



Mech-DLK SDK用户手册

v2.0.1

目录

1. 欢迎	1
2. 安装指南	2
3. 快速入门	3
4. 例程使用指南	7
4.1. C# 语言 (Windows)	7
4.1.1. 运行 Basic 例程	9
4.1.2. 运行 Advanced 例程 (HALCON)	11
4.1.3. 运行 Advanced 例程 (OpenCV)	13
4.2. C 语言 (Windows)	15
5. API 参考手册	22
6. 常见问题	23

1. 欢迎

欢迎使用 Mech-DLK SDK 用户手册。

概述

Mech-DLK SDK 是专门配合 Mech-DLK 使用的二次开发软件包，主要用来帮助用户在已有的软件体系内轻松构建深度学习推理部分。用户可以在不依赖 Mech-Vision 的情况下，快速部署深度学习模型，灵活地集成深度学习功能到自己的应用中。目前支持 C 及 C# 语言开发。

你可以应用 Mech-DLK SDK 实现 Mech-DLK（2.4.2 及以上版本）导出模型的推理。

更新说明

Mech-DLK SDK 2.0.1

新增功能

- 新增了 C# API，支持使用 C# 语言进行二次开发。查看 [C# API 参考手册](#)。
- 提供了 C# 语言例程，实现了与 OpenCV 和 HALCON 的协同开发。查看 [例程简介与使用前提](#)。

调整配置方式

- 修改了 Mech-DLK SDK 环境变量添加方式，具体内容请参考各语言例程使用指南中的“添加环境变量”部分。

问题修复

- 修复了使用缺陷分割模型进行推理时的结果解析问题。

目录

本手册由以下几个部分构成，可根据需求查阅：

序号	章节	内容
1	安装指南	查看系统要求、获取 Mech-DLK SDK 及其依赖的第三方库和资源文件。
2	快速入门	了解如何使用 Mech-DLK SDK 进行缺陷分割模型的推理。
3	例程使用指南	了解 C 及 C# 语言例程类型及运行前提，构建及运行例程。
4	API 参考手册	查看 C 及 C# API 和结构体。
5	常见问题	查看常见问题。

2. 安装指南

系统要求

建议应用 Mech-DLK SDK 进行模型推理的设备满足以下软硬件要求。

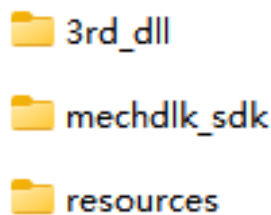
加密狗授权版本	Pro-Run	Pro-Train
操作系统	Windows 10 及以上	
CPU	Intel® Core™ i7-6700 及以上	
内存	8 GB 及以上	16 GB 及以上
显卡	GeForce GTX 1660 及以上	GeForce RTX 3060 及以上
显卡驱动	驱动版本 472.50 及以上	



Pro-Run 版本具有 Mech-DLK SDK 部署、标注、运行模式功能；Pro-Train 版本具有级联、标注、训练、验证以及 Mech-DLK SDK 部署功能。

获取 Mech-DLK SDK

1. 在本地创建一个项目文件夹，例如“dlk_sdk”。
2. 将 [Mech-DLK SDK](#) 从 GitHub 克隆到该项目文件夹下。
3. 从[下载中心](#)下载 Mech-DLK SDK 依赖的第三方库（3rd_dll.zip）和资源文件（resources.zip）到该项目文件夹下。
4. 解压第三方库压缩包和资源文件压缩包。此时，该项目文件夹应包含以下内容：




请勿随意更改文件夹里的文件，并记住文件的位置，供后续使用。

3. 快速入门

本章介绍如何应用 Mech-DLK SDK 实现 Mech-DLK 训练的缺陷分割模型的推理。

准备工作

- 安装 Mech-DLK SDK。
- 将梅卡曼德的加密狗插入电脑。
- 安装 CodeMeter，并确认 CodeMeter 正在运行，即 CodeMeter 的图标  显示在系统托盘中。



如已安装 Mech-DLK，则不需要重复安装 CodeMeter，只需确认 CodeMeter 正在运行即可。

推理流程



函数说明

下面以 Mech-DLK 导出的缺陷分割模型为例，介绍应用 Mech-DLK SDK 实现模型推理时使用的函数。

创建输入图像

调用以下函数创建输入图像。

C#

```
MMindImage image = new MMindImage();  
image.CreateFromPath("path/to/image.png");  
List<MMindImage> images = new List<MMindImage> { image };
```

C

```
MMindImage input;  
createImage("path/to/image.png", &input);
```

创建推理引擎

调用以下函数创建推理引擎。

C#

```
InferEngine inferEngine = new InferEngine();
inferEngine.Create("path/to/xxx.dlpack", BackendType.GpuDefault, 0);
```



- 当部署电脑上配置有英伟达的独立显卡时，你可以将推理后端（即第二个参数）设置为 GpuDefault 或 GpuOptimization。
 - 当此参数设置为 GpuOptimization 时，需要等待 1~5 min 的模型优化时间。
- 当部署电脑上未配置英伟达的独立显卡时，推理后端只能设置为 CPU。
- 此函数中，第三个参数 0 为英伟达显卡 ID，如果只有一张显卡，填写 0 即可。当推理后端设置为 CPU 时，此参数无效。

C

```
Engine engine;
createPackInferEngine(&engine, "path/to/xxx.dlpack", GpuDefault, 0);
```



- 当部署电脑上配置有英伟达的独立显卡时，你可以将推理后端（即第三个参数）设置为 GpuDefault 或 GpuOptimization。
 - 当此参数设置为 GpuOptimization 时，需要等待 1~5 min 的模型优化时间。
- 当部署电脑上未配置英伟达的独立显卡时，推理后端只能设置为 CPU。
- 此函数中，第四个参数 0 为英伟达显卡 ID，如果只有一张显卡，填写 0 即可。当推理后端设置为 CPU 时，此参数无效。

深度学习引擎推理

调用以下函数进行深度学习引擎推理。

C#

```
inferEngine.Infer(images);
```

C

```
infer(&engine, &input, 1);
```



此函数中，参数 1 表示推理的图片数量，必须与 input 中的图片数量相同。

获取缺陷分割结果

调用以下函数得到缺陷分割模型的结果。

C#

```
List<Result> results;
inferEngine.GetResults(out results);
```

C

```
DefectAndEdgeResult* defectAndEdgeResult = NULL;
unsigned int resultNum = 0;
getDefectSegmentationResult(&engine, 0, &defectAndEdgeResult, &resultNum);
```



此函数中，第二个参数 **0** 表示深度学习模型推理包中的模型索引。

- 如果是单模型推理包，该参数只能设置为 0；
- 如果为级联模型推理包，该参数需要根据模型推理包中算法的顺序进行设置。

结果可视化

调用以下函数可视化模型推理结果。

C#

```
inferEngine.ResultVisualization(images);
image.Show("result");
```

C

```
resultVisualization(&engine, &input, 1);
showImage(&input, "result");
```



此函数中，参数 **1** 表示推理的图片数量，必须与 **input** 中的图片数量相同。

释放内存

释放内存，防止内存泄漏。

C#

```
inferEngine.Release();
```

C

```
releaseDefectSegmentationResult(&defectAndEdgeResult, resultNum);
releaseImage(&input);
```

```
releasePackInferEngine(&engine);
```


4. 例程使用指南

本章介绍 C#、C 例程的使用指南。

C# 例程

查看以下内容，了解 C# 例程的**使用前提**和**例程运行方式**。

[了解使用前提](#)

[运行 Basic 例程](#)

[运行 Advanced 例程 \(HALCON\)](#)

[运行 Advanced 例程 \(OpenCV\)](#)

C 例程

查看以下内容，了解 C 例程的**使用前提**和**例程运行方式**。

[了解使用前提](#)

[运行 Basic 和 Advanced 例程](#)

4.1. C# 语言 (Windows)

例程简介

C# 语言例程分为 2 类：**Basic** 和 **Advanced**。

- **Basic** 例程：使用 Mech-DLK 导出的模型进行单图推理或多图同时推理、获取并可视化结果。
 - [ImageInfer](#)
单图推理示例（支持单级模型与级联模型）
 - [MultiImageInfer](#)
多图推理示例（支持单级模型与级联模型）
- **Advanced** 例程：Mech-DLK SDK 与 HALCON/OpenCV 协同开发。
 - [ImageInferWithHalcon](#)
Mech-DLK SDK 与 HALCON 配合使用示例（需本地安装 HALCON）

- [ImageInferWithOpenCV](#)

Mech-DLK SDK 与 OpenCV 配合使用示例（需本地安装 OpenCV）

使用前提

使用 Mech-DLK SDK 的 C# 语言例程，需满足以下使用前提：

- [安装必需软件](#)。
- [添加相关的环境变量](#)。

安装必需软件

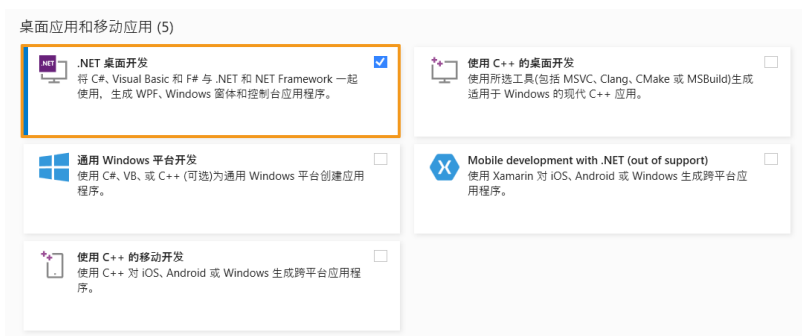
使用 Mech-DLK SDK 的 C# 语言例程，必须安装 Mech-DLK SDK 及 Visual Studio。

安装 Mech-DLK SDK

请根据 [安装指南](#) 获取最新版本的 Mech-DLK SDK 及其依赖的第三方库和资源文件。

安装 Visual Studio（2017 或以上版本）

1. 下载 [Visual Studio 安装包](#)。
2. 安装时，请勾选“桌面应用和移动应用”分类中的 **.NET 桌面开发** 工作负荷，再单击右下角的[安装]。



添加环境变量

请根据以下步骤添加相关的环境变量。

1. 右键单击桌面上的“此电脑”，选择“属性”。
2. 选择“高级系统设置”，在弹出的“系统属性”对话框的“高级”选项卡上，单击[环境变量]，进入“环境变量”界面。
3. 在“系统变量”下，单击[新建]，在“新建系统变量”对话框的“变量名”框中输入 **MECHDL_DIR**，“变量值”框中输入 **xxx/mechdlk_sdk**，然后单击[确定]。
4. 在“系统变量”框中，滚动到“Path”并双击它进入到“编辑环境变量”页面。

5. 单击右上角 [新建], 添加 %MECHDL_DIR%, 再单击右下角 [确定]。



如果“编辑环境变量”框中有 %MMIND_DLK%, 选中后单击右侧的[删除]按钮移除此变量。



4.1.1. 运行 Basic 例程

根据 C# 例程的 [使用前提](#) 部分完成相关操作后, 你可以按以下步骤构建和运行例程。

构建例程

1. 在 `xxx\mechdlk_sdk\samples\csharp` 目录下, 双击 `MechDLShapeSamples.sln` 使用 Visual Studio 打开解决方案。

- 在菜单栏上选择 生成 > 生成解决方案。此时，会生成例程对应的可执行文件（.exe），保存在 **bin** 文件夹中，该文件夹位于 `xxx\mechdlk_sdk\samples\csharp` 目录下。



- 将项目文件夹中的 **resources** 文件夹拷贝至 `xxx\mechdlk_sdk\samples\csharp\bin` 目录下。
- 将项目文件夹中的 **3rd_dll** 文件夹内的全部文件拷贝至 `xxx\mechdlk_sdk\samples\csharp\bin` 目录下。
- 将项目文件夹中 **mechdlk_sdk\dll** 文件夹内的全部文件拷贝至 `xxx\mechdlk_sdk\samples\csharp\bin` 目录下。



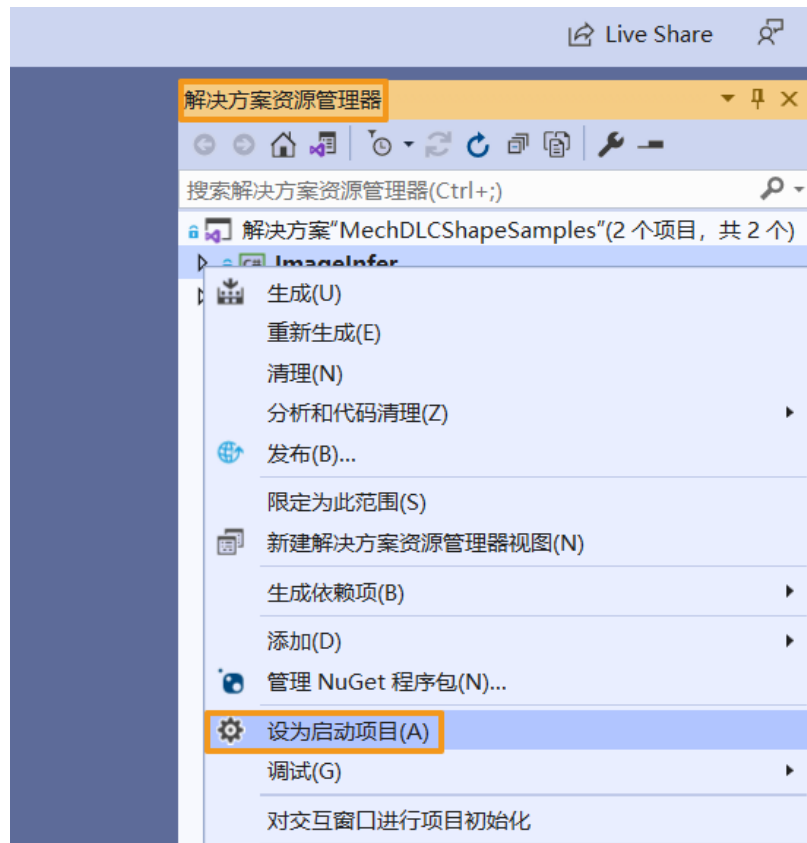
构建 Mech-DLK SDK 中任一 C# 例程，第 3、4、5 步只需操作一次，无须重复拷贝。

运行例程

你可以在 Visual Studio 中直接运行例程，也可以双击运行例程的可执行文件。

在 Visual Studio 中运行例程

- 在“解决方案资源管理器”窗口中右键单击想要运行的例程，并选择“设为启动项目”。



2. 单击工具栏中的[启动]运行该例程。

运行例程可执行文件

进入 **bin** 文件夹 (`xxx\mechdlk_sdk\samples\csharp\bin`)，双击运行与例程同名的可执行文件 (.exe)，即可得到运行结果。

4.1.2. 运行 Advanced 例程 (HALCON)

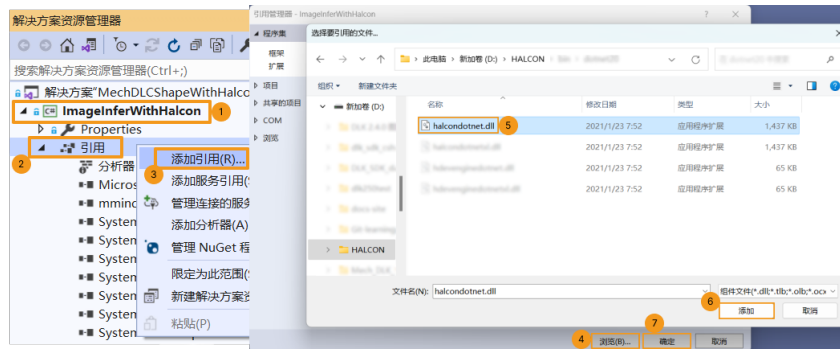
根据 C# 例程的 [使用前提](#) 部分完成相关操作后，你可以按以下步骤构建和运行例程。

安装 HALCON

1. 下载并安装 C# 版本的 HALCON。请记录 HALCON 的存放路径。
2. 在 `xxx\mechdlk_sdk\samples\csharp` 目录下，双击 `MechDLCSHAPEWithHalconSamples.sln` 使用 Visual Studio 打开解决方案。
3. 在“解决方案资源管理器”窗口中选择 `ImageInferWithHalcon` > 引用，右键单击“引用”，选择“添加引用”。
4. 在弹出的“引用管理器”对话框中，单击右下角[浏览]。
5. 找到 HALCON 的存放路径，在 `xxx\bin\dotnet35` 目录下找到 `halcondotnet.dll` 文件，并单击[添加]。
6. 在“引用管理器”对话框中，单击右下角的[确定]完成添加。



提供的例程已使用 HALCON 20.11.1.2 完成测试，可正常运行。不同版本软件库中相应的 DLL 文件路径可能不一致，请根据实际情况完成上述操作。



构建例程

1. 在 Visual Studio 菜单栏上选择 生成 > 生成解决方案。此时，会生成例程对应的可执行文件 (.exe)，保存在 **bin** 文件夹中，该文件夹位于 `xxx\mechdlk_sdk\samples\csharp` 目录下。



2. 将项目文件夹中的 **resources** 文件夹拷贝至 `xxx\mechdlk_sdk\samples\csharp\bin` 目录下。
3. 将项目文件夹中的 **3rd_dll** 文件夹内的全部文件拷贝至 `xxx\mechdlk_sdk\samples\csharp\bin` 目录下。
4. 将项目文件夹中 **mechdlk_sdk\dll** 文件夹内的全部文件拷贝至 `xxx\mechdlk_sdk\samples\csharp\bin` 目录下。



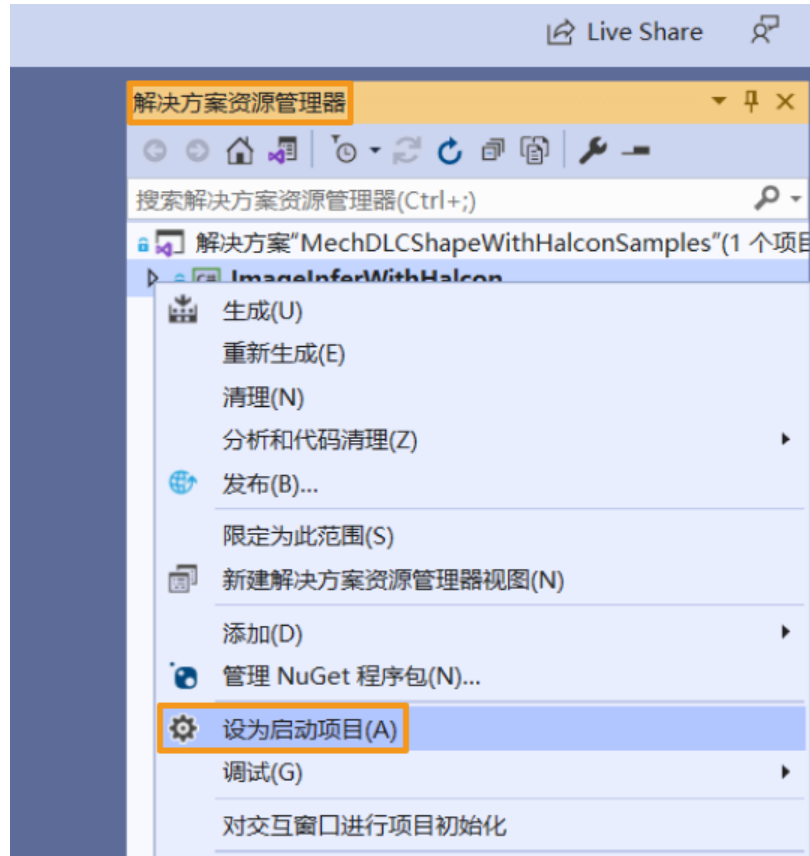
构建 Mech-DLK SDK 中任一 C# 例程，第 2、3、4 步只需操作一次，无需重复拷贝。

运行例程

你可以在 Visual Studio 中直接运行例程，也可以双击运行例程的可执行文件。

在 Visual Studio 中运行例程

1. 在“解决方案资源管理器”窗口中右键单击想要运行的例程，并选择“设为启动项目”。



2. 单击工具栏中的[启动]运行该例程。

运行例程可执行文件

进入 **bin** 文件夹（`xxx\mechdlk_sdk\samples\csharp\bin`），双击运行与例程同名的可执行文件（.exe），即可得到运行结果。

4.1.3. 运行 Advanced 例程（OpenCV）

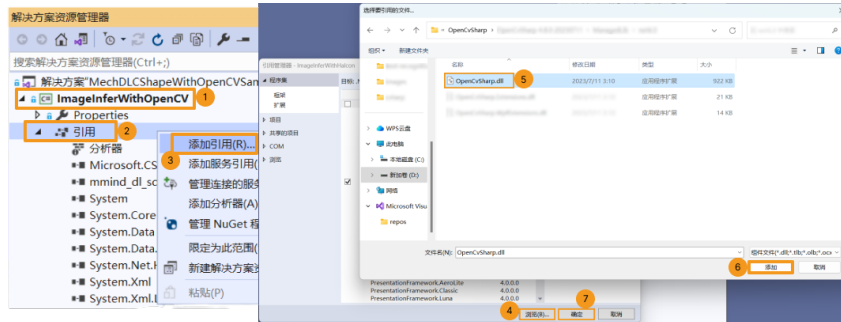
根据 C# 例程的 [使用前提](#) 部分完成相关操作后，你可以按以下步骤构建和运行例程。

安装 OpenCV

1. 下载并安装 [C# 版本的OpenCV](#)，即 OpenCvSharp。请记录 OpenCvSharp 的存放路径。
2. 在 `xxx\mechdlk_sdk\samples\csharp` 目录下，双击 `MechDLCSapeWithOpenCVSamples.sln` 使用 Visual Studio 打开解决方案。
3. 在“解决方案资源管理器”窗口中选择 `ImageInferWithOpenCV` > 引用。右键单击“引用”，选择“添加引用”。
4. 在弹出的“引用管理器”对话框中，单击右下角[浏览]。
5. 找到 OpenCV 的存放路径，在 `xxx\ManagedLib\net461` 目录下找到 `OpenCvSharp.dll` 文件，并单击[添加]。
6. 在“引用管理器”对话框中，单击右下角的[确定]完成添加。



提供的例程已使用 OpenCvSharp 4.5.3 完成测试，可正常运行。不同版本软件库中相应的 DLL 文件路径可能不一致，请根据实际情况完成上述操作。



构建例程

1. 在 Visual Studio 菜单栏上选择 生成 > 生成解决方案。此时，会生成例程对应的可执行文件 (.exe)，保存在 **bin** 文件夹中，该文件夹位于 `xxx\mechdlk_sdk\samples\csharp` 目录下。



2. 将项目文件夹中的 **resources** 文件夹拷贝至 `xxx\mechdlk_sdk\samples\csharp\bin` 目录下。
3. 将项目文件夹中的 **3rd_dll** 文件夹内的全部文件拷贝至 `xxx\mechdlk_sdk\samples\csharp\bin` 目录下。
4. 将项目文件夹中 **mechdlk_sdk\dll** 文件夹内的全部文件拷贝至 `xxx\mechdlk_sdk\samples\csharp\bin` 目录下。
5. 打开 OpenCvSharp 的存放路径，在 **NativeLib\win\x64** 中找到并拷贝 **OpenCvSharpExtern.dll** 文件至 `xxx\mechdlk_sdk\samples\csharp\bin` 目录下。



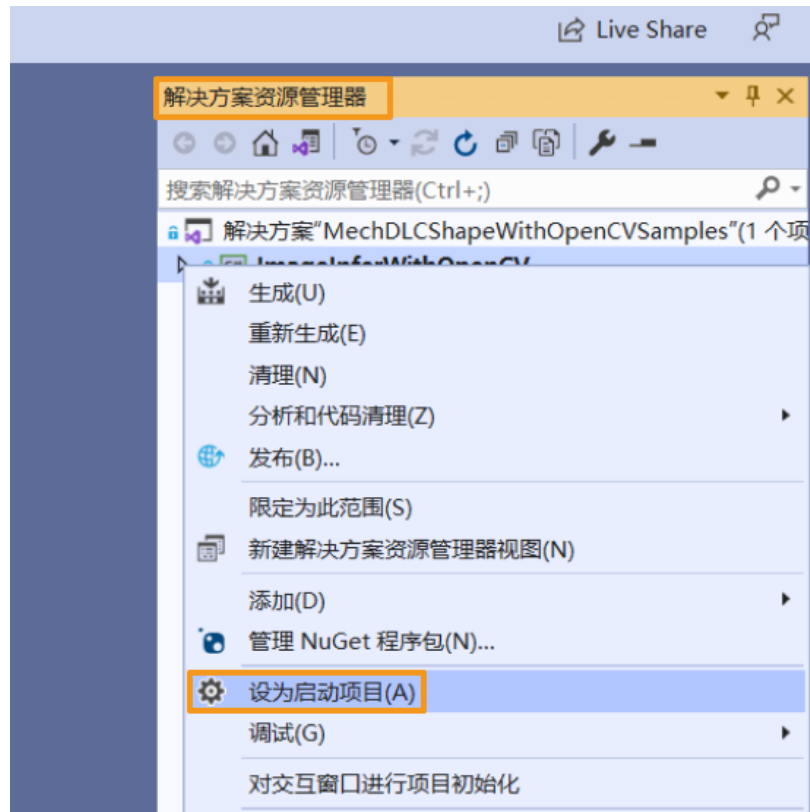
构建 Mech-DLK SDK 中任一 C# 例程，第 2、3、4 步只需操作一次，无需重复拷贝。

运行例程

你可以在 Visual Studio 中直接运行例程，也可以双击运行例程的可执行文件。

在 Visual Studio 中运行例程

1. 在“解决方案资源管理器”窗口中右键单击想要运行的例程，并选择“设为启动项目”。



2. 单击工具栏中的[启动]运行该例程。

运行例程可执行文件

进入 **bin** 文件夹 (`xxx\mechdlk_sdk\samples\csharp\bin`)，双击运行与例程同名的可执行文件 (.exe)，即可得到运行结果。

4.2. C 语言 (Windows)

本章介绍如何在 Windows 系统上用 CMake 和 Visual Studio 配置 Mech-DLK SDK 中的 C 语言例程。

例程简介

例程分为 2 类：**Basic** 和 **Advanced**。

- **Basic** 例程：使用 Mech-DLK 导出的各类单级模型推理单张图片、获取并可可视化结果；
- **Advanced** 例程：级联模型推理、多图同时推理。

Basic

- [Classification](#): 图像分类模型推理示例;
- [DefectSegmentation](#): 缺陷分割模型推理示例;
- [FastPositioning](#): 快速定位模型推理示例;
- [InstanceSegmentation](#): 实例分割模型推理示例;
- [ObjectDetection](#): 目标检测模型推理示例。

Advanced

- [CascadeModel](#): 级联模型推理示例;
- [FolderImagesInfer](#): 依次推理文件夹中图像示例;
- [MultiImageInfer](#): 多图同时推理示例。

使用前提

使用 Mech-DLK SDK 中的 C 语言例程，需先满足以下使用前提：

- [安装必需软件](#)。
- [添加相关的环境变量](#)。

安装必需软件

使用 Mech-DLK SDK 的 C 语言例程，必须安装 Mech-DLK SDK、CMake 及 Visual Studio。

安装 Mech-DLK SDK

请根据 [安装指南](#) 获取最新版本的 Mech-DLK SDK 及其依赖的第三方库和资源文件。

安装 CMake (3.2 或以上版本)

1. 下载 [CMake](#): 选择 **Windows x64 Installer** 右侧的安装包。

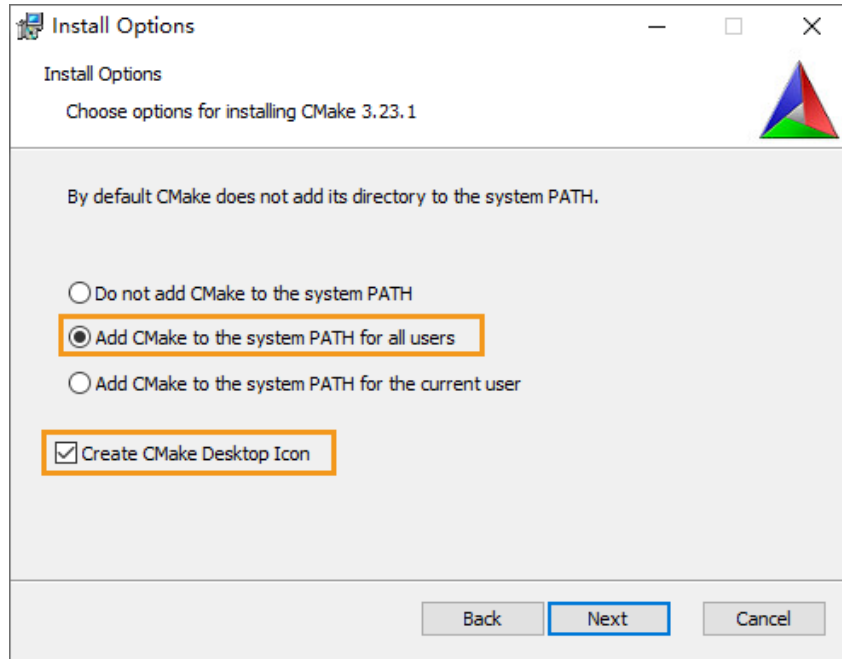
Source distributions:

Platform	Files
Unix/Linux Source (has \n line feeds)	cmake-3.23.1.tar.gz
Windows Source (has \r\n line feeds)	cmake-3.23.1.zip

Binary distributions:

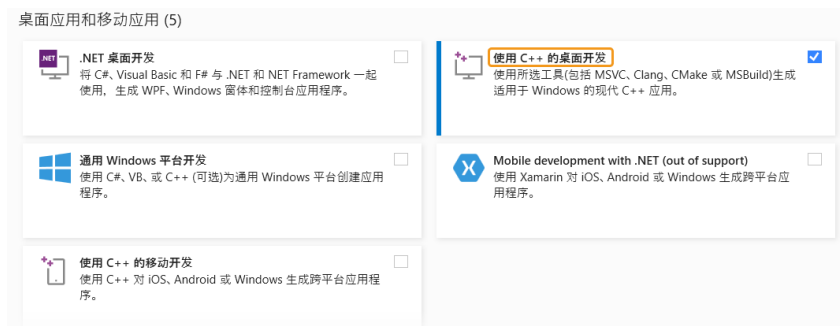
Platform	Files
Windows x64 Installer	cmake-3.23.1-windows-x86_64.msi
Windows x64 ZIP	cmake-3.23.1-windows-x86_64.zip
Windows I386 Installer	cmake-3.23.1-windows-i386.msi
Windows I386 ZIP	cmake-3.23.1-windows-i386.zip

2. 安装时，请勾选下图中的两个选项，以将 CMake 添加至环境变量，并创建 CMake 的桌面快捷方式。



安装 Visual Studio (2017 或以上版本)

1. 下载 [Visual Studio 安装包](#)。
2. 安装时，请勾选 **桌面应用和移动应用** 分类中的“使用 C++ 的桌面开发”工作负荷，再单击右下角的 [**安装**]。



添加环境变量

请根据以下步骤添加相关的环境变量。

1. 右键单击桌面上的“此电脑”，选择“属性”。
2. 选择“高级系统设置”，在弹出的“系统属性”对话框的“高级”选项卡上，单击 [**环境变量**]，进入“环境变量”界面。
3. 在“系统变量”下，单击 [**新建**]，在“新建系统变量”对话框的“变量名”框中输入 **MECHDL_DIR**，“变量值”框中输入 **xxx/mechdlk_sdk**，然后单击 [**确定**]。
4. 在“系统变量”框中，滚动到“Path”并双击它进入到“编辑环境变量”页面。
5. 单击右上角 [**新建**]，添加 **%MECHDL_DIR%**，再单击右下角 [**确定**]。



如果“编辑环境变量”框中有 %MMIND_DLK%，选中后单击右侧的[删除]按钮移除此变量。



构建及运行例程

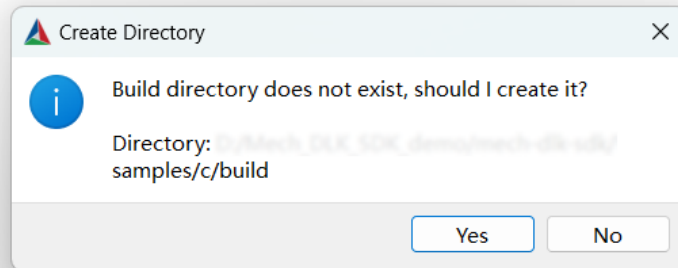
你可以一次性构建全部例程。

使用 CMake 配置例程

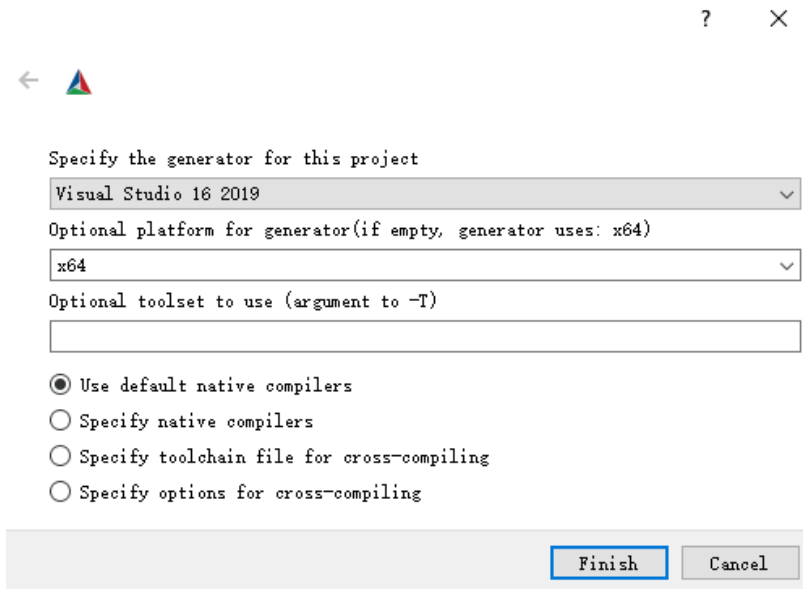
1. 双击 CMake 桌面图标打开软件。
2. 选择或输入源码路径与构建目录路径。如果没有构建目录路径，可以直接在源码路径后添加 `|build`，并继续以下的配置操作。

Where is the source code	xxx\mechdlk_sdk\samples\c
Where to build the binaries	xxx\mechdlk_sdk\samples\c\build

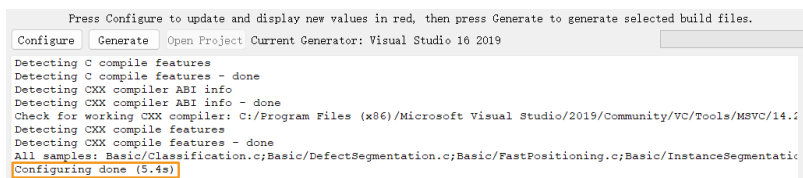
3. 单击 [**Configure**]。首次配置时，因没有构建目录路径，此时会弹窗请求构建 **build** 文件夹，请单击 [**Yes**]。



进入配置页面，选择 Visual Studio 的版本，然后单击 [**Finish**]。



配置成功后，日志最末行将显示 **Configuring done**。



4. 单击 [**Generate**] 生成解决方案。生成成功后，日志最末行将显示 **Generating done**。然后，单击 [**Open Project**]，使用 Visual Studio 打开解决方案。

```

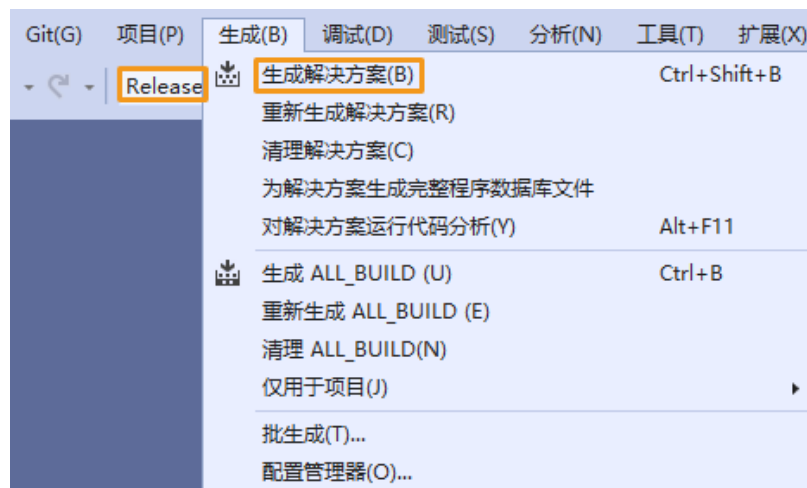
Press Configure to update and display new values in red, then press Generate to generate selected build files.
Configure Generate Open Project Current Generator: Visual Studio 16 2019
Detecting C compile features - done
Detecting CXX compiler ABI info
Detecting CXX compiler ABI info - done
Check for working CXX compiler: C:/Program Files (x86)/Microsoft Visual Studio/2019/Community/VC/Tools/MSVC/14.2...
Detecting CXX compile features
Detecting CXX compile features - done
All samples: Basic/Classification.c;Basic/DefectSegmentation.c;Basic/FastPositioning.c;Basic/InstanceSegmentation.c
Configuring done (5.4s)
Generating done (0.0s)
    
```



你也可以打开 **build** 文件夹（路径：`xxx\mechdlk_sdk\samples\c\build`），找到并双击 **MechDLKSDKCSamples.sln** 文件，使用 Visual Studio 打开解决方案。

使用 Visual Studio 构建例程

1. 确认 Visual Studio 工具栏中的解决方案配置为 **Release** 模式。当前未提供 **Debug** 版本的 DLL 文件。
2. 在菜单栏上选择 **生成 > 生成解决方案**。每个例程会生成对应的可执行文件（.exe），保存在 **Release** 文件夹中，位于 `xxx\mechdlk_sdk\samples\c\build` 目录下。



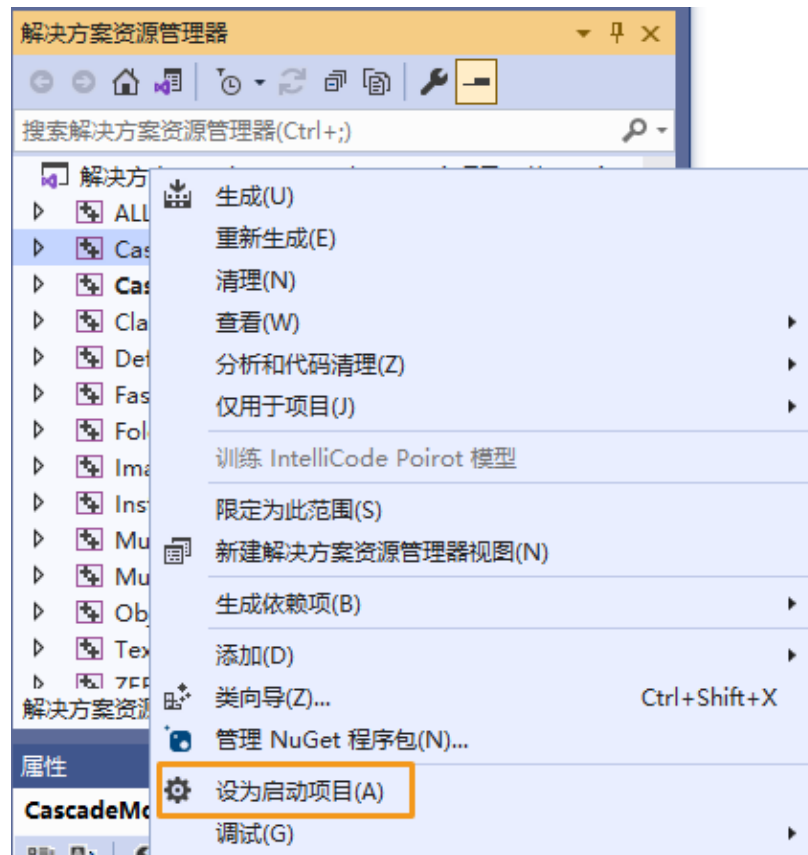
3. 将项目文件夹中的 **resources** 文件夹拷贝至以下目录下：
 - `xxx\mechdlk_sdk\samples\c\build\Release`
4. 将项目文件夹中的 **3rd_dll** 文件夹内的全部文件拷贝至 `xxx\mechdlk_sdk\samples\c\build\Release` 目录下。
5. 将项目文件夹中的 **mechdlk_sdk\dll** 文件夹内的全部文件拷贝至 `xxx\mechdlk_sdk\samples\c\build\Release` 目录下。

运行例程

你可以在 Visual Studio 中直接运行例程，也可以双击运行例程的可执行文件。

在 Visual Studio 中运行例程

1. 在 **解决方案资源管理器** 窗口中右键单击想要运行的例程，并选择 **设为启动项目**。



2. 单击工具栏中的[**本地 Windows 调试器**]运行该例程。

运行例程可执行文件

打开 **Release** 文件夹，双击运行与例程同名的可执行文件（.exe），即可得到运行结果。

5. API 参考手册

最新资源

C# API 参考手册

[查看详情](#)

C API 参考手册

[查看详情](#)

6. 常见问题

以下为有关 Mech-DLK SDK 问题的解答。如果你的问题没有解决，欢迎前往[梅卡曼德在线社区](#)的问答区发帖提问和交流。

模型相关

Mech-DLK 训练的模型可以在其他软件中使用吗？

Mech-DLK 训练的模型可以在其他软件中使用，只需该软件满足如下条件：

- 该软件支持使用 Mech-DLK SDK 加载推理模型。
- 该软件能够调用相应语言 API 编写的程序来使用该模型。

Mech-Vision / Mech-DLK SDK 可以支持什么模型？

Mech-Vision / Mech-DLK SDK 只能使用 Mech-DLK 训练出来的模型或由梅卡曼德提供的[超级模型](#)，无法使用其他模型。

Mech-DLK 训练的模型能不能转换成其他格式的模型？

不能，模型格式目前不能进行转换。

开发相关

Mech-DLK SDK 支持与哪些第三方软件集成？

Mech-DLK SDK 目前支持与 LabVIEW 进行集成，通过加载 C DLL 的方式实现调用。除此之外，只要第三方软件能够支持调用使用 C、C# 或 C++ API 编写的程序块，就可以集成或调用 Mech-DLK SDK。

Mech-DLK SDK 支持哪些语言的 API？

目前支持 C 及 C# APIs。C++ 以及 Python 语言的 API 正在开发中，如有需要，请联系梅卡曼德技术支持。