如何选择Halcon算法库应用软件集成开发环境

大多数HALCON解决方案都必须嵌入到PC端的应用程序中（例如，提供图形用户界面），然后大多数机器视觉应用程序使用C ++或C＃实现。这是我们的视角，对您的下一个HALCON项目应使用哪种编程语言提供参考

## 不易的选择

选择编程语言是并不容易，各有优劣和学习成本。很难在单个博客文章中对其进行详细说明。我们将尽力强调C ++和C＃之间的一些差异-评估和决定必须由公司或其HALCON开发人员来执行。

有些文章说编程语言不重要，从原理上看，编程语言都类似，对一门编程语言掌握透彻后，其他编程语言可以触类旁通。从另外一个角度，不同的编程语言应用领域不同，编程语言相关的库，框架，各种工具（编译器，调试工具，测试工具等），易维护程度，招聘相关开发人员难易程度及质量等等，导致在集成Halcon解决方案时，不同编程语言之间有些许差异

## 开发经验

如果您或您的HALCON视觉应用程序开发人员仅精通其中一种语言，则对于选择该编程语言来说，这是非常重要的论据。您需要充分的理由（可能是以下原因之一）才能更改为另一种语言。

## 对HALCON的支持

HALCON支持通过HALCON / C ++和HALCON / .NET接口集成到C ++和C＃的应用程序中。在大多数情况下，HALCON脚本的性能将相似。

## 垃圾收集器/确定性销毁（RAII）

C++具有RAII模式和具有超出范围的对象的确定行为 。因此，我们可以从C ++语言中保证，每个分配的HImage（例如在图像采集循环中）之后都会自动释放，并且不会发生内存泄漏。在C＃中，通常很难实现这种可靠性，并且经常需要手动处理和需要额外的处理出错的工作，例如，手动调用内存清理函数myImage.Destroy()或GCHandle.Free()。但是，如果出现异常或在循环或函数过程中（continue，return）起作用， C#中手动编写的内存清理函数调用时很容易被跳过。此外，通常不可能确保垃圾回收器不会在应用程序的性能特别关键的部分运行。

## 托管与本地

本机代码是由C ++编译器生成的目标CPU的机器代码。它将进行高度优化，但是在编译过程中此优化需要一些时间。因此，C ++程序往往会增加构建时间，但最终应用程序的启动时间却很快，尤其是处理速度更快。

相反，C＃程序仅编译为中间字节码。这可以在构建过程中相对快速地完成，但是代码的优化不会像使用良好的C ++编译器所能实现的那样优化。最终在运行时生成的机器代码将取决于目标系统，因此无法在开发人员的机器上进行分析。

作为现代C++编译器可以实现的 优化功能 一个例子，使用下面的简单C++片段计算整数中的已置位的位数：

**int** **count\_bits\_1**(**int** a) {  
    **int** count = 0;  
    **while** (a) {  
        count++;  
        a &amp;= (a-1);  
    }  
    **return** count;  
}

此代码编译为

count\_bits\_1(**int**): # @count\_bits\_1(**int**)  
popcnt eax, edi  
ret

使用Gcc编译器，很容易看到生成的机器代码/汇编代码，例如使用编译器资源管理器。C++编译器的优化步骤了解底层算法并将其转换为英特尔x86/x64 CPU的一个优化的POCPNT指令。

## Windows

在开发C#代码时，他将Windows作为唯一的操作系统，尽管通过使用Mono项目，其他目标系统也可能以有限的形式运行（例如没有WPF）。C++代码可以非常便携，然后在许多不同的系统上运行，例如使用QT跨平台应用程序框架和WIDGET工具包。

## HALCON扩展包，图像采集接口，数字I / O接口

如果HALCON / HDevelop中缺少某些功能（某些硬件接口，客户特定的图像采集等），则可以通过自定义扩展包，图像获取接口或数字I / O接口来提供。这些只能用C或C ++语言编写。

## 外部库/开源库

许多外部库使用 C或C ++ 编程语言：可以为C / C ++库编写C＃封装接口，但是为C＃库编写C ++封装接口并不常见。这些用于C ++库的C＃接口与C++最新版本有些延后：例如，OpenCV 4.5.2于2021年4月发布，但直到2021年7月，相应的.NET-Wrapper Emgu.CV仅可用于OpenCV 4.5.2。

封装接口通常使用起来更复杂，因为它们必须将C ++结构映射到C＃上，但并不总是100％适合：例如，HALCON库是用C开发的，并且具有C＃封装器（HALCON / .NET）。在C＃中，访问图像矩阵很复杂：

GCHandle gch = GCHandle.Alloc(color\_image);  
image.GetImagePointer3(**out** pr, **out** pg, **out** pb, **out** type, **out** width, **out** height);  
**byte**[] red\_managed= **new** **byte**[width\*height];  
Marshal.Copy(pr, red\_managed, 0, width\*height);  
*// ...*  
gch.Free();

大多数Windows内部库本身都是用C / C ++开发的，因此.NET框架只是这些库的封装。

## 硬件驱动

某些开发仅在C或C ++中可行，例如硬件驱动程序。如果以后需要进行这类工作并且公司中仅熟悉C＃知识，则必须熟悉必要的C ++知识。

## 调试

使用C＃，有时在Visual Studio中进行调试会更容易。

### 示例：HALCON异常

HImage color\_image = color\_image\_small.ZoomImageSize(4000, 3000, "WRONG");

在C＃中抛出的异常信息更有有意义

HalconDotNet.HOperatorException: 'HALCON error #3147: Wrong interpolation mode in operator zoom\_image\_size'

但是C ++异常是没有意义的：

Unhandled exception at 0x00007FFB288F3FB8 **in** MyApp.exe: Microsoft C++ exception: HalconCpp::HOperatorException at memory location 0x0000007469EFF718.

### 示例：查看HALCON元组

c#:

**var** t = **new** HTuple();  
t[0] = 42;  
t[1] = "string";  
Watch window: Name: t, Value: {[42, "string"]}

在C ++中，就不能直接查看

HTuple t = HTuple(42).Append("string");  
Watch window: Name: t, Value: &lt;Information **not** available...

另一方面，两种编程语言都可以使用HALCON变量检查扩展。

## 低级代码

使用C ++开发时，低级代码通常运行得更快。

### 示例：颜色转换

有时，必须在GUI框架中使用HALCON彩色图像。GUI框架通常使用交错的颜色格式（RGB RGB RGB…），而HALCON则使用每个通道的灰度值块（RRR…GGG…BBB…）。某些彩色图像到交错格式的转换在C ++中花费25毫秒，而在C＃中花费60毫秒。另一方面，通常可以忽略从C ++调用的HALCON代码或从C＃调用的HALCON代码之间的性能差异。

## 集成开发环境

有许多用于C ++的集成开发环境（IDE），但对于C＃IDE的选择却很少。