

#### **1）Airtest框架[¶](https://airtest.doc.io.netease.com/" \l "1airtest" \o "Permanent link)**

Airtest 是一个跨平台的、 **基于图像识别** 的UI自动化测试框架，适用于游戏和App，支持平台有Windows、Android和iOS：

[Airtest源码](https://github.com/AirtestProject/Airtest)

[Airtest API文档](https://airtest.readthedocs.io/zh_CN/latest/)

#### **2）Poco框架[¶](https://airtest.doc.io.netease.com/" \l "2poco" \o "Permanent link)**

Poco 是一款 **基于UI控件识别** 的自动化测试框架，目前支持Android原生、iOS原生、Unity3D、cocos2dx、UE4和Egret等平台，也可以在其他引擎中自行接入poco-sdk来使用。

* [Poco 源码](https://github.com/AirtestProject/Poco)
* [Poco API文档](http://poco.readthedocs.io/zh_CN/latest/)

#### **3）AirtestIDE[¶](https://airtest.doc.io.netease.com/" \l "3airtestide" \o "Permanent link)**

AirtestIDE 是一款跨平台的 **UI自动化测试编辑器** ，内置了Airtest和Poco的相关插件功能，能够使用它快速简单地编写 Airtest 和 Poco 代码。

* [官方下载网址](https://airtest.netease.com/home/)
* [AirtestIDE使用文档](https://airtest.doc.io.netease.com/)

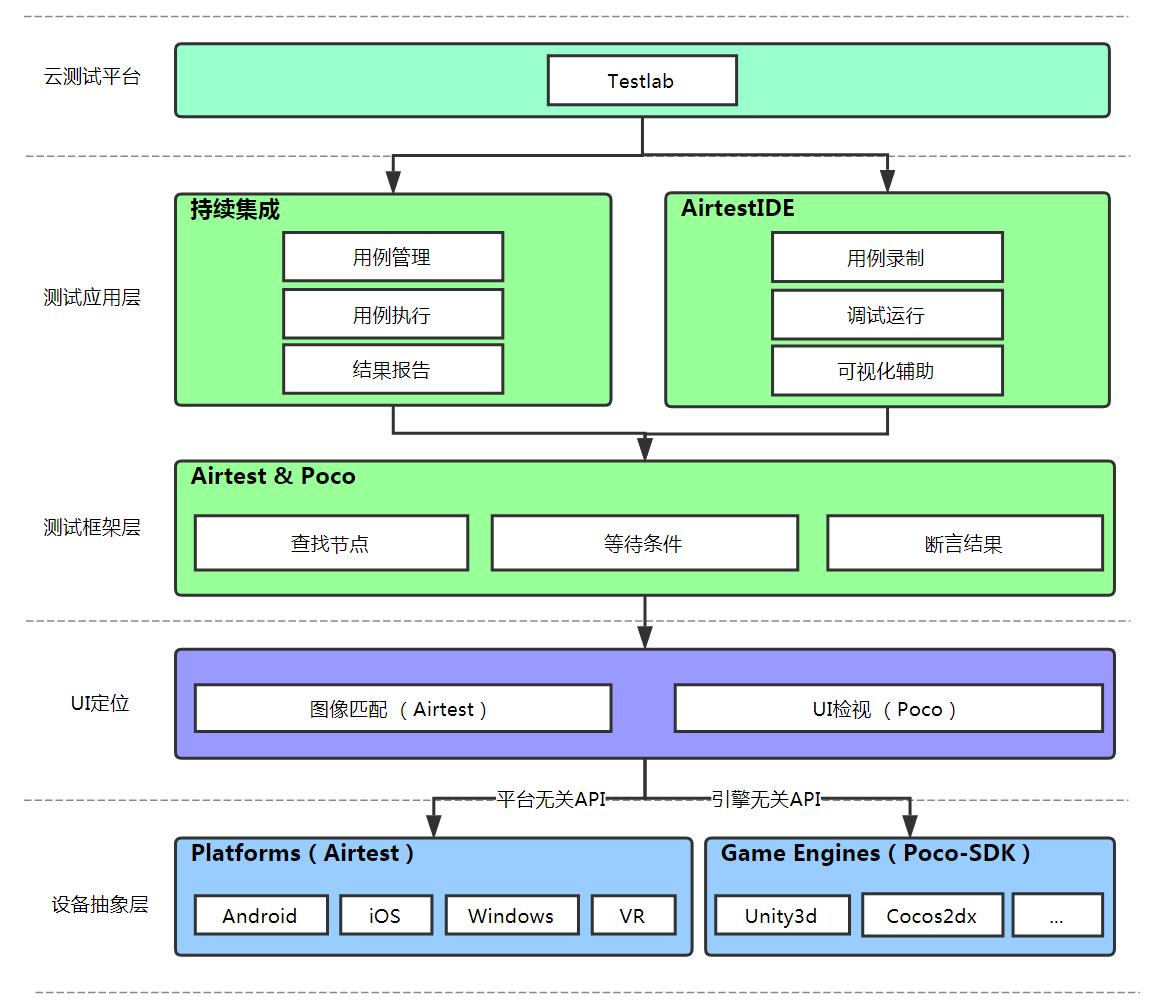
#### **4）手机集群解决方案-DeviceFarm[¶](https://airtest.doc.io.netease.com/" \l "4-devicefarm" \o "Permanent link)**

DeviceFarm 是网易推出的自动化测试集群解决方案，它是软硬件一体化方案，包含设备集群建设、集群设备批量维护管理、监控报警和云端真机等功能，可以为您在企业内部搭建稳定高效的设备管理平台, 提升设备利用率，赋能自动化测试流程搭建：

* [DeviceFarm集群解决方案](https://airtest.doc.io.netease.com/commercial/commercial/" \l "1_2)

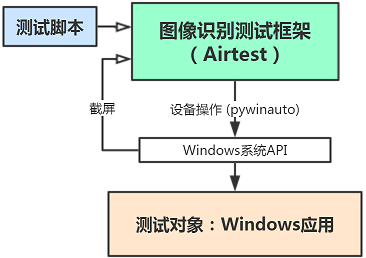
#### **5）Airlab云测试平台[¶](https://airtest.doc.io.netease.com/" \l "5airlab" \o "Permanent link)**

Airlab云测试平台支持用例管理、脚本管理、任务预约、任务调度、云端报告等功能，可以快速构建企业自动化测试全流程，支持ios和安卓的回归测试及兼容测试。



## Airtest测试windos应用程序【基本原理】

AirtestIDE对普通Windows应用程序的测试支持，主要依靠图像识别框架（Airtest）进行位置定位，使用 [pywinauto](https://pywinauto.readthedocs.io/en/latest/index.html)的操作接口进行模拟操作。



# Airtest的图像脚本介绍

其实它本质上还是一条Python脚本，只不过Airtest把图像封装起来了，并且可以直接显示图像，更方便同学们查看：

## Airtest的图像识别算法

#### **Airtest的图像识别算法介绍**

在Airtest1.2.0版本之后，图像识别过程中，会使用这MultiScaleTemplateMatchingPreSURFMatchingMultiScaleTemplateMatchingPre、TemplateMatching、SURFMatching、BRISKMatchingBRISKMatching这4个算法依次进行图像识别，找到结果将停止识别，未找到结果将会一直按照这个算法的识别顺序一直循环识别直到超时。

这些算法有的是模板匹配算法，也有的是特征点匹配算法，关于Airtest使用的图像识别算法的详细介绍，我们可以参考下面2篇推文：

这些算法有的是模板匹配算法，也有的是特征点匹配算法，模板匹配算法（也就是上文的TemplateMatching），特征点匹配（包含了上文的SURFMatching和BRISKMatching）

##### ****模板匹配****

无法跨分辨率识别

一定有相对最佳的匹配结果

方法名："tpl"

##### ****特征点匹配****

跨分辨率识别

不一定有匹配结果

方法名列表：["kaze", "brisk", "akaze", "orb", "sift", "surf", "brief"]

在这里我们还需要解释一下：  
1.无法跨分辨率识别，意思是一旦换一台不同分辨率的设备就可能识别失败；  
2.一定有相对最佳的匹配结果，即是不管怎么说也会给你找一个结果出来，虽然可能差别很大。比如上文例子中，我们使用了当前设备不存在的截图让程序去查找，屏幕上根本没有结果，TemplateMatching算法也找出了一个可信度为0.4的结果。

##### ****算法小结****

模板匹配算法的优点是速度很快，如果分辨率不会改变的话，不妨选择它作为首选算法

在分辨率可能会发生改变的情况下，我们会用特征点匹配的办法来找图，这样跨平台的适应能力会更高，脚本容易适配不同型号的手机