# **基于HALCON的机器视觉开发，C++或C#如何选择？**

大多数HALCON解决方案必须嵌入到主应用程序中(例如，提供图形用户界面)，然后大多数机器视觉应用程序都是用c++或c#实现的。下面是我们关于下一个HALCON项目应该使用哪种主机语言的观点，供读者参考。



****语言选择****

编程语言的选择对于公司来说是一个战略和长期的决策。这很难在一篇文章中详细解释。我们将尽最大努力强调C++和C语言之间的差异。评估和最终决策必须由一个公司或其HalCon开发人员来执行。

如果您或您的HALCON视觉应用程序开发人员只熟悉其中一种语言，那么这种语言就是一个非常重要的论据。您可能需要有很充足的理由才能更改为其他语言。

****支持HALCON****

HALCON支持使用HALCON / C ++和HALCON / .NET接口集成到C ++和C＃主应用程序中。在大多数情况下，HALCON脚本的性能是类似。

****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性行为。因此，我们可以从C++语言中保证每一个分配的HIMAN，例如在图像采集循环中，自动释放之后，不会有内存泄漏。在C#中，通常很难实现这种可靠性，并且通常需要手动和容易出错的工作，例如手动调用my..Destroy()或GCHandle.Free()。但是，如果出现异常、早期循环或函数退出（继续、返回），C#中的这些清理调用很容易被忽略。此外，通常无法确保垃圾收集器不在应用程序的性能关键部分中运行。

****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，但这种优化在编译期间需要一些时间。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会快，尤其是快速处理性能。

相反，C＃程序仅编译为中间字节码。这可以在构建期间相对快速地完成，但代码将不会像使用优秀的C ++编译器那样进行优化。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 ++编译器可以实现的示例，以下使用简单的C ++代码来计算整数中的设置位数：

int count\_bits\_1(int a) {

int count = 0;

while (a) {

count++;

a &= (a-1);

}

return count;

}

This code compiles to

count\_bits\_1(int): # @count\_bits\_1(int)

popcnt eax, edi

ret

使用clang 6.0.0编译器，很容易看到生成的机器代码/汇编代码，例如使用Compiler Explorer。C ++编译器的优化步骤了解底层算法并将其转换为针对Intel x86 / x64 CPU的单个优化popcnt指令。

****视窗****

在开发C＃代码时，其中一个目标是Windows作为唯一的操作系统，尽管其他目标系统可以通过使用Mono项目以有限的形式（例如，没有WPF）实现。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（某些硬件接口，客户特定图像采集……）中缺少某些功能，则可以通过自定义扩展包，图像采集接口或数字I / O接口提供。这些只能用C或C ++语言编写。

****外部库/开源库****

许多外部库使用最低的公分母作为编程语言，即C或C ++：可以为C / C ++库编写C＃包装器，但为C＃库编写C ++包装器并不常见。这些用于C ++库的C＃包装器有时已过时：例如，OpenCV 3.4于2017年12月发布，但直到2018年2月，相应的.NET-Wrapper Emgu.CV仅适用于OpenCV 3.3。

包装器通常很难使用，因为它们必须将C ++结构映射到C＃，这并不总是100％适合：例如，HALCON库是用C开发的，并且有一个C＃包装器（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，并且.NETFramework仅仅是这些库的包装器。

****硬件驱动程序****

有些开发只可能在C或c++中实现，例如硬件驱动程序。如果以后需要做这些工作，并且公司中只存在c#知识，那么必须重新开发c++知识。

****调试****

在Visual Studio中调试c#有时更容易。

示例：HALCON异常

代码

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

throws in C# a meaningful

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

but the C++ exception is meaningless:

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

****示例：观看HALCON Tuples****

C＃：

var t = new HTuple（）;

t [0] = 42;

t [1] =“字符串”;

监视窗口：名称：t，值：{[42，“string”]}

在C ++中，这种内省不可能是开箱即用的：

HTuple t = HTuple（42）.Append（“string”）;

观察窗口：名称：t，值：<信息不可用…

另一方面，HALCON Variable Inspect扩展可用于两种编程语言。

****低级代码****

在C ++中开发时，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框架通常使用交错颜色格式（RGB RGB RGB …），而HALCON使用每个通道的灰度值块（RRR … GGG … BBB …）。对于某些彩色图像，转换为交错格式在C ++中需要25毫秒，在C＃中需要60毫秒。另一方面，从C＃调用的HALCON代码或从C＃调用的HALCON代码之间的性能差异通常可以忽略不计。

****集成开发环境****

C ++有许多集成开发环境（IDE），但C＃IDE的选择较少。