0.0  
PyQt5==5.14.0  
skimage==0.0
```

5.3 运行

5.3.1 运行可执行文件

在 macOS 系统上，可以直接在 `detect_caps` 所在目录中运行 exec 可执行文件，无需搭建环境。

```
$ ./detect_caps
```

5.3.2 运行 python 脚本

```
$ pip install -r requirements.txt  
$ python gui.py
```

5.4 UI 使用说明

点击 `select images` 选择多张图片，点击 `detect all` 开始瓶盖检测算法。检测完成后，会

显示用颜色标记的瓶盖位置，方向和中心。点击 prev 和 next 可切换图片。

6 感想

经过三周的努力，我们圆满完成了此次瓶盖检测匹配算法的大作业。这次作业是一个计算机视觉方向比较完整的项目，从制作数据集到设计算法，程序编写，再到调整和展示，各方面的参与让我们对工业领域中如何完成高质量的物体姿态检测算法有了大致的了解。经过不断的尝试，我们阅读了多种算法的原理，并根据我们数据集的特点选择了最为合适的算法。为了达到最佳的匹配效果，我们也尝试结合不同的特征进行匹配；同时我们也认识到数据准备、预处理是除了算法之外能够很大影响到匹配准确度的因素。

最后，感谢孙炎老师与助教为我们设计项目，热情地解答疑惑，让我们在学习过程中收获了很多知识，也在实践中对于课程内容有了更深刻的理解。感觉各位组员的合作，让我们共同完成了这个项目。