60分钟轻松搞定树莓派 AI 服务开发

1. 引言

目前，物联网、人工智能已经深入到医疗、家居、交通、教育和工业等多个领域，正在极大改变人们的日常生活。树莓派受众多物联网技术爱好者和创客的欢迎，除官方的 Raspbian 系统以外，还可以运行微软的 Windows 10 IoT Core 和 Google 的 Android Things 等面向物联网应用的操作系统。微软认知服务（Microsoft Cognitive Services）集合了多种智能 API 以及知识 API，能够运行在 Windows、Linux 和 Mac 等多种平台。借助这些 API，开发者可以开发能看、能听、能说话，并且能理解和解读人类通过自然交流方法所传达的需求，从而创建更智能，更有吸引力的产品。

本场 Chat 将介绍如何在运行 Windows 10 IoT Core 的树莓派上开发微软认知服务，主要包括以下内容：

1. Windows IoT 上手：Windows IoT 运行平台、开发环境搭建和系统烧写。
2. 微软认知服务介绍：微软认知服务的分类、微软认知服务的创建。
3. 开发基于 Windows IoT 的认知服务：Windows IoT 运行认知服务所需硬件资源、计算机视觉服务开发、人脸识别服务开发、情绪认知服务开发。

2. Windows IoT 上手

2.1 Windows IoT 运行平台

从 Windows 10 开始，微软针对操作系统的产品规划有了较大的变化，对平板、PC、体感、游戏和物联网设备的操作系统都统一命名为 Windows 10 操作系统。同时，对于全新的平台，推出了 “通用应用” 的模型，真正实现了一个工程、全平台设备通用的目的。

针对物联网应用领域，微软推出了 Windows IoT 产品线。目前，Windows 10 IoT 分为两个分支，一个为 Windows 10 IoT Core，另一个为 Windows 10 IoT Enterprise，分别面向小型物联网应用和企业级物联网应用。本项目的运行平台为 Windows 10 IoT Core，它支持英特尔、高通和博通的一系列 SoC，涵盖了 ARM 和 x64 产品线，能够部署到近 60 种板卡中运行。其中，市场上常见的板卡包含树莓派（ARM）、DragonBoard 410c（ARM）、MinnowBoard MAX（x64）和 Intel Compute Stick（x64）。相比较而言，性价比最高的莫过于树莓派了。目前，能够运行 Windows 10 IoT Core 的树莓派型号为 Raspberry Pi 2B 和 Raspberry Pi 3B[1]，最新的 Raspberry Pi 3B + 还没有支持，但已经列入 Windows 10 IoT Core 的支持列表，相信不久微软就会开发出相应的 FFU。

更多有关运行 Windows 10 IoT Core 设备的信息，请参考以下链接：Windows 10 IoT Core Device List [2]。

2.2 开发环境搭建

目前，能够支持 Windows 10 IoT Core 开发的工具就是免费的 Visual Studio 2017 Community 版本。当然，Visual Studio 2015 Community 也支持面向早期 Build 型号（15063 之前）的应用开发，但还是建议大家安装 Visual Studio 2017。在安装 Visual Studio 2017 Community 版本时，请勾选 Universal Windows Platform development 的选项。另外，对于想要使用 C++ 开发应用的朋友，还需要勾选 C++ Universal Windows Platform tools 的选项。如下图 1 所示。

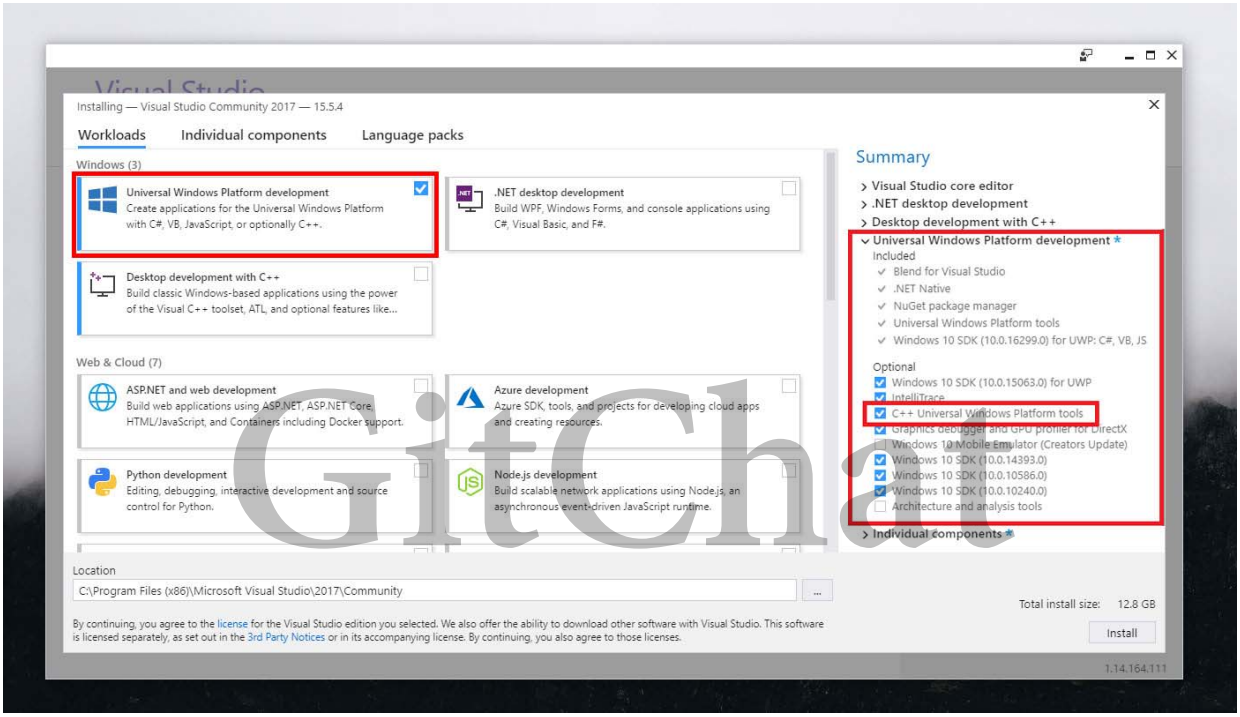

图 1：Visual Studio 2017 安装选项

2.3 Windows 10 IoT Core 系统烧写

对于树莓派而言，其系统是烧写在外部的 SD 中的，微软提供了一个图形化的系统烧写软件 Windows 10 IoT Core Dashboard。用户可以前往该链接下载：
http://go.microsoft.com/fwlink/?LinkID=708576

在 SD 卡选择上，建议使用高速的 SD 卡，如 Samsung 32GB EVO Class 10 Micro SDHC 和 SanDisk Ultra Micro SDHC 16GB。

安装完 Windows 10 IoT Core Dashboard，将 SD 卡插入读卡器，并将读卡器插入 USB，开始系统烧写。其界面如下图 2 所示。

图 2：Windows 10 IoT Core Dashboard 界面

其中，设备类型选 Raspberry-Pi，OS 版本可以选 Release 和 Insider Build 两种。建议选 Release 版本。设置设备名称和管理员密码，就可以下载并烧录系统到 SD 卡了。系统烧写完毕，就可以将 SD 卡拔出，插入到树莓派，准备给树莓派上电运行了。

2.4 树莓派的外围设备

树莓派能够正常工作，需要以下外围设备：

（1）电源：5V/2.5A 电源，Micro USB 接口。
（2）显示设备：可以接 HDMI 接口的显示器，或者使用有源 HDMI 转 VGA 模块，再转接 VGA 接口的显示器。这里需要注意的是，一定要使用有源的 HDMI 转 VGA 模块，不然将无法正常输出视频信号给显示器。
（3）鼠标和键盘：可以使用 USB 的鼠标和键盘，一般品牌的都支持。
（4）摄像头：目前，官方给出的摄像头中，仅支持 USB 接口的摄像头，如 Microsoft Lifecam 3000、Microsoft Lifecam HD-5000 和 Microsoft® LifeCam Studio 等。注意，树莓派官方的 CSI 接口的摄像头没有被 Windows 10 IoT Core 所支持。

有关 Windows IoT Core 支持的外围设备型号，请参考官方文档：hardware compatible list[3]。

3. 微软认知服务

3.1 微软认知服务的分类

微软认知服务 (Cognitive Service) 是其 AI 平台的一部分，其前身是牛津计划（Project Oxford）。在 2015 年 5 月，微软正式发布了牛津计划，它包含了人脸识别、语音处理和计算机视觉这三部分 API。经过近 3 年的发展，牛津计划已更名为认知服务，包含影像、语音、语言、知识和搜索这五大类服务，涵盖计算机影像、人脸、必应语音、说话人识别（预览）、必应拼写、文本分析、自定义决策服务（预览）和必应搜索等八个细分领域的 API。本场 Chat 所关注的认知服务主要包括计算机视觉和人脸这两个部分的 API。

3.2 微软认知服务的创建

目前，微软已经将认知服务托管于 Azure 平台。对于想要试用的用户，微软为开发者提供了一个月的免费试用机会。因此，这里将介绍两种不同的认知服务创建方法。方法一针对有 Azure 订阅的用户，方法二针对免费试用的用户。

3.2.1 针对 Azure 订阅用户

首先，对于具有 MSDN 订阅的用户，可以登录 Azure 的门户 [4]。点击 “Create a resource”，在搜索栏中，输入 “Computer Vision API” 进行搜索。接着，点击 “Create” 进行创建，指定 Computer Vision 的名称、位置（亚洲东部）和等级（此处我们可以选择 F0 免费的级别，包含每月 5000 次的事务）。进一步的定价信息，可以参考官方链接：认知服务定价 - 计算机影像 API[5]。Computer Vision Service 创建完成之后，可以点击资源中的 Keys 属性，并且拷贝 KEY 1 到本地，我们在后续的 UWP 应用程序中要用到它。如下图 3 所示。

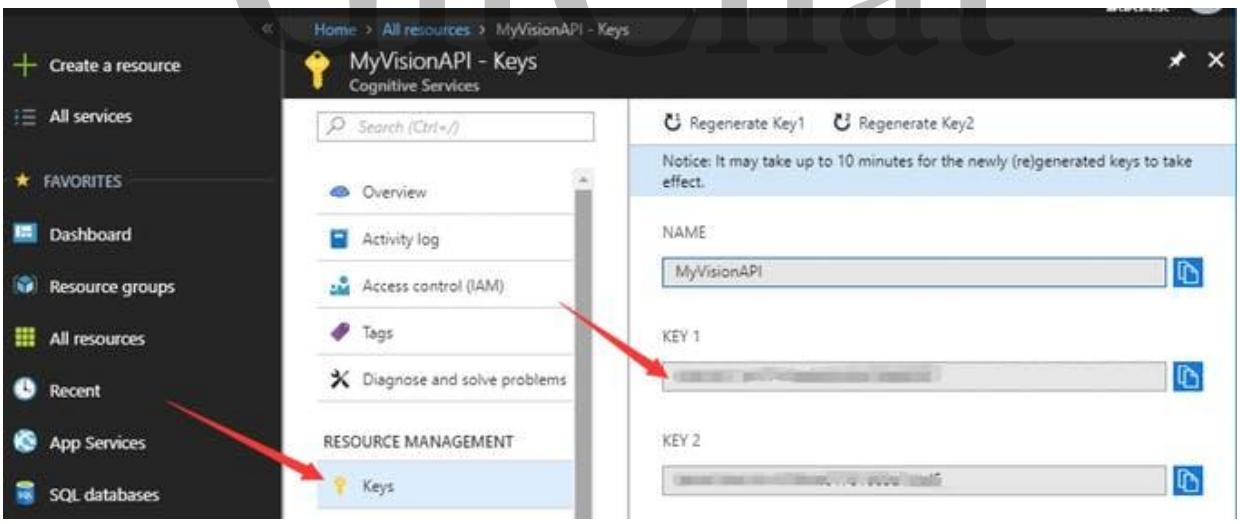

图 3 Computer Vision 属性页面

接着，使用同样的方法，在 Azure Portal 中创建 Face API 服务。注意，后面我们要用到的情感识别（Emotion API）服务已经于今年的 2 月份完成预览版，正式融入到 Face API 服务中。因此，这里只需要创建 Face API 资源就可以完成人脸和情绪的识别。Face API Service 创建完成之后，可以点击资源中的 Keys 属性，并且拷贝 KEY 1 到本地，我们在后续的 UWP 应用程序中要用到它。如下图 4 所示。

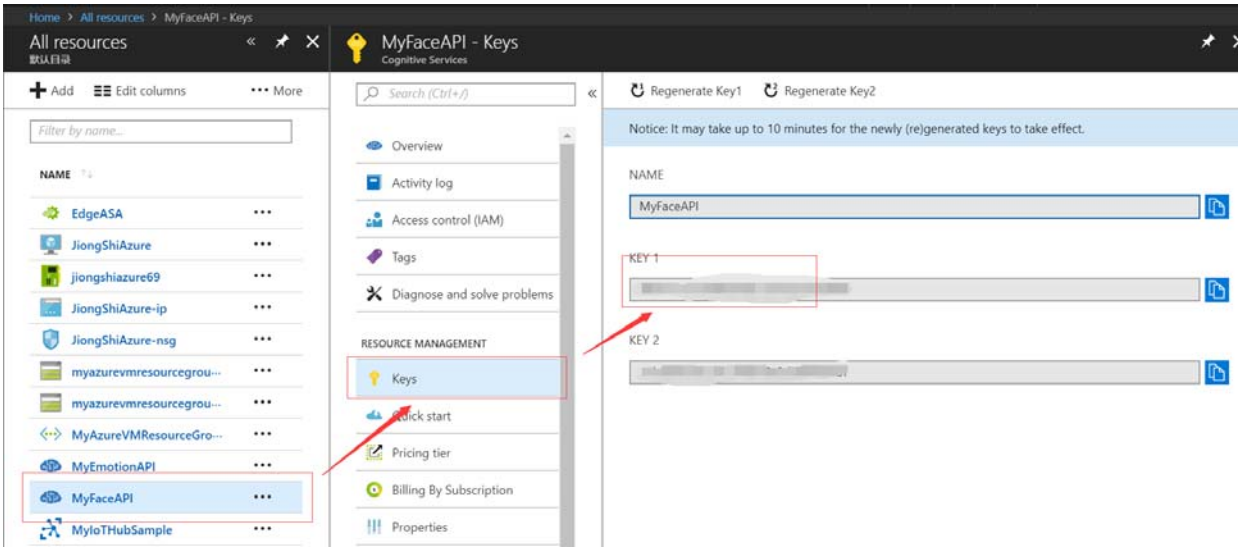

图 4 Face API 属性页面

3.2.2 针对试用的用户

如果没有 Azure 订阅，可以通过如下方法免费试用微软认知服务，免费试用的时间限制为 1 个月。用户首先导航到官方认知服务的试用页面：
https://azure.microsoft.com/zh-cn/try/cognitive-services/?api=computer-vision

从列表中选择需要试用的服务，例如，这里选择计算机影像（Computer Vision API），如下图 5 所示。

图 5：认知服务试用页面

之后，选择国家和地区为中国，使用微软账户、LinkedIn 账户、Facebook 账户或者 Github 账户中的任意一个账户就可以登录进行创建。之后，记住生成的 API KEY，拷贝到本地。如下图 6 所示。

图 6： 认知服务 API KEY 页面

接着，用同样的方法生成 Face API 的 KEY，并拷贝到本地。

至此，我们已经完成微软认知服务的创建，生成需要的 API KEY，并保留到本地，为后续的 UWP 应用开发做好了准备工作。

4. 基于 Windows IoT 的认知服务开发

4.1 Windows IoT 运行认知服务所需硬件资源

本项目使用树莓派，如果仅仅是对树莓派本地保存的图片进行识别，那就不需要摄像头。大部分应用场景中，需要完成实时图片的捕获和识别，则需要摄像头的支持。正如前文所述，目前 Windows IoT Core 仅支持 USB 接口的摄像头，如 Microsoft Lifecam 3000、 Microsoft Lifecam HD-5000 和 Microsoft® LifeCam Studio 等。注意，树莓派官方的 CSI 接口的摄像头没有被 Windows 10 IoT Core 所支持。另外，Microsoft Lifecam 3000 已经包含了麦克分，同样可以进行语音相关的认知服务。

4.2 项目的创建和设置

在 Visual Studio 中新建工程，模板选择 Visual C#->Windows Universal，将其命名为 RPiCognitiveService，如下图 7 所示。

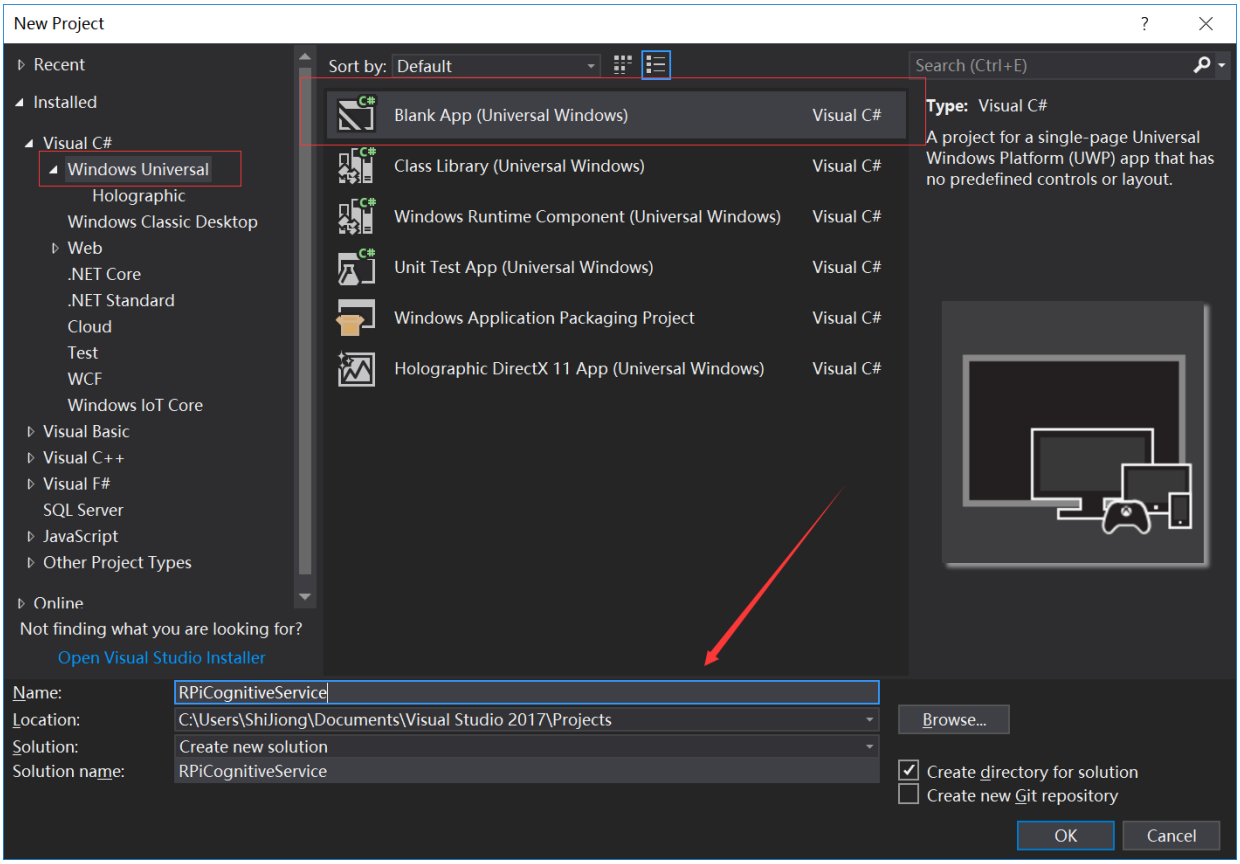

图 7：新建工程界面

新建工程之后，双击打开 Package.appxmanifest 文件，确认 Internet（Client）、Web Camera、Pictures Library 已经勾选，因为后续应用程序要使用网络、摄像头和 Picture 存放目录。

接着，选中项目，右键菜单中选择 Manage NuGet Packages，如下图 8 所示。

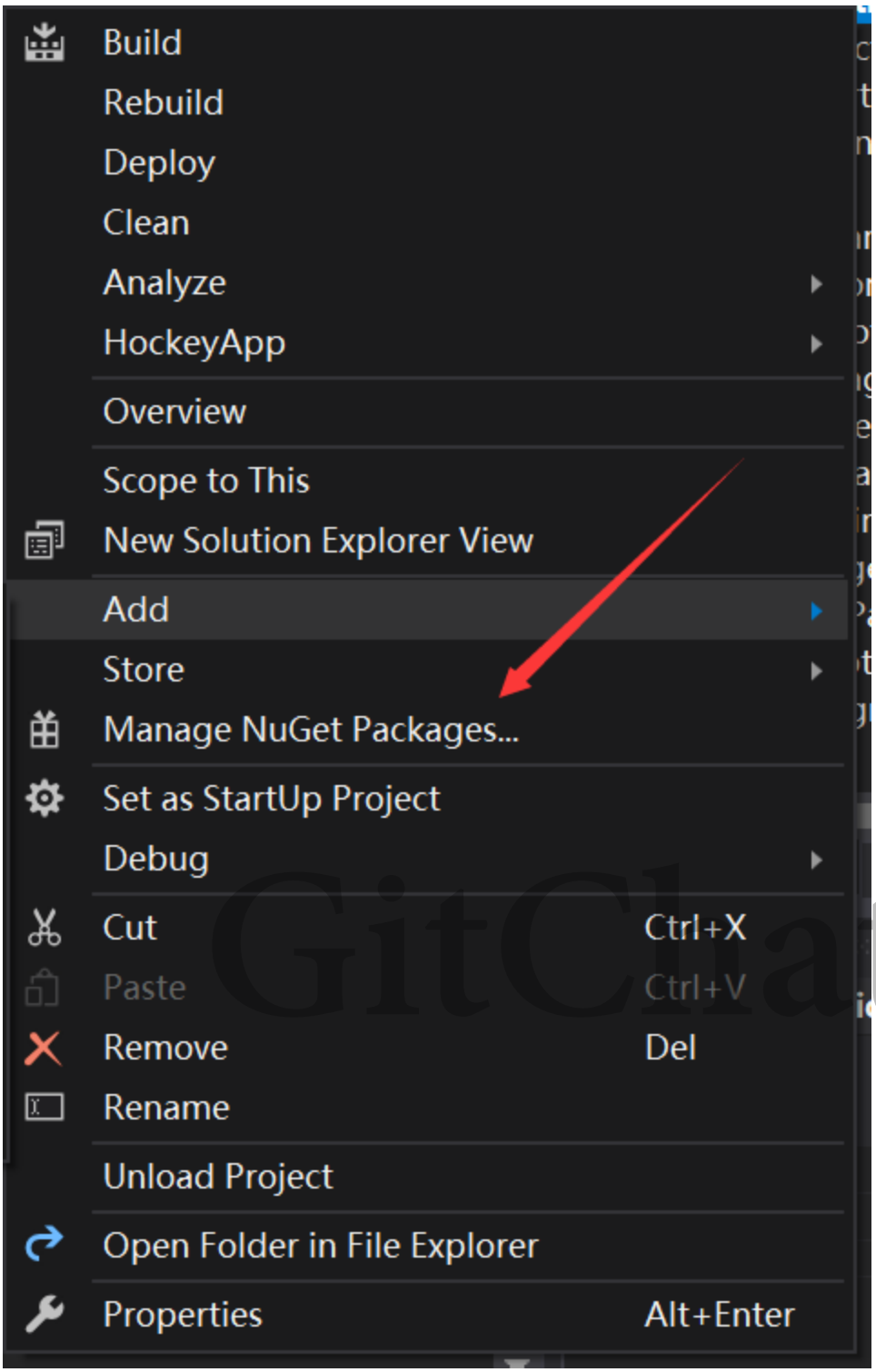

图 8： Manage NuGet Packages 界面

在 Browse 中分别输入 Microsoft.ProjectOxford.Commom、 Microsoft.Project Oxford.Face 和 Microsoft.ProjectOxford.Vision，对搜索到的 NuGet 包进行安装，之后，可以在已 Installed NuGet 包界面中找到已安装的组件，如下图 9 所示。

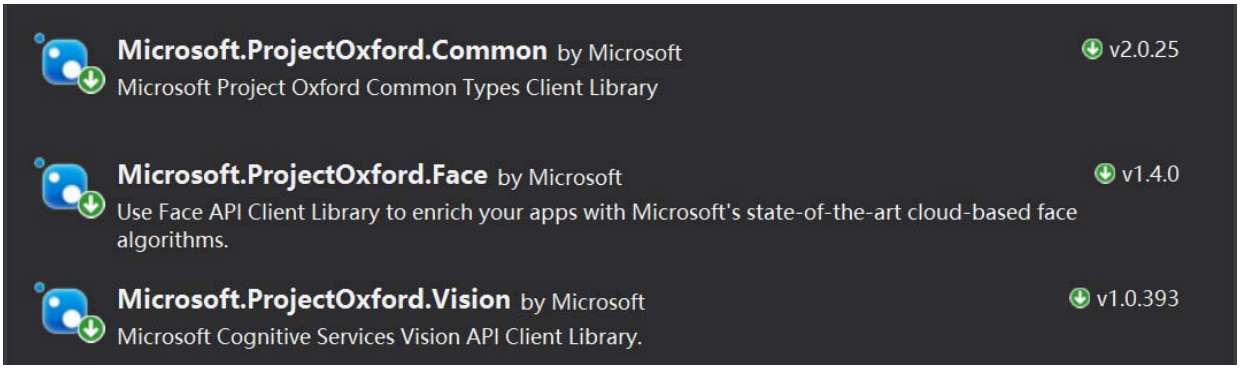

图 9：Installed NuGet 包界面

4.3 界面设计

4.3.1 主页面

主界面 MainPage 采用类似汉堡菜单的设计方式，使用 SplitView 进行分割。左边的 SplitView 嵌套 ListView，在其中加入三个 ListViewItem。右边的 SplitView 直接嵌套 Frame，其设计如下图 10 所示。

图 10：主界面设计

4.3.2 计算机视觉服务页面

计算机视觉服务页面（PhotoPage）主体由四部分组成，分别用于显示标题、按钮操作、图片显示和识别结果显示。界面设计如图 11 所示。

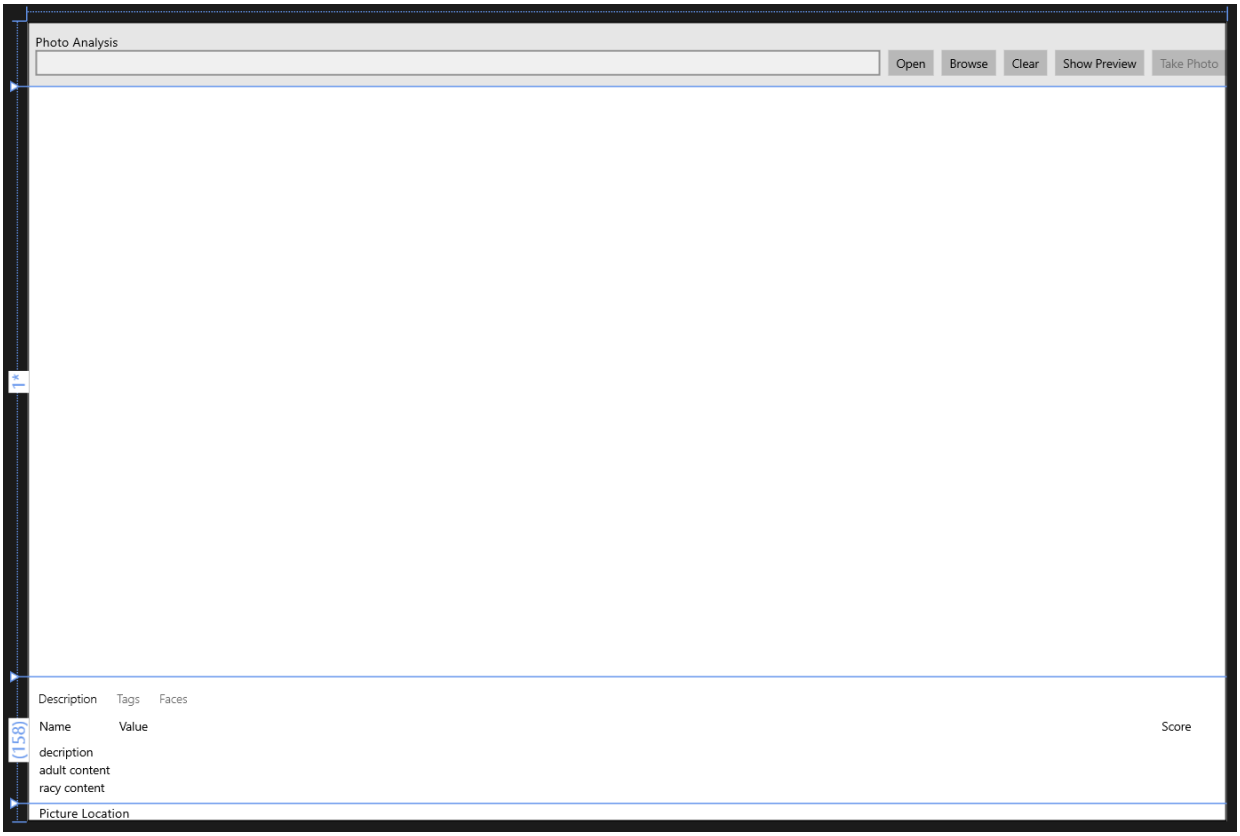

图 11： 计算机视觉服务页面设计

其中，识别结果的显示采用 Pivot 控件设计，分别加入 3 个 PivotItem 用于显示描述（Description）、标签（Tags）和人脸（Faces）。而 Description 中又包含整体描述、是否涉黄（adult content）以及是否涉及低俗的内容（Racy content）。5 个按钮用于打开、浏览、清除当前图片，以及显示摄像头实时预览、拍照识别等功能。

4.3.3 人脸和情绪识别页面

人脸和情绪识别页面（FacePage）的设计类似于计算机视觉服务页面。主体由四部分组成，分别用于显示标题、按钮操作、图片显示和识别结果显示。界面设计如图 12 所示。

图 12： 人脸和情绪识别页面

其中，识别结果的显示采用 Pivot 控件设计，分别加入两个 PivotItem 用于显示人脸（Faces）识别结果和情绪（Emotions）识别结果。人脸识别结果包含年龄、性别、是否微笑和是否佩戴眼镜这四部分内容。情绪识别结果包含 Anger、Contempt、 Disgust、Fear、Happiness、 Neutral、Sadness 和 Surprise。5 个按钮用于打开、浏览、清除当前图片，以及显示摄像头实时预览、拍照识别等功能。

4.4 后台代码设计

4.4.1 主页面后台代码

主页面 MainPage 的后台代码 MainPage.xaml.cs 设计相对简单，只需要处理 ListView 控件的 SelectionChanged 事件即可。其流程如图 13 所示。

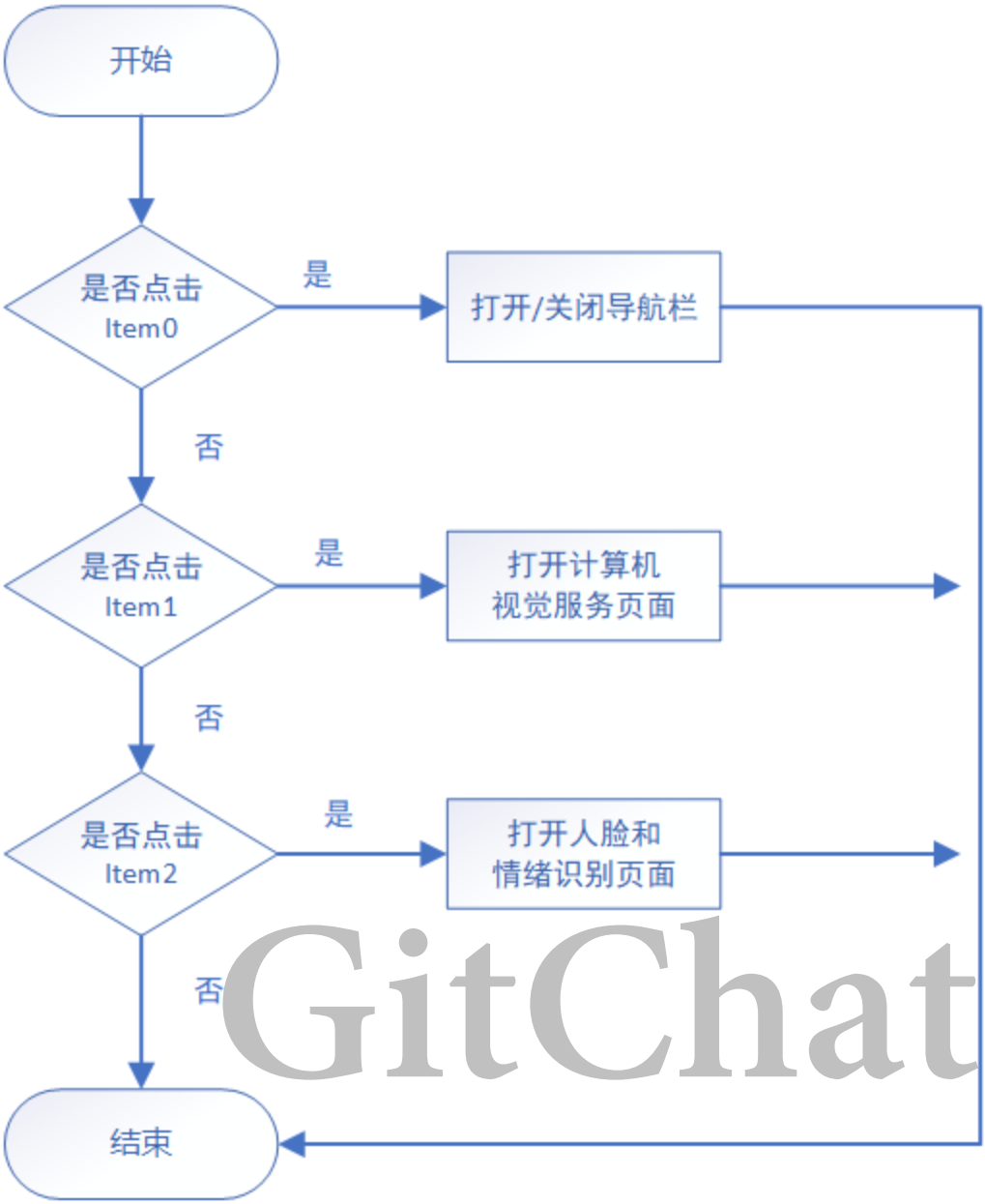

图 13：主页面 ListView 控件的 SelectionChanged 事件处理流程图

4.4.2 计算机视觉服务后台代码

在计算机视觉服务后台代码 PhotoPage.xaml.cs 设计中，首先声明了 Computer Vision API 相关的一些全局变量，如下代码段 1 所示。

代码段 1：PhotoPage 的部分全局变量

```
//Computer Vision API KEY
string key = "your Computer Vision Key";  //API key

Size size_image;  //当前图片的尺寸
AnalysisResult thisresult;  //分析结果
StorageFolder currentFolder;//文件夹
StorageFile Picker_SelectedFile;//选中的文件
private MediaCapture mediaCapture;// MediaCapture对象
private StorageFile photoFile;//存储的图片文件
private readonly string PHOTO_FILE_NAME = "photo.jpg";//图片文件的
```

```
文件名
private bool isPreviewing;//是否预览的状态指示
```

注意，这里需要将之前在 3.2 节中保存的 Computer Vision API Key 拷贝到代码段的第一行中，替换掉 “”your Computer Vision Key”。

在 PhotoPage 的构造函数中，加入对摄像头的初始化函数 initCamera ()。这么做的意图是一进入该页面，程序就初始化摄像头，为后续的拍照识别做好准备。该函数的关键代码如下。

代码段 2：摄像头初始化

```
// 初始化mediaCapture对象
mediaCapture = new MediaCapture();
await mediaCapture.InitializeAsync();
// 设置mediaCapture的mediaCapture_Failed回调
txtLocation.Text = "Device successfully initialized for video
recording!";
mediaCapture.Failed += new
MediaCaptureFailedEventHandler(mediaCapture_Failed);
// 开始图像预览
previewElement.Source = mediaCapture;
await mediaCapture.StartPreviewAsync();
isPreviewing = true;
txtLocation.Text = "Camera preview succeeded";
// 使能拍照按钮
btnTakePhoto.IsEnabled = true;
```


另外，在拍照按钮的用户单击事件处理程序 takePhoto_Click 中，完成照片拍摄、存储、识别和结果显示的工作。其处理流程如下图 14 所示。

图 14：拍照按钮处理流程图

其中，利用 Vision Service 进行图片处理时，直接调用 VisionServiceClient 的 AnalyzeImageAsync 方法与 Azure 上创建的 Computer Vision 服务进行交互，并将获得的结果显示在界面上，其关键代码如下。

代码段 3：Computer Vision 服务

```
//Computer Vision 服务
VisionServiceClient client = new VisionServiceClient(key);
var feature = new VisualFeature[] { VisualFeature.Tags,
VisualFeature.Faces, VisualFeature.Description,
VisualFeature.Adult, VisualFeature.Categories };
var result = await
client.AnalyzeImageAsync(stream_send.AsStream(), feature);
thisresult = result;
if (result != null)
{
       DisplayData(result);
}
```

4.4.3 人脸和情绪识别后台代码

情感识别（Emotion API）服务已经于今年的 2 月份完成预览版，正式融入到人脸（Face API）识别服务中，因此本项目情绪识别的结果包含在了人脸识别的结果之中。在人脸和情绪识别服务后台代码 FacePage.xaml.cs 设计中，同样需要注意的是：

将之前在 3.2 节中保存的 Face API Key 拷贝到代码段的第一行中，替换掉 “”your Face API Key”。

FacePage 的页面构造函数中加入了对摄像头的初始化，方法与前文代码段 2 类似，这里不再赘述。在拍照按钮的用户单击事件处理程序 takePhoto_Click 中，完成照片拍摄、存储、人脸识别和结果显示的工作，其流程与图 14 类似，这里不再赘述。与前文不同的是，这里使用 FaceServiceClient 的 DetectAsync 方法与 Azure 上创建的 Face API 服务进行交互，并将获得的结果显示在界面上，其关键代码如下。

代码段 4：Face API 服务

```
//Face API 服务
FaceServiceClient f_client = new FaceServiceClient(key_face);
var requiedFaceAttributes = new FaceAttributeType[] {
                                FaceAttributeType.Age,
                                FaceAttributeType.Gender,
                                FaceAttributeType.Smile,
                                FaceAttributeType.FacialHair,
                                FaceAttributeType.HeadPose,
                                FaceAttributeType.Emotion,
                                FaceAttributeType.Glasses
                                };
var faces_task = f_client.DetectAsync(stream_send.AsStream(),
true, true, requiedFaceAttributes);
faces = await faces_task;
if (faces != null)
{
     DisplayFacesData(faces);
```

```
            DisplayEmotionsData(faces);
        }
```

用户可以看到，请求的 Face 属性包含年龄、性别、微笑、头发、姿势、情绪和是否佩戴眼镜等内容。在返回识别结果之后，利用 DisplayFacesData 和 DisplayEmotionsData 这两个方法显示人脸识别和情绪识别的结果。

5. 测试

5.1 测试工具和软件

本应用程序采用 Visual Studio 2017 Community 版本进行开发，Windows Software Development Kit (SDK) 的版本为 16299。本项目的源代码已经开源到 Github，地址为：RPiCognitiveService [7]。同时，本项目已被微软 IoT Core 团队采用，编入 Windows-iotcore-samples 中，具体地址为："RPi Cognitive Service Sample"[8]。大家可以去上述的任意一个地址下载，并解压到本地。用 Visual Studio 2017 工具打开，修改两个文件中的 API KEY，选择 ARM 编译，填写树莓派的 IP 地址或名称，就可以将应用部署到设备上了。具体设置如图 15 所示。

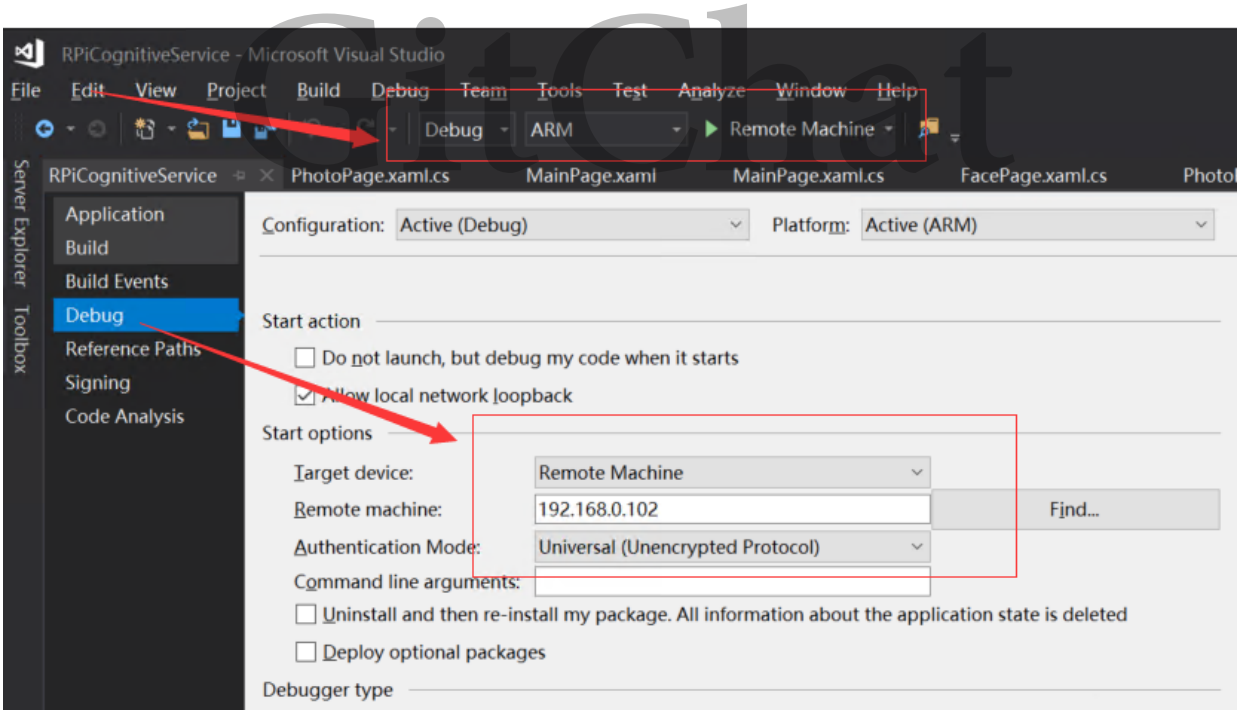

图 15: 应用程序部署设置

5.2 测试结果

应用程序运行之后，点击菜单，选择 PhotoPage 或者 FacePage 两者中的任何一个，页面会显示摄像头初始化信息，用户可以点击预览来查看摄像头的预览图像，如图 16 所示。

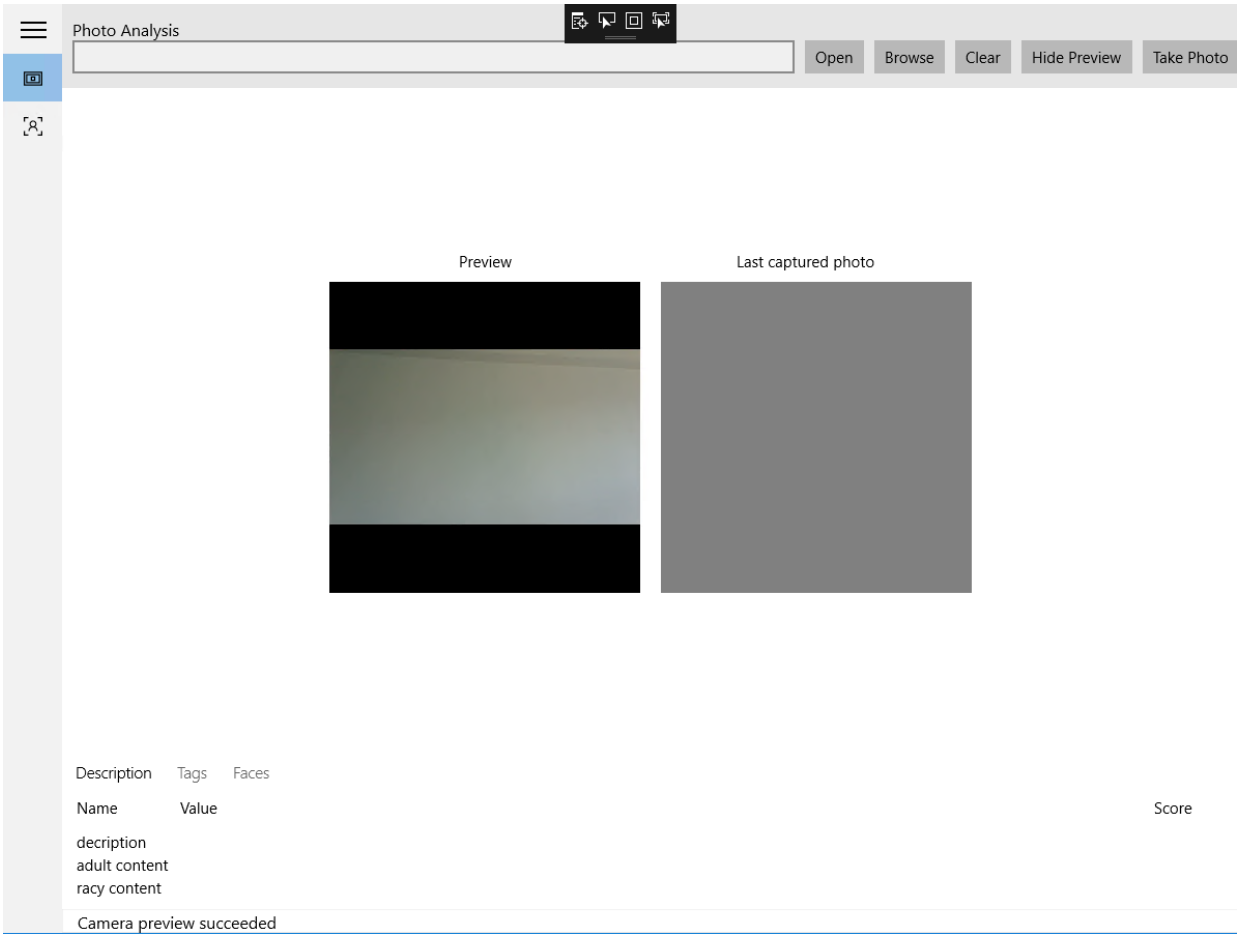

图 16：应用程序运行页面

选择要识别的场景，在点击 Take Photo 按键之后，应用程序会返回识别结果，并显示在页面上。如图 17 所示。

图 17：Computer Vision 识别结果页面

点击 FacePage 页面，选择要识别的场景，在点击 Take Photo 按键之后，应用程序会返回识别结果，并显示在页面上。如图 18 所示。

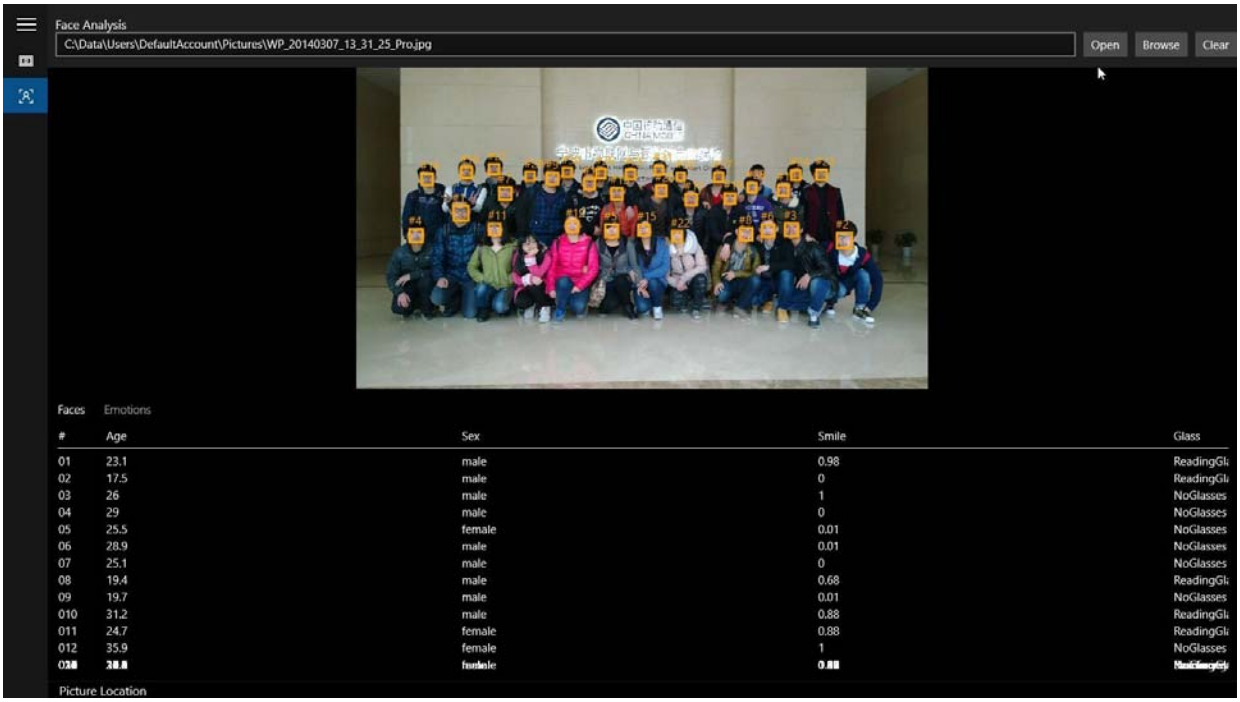

图18： Face API识别结果页面

6. 总结

本场 Chat 介绍了如何在运行 Windows 10 IoT Core 的树莓派上开发微软认知服务。首先，从 Windows IoT 上手开始，描述了 Windows IoT 运行平台、开发环境搭建和系统烧写的过程。接着，简要介绍了微软认知服务，主要包括微软认知服务的分类和创建，并针对具有 Azure 订阅和免费试用的两种场景给出了创建流程。然后，描述了开发基于 Windows IoT 的认知服务的要点，包括 Windows IoT 运行认知服务所需硬件资源、计算机视觉服务开发、人脸和情绪识别服务开发。最后，结合测试工具和环境，给出了测试的结果，达到了预期的目标。

更多有关项目信息，请大家参考 Github 链接：RPiCognitiveService。最后，对 Github 用户 sherlockchou86 表示感谢，本项目部分设计参考了您的项目。

参考链接：

[1] https://www.raspberrypi.org/products/
[2] https://developer.microsoft.com/en-us/windows/iot/getstarted/prototype/selectdevice
[3] https://docs.microsoft.com/zh-cn/windows/iot-core/learn-about-hardware/hardwarecompatlist
[4] https://portal.azure.com
[5] https://azure.microsoft.com/zh-cn/pricing/details/cognitive-services/computer-vision/
[6] https://azure.microsoft.com/zh-cn/try/cognitive-services/?api=computer-vision
[7] https://github.com/shijiong/RPiCognitiveService
[8] https://github.com/Microsoft/Windows-iotcore-samples/tree/develop/Samples/RPiCognitiveService