インテル® INDE 2015 で OpenCL* アプリケーションを開発する

この記事は、インテル® デベロッパー・ゾーンに公開されている「Using Intel® INDE 2015 to Develop OpenCL™ Applications」の日本語参考訳です。

はじめに

この記事は、インテル® INDE 2015 を使用した OpenCL* アプリケーション開発の概要を説明します。ここでは、Microsoft Visual Studio* で OpenCL* プロジェクトを作成し、添付の Sobel フィルター・アプリケーション・ファイルを読み込んで、OpenCL* コードを実行するデバイスを選択し、アプリケーションをビルドして実行します。インテル® INDE 2015 を利用することで、OpenCL* プロジェクトを簡単にセットアップできます。OpenCL* のインクルード・ディレクトリーやライブラリー・ディレクトリーをセットアップする必要はありません。インテル® INDE 2015 には、OpenCL* 向けにビルトインのコード強調表示機能やコード入力候補機能が用意されています。また、すべての OpenCL* ビルトイン関数とキーワードのヒントも提供されます。さらに、インテル® INDE 2015 に付属の OpenCL* Code Builder は、OpenCL* カーネルの迅速な開発、デバッグ、解析を可能にします。OpenCL* Code Builder については、別の記事で取り上げます。

動作環境

このチュートリアルには、Microsoft* Visual Studio* 2012 以上がインストールされた、インテル® プロセッサー・グラフィックス (インテル® Iris™ グラフィックスまたはインテル® Iris™ Pro グラフィックスを推奨) を搭載したインテル® プロセッサー・ベースの Microsoft* Windows* 7 以上の PC が必要です。さらに、チュートリアルを開始する前に、インテル® INDE 2015 をインストールする必要があります。また、サンプルの実行結果を視覚的に確認できるように、.ppm ファイルビューアーをインストールすることを推奨します。

Sobel オペレーターの概要

このチュートリアルでは、Sobel オペレーターの OpenCL* 実装を使用します (Sobel オペレーターの詳細は、http://en.wikipedia.org/wiki/Sobel_operator (英語) を参照してください)。このオペレーターは、2 つの 3×3 カーネル (1 つが水平の変更、もう 1 つが垂直の変更に対応) をオリジナル・イメージで畳み込み、微分係数の近似値を計算します。ソースイメージを A、2 つのイメージを Gx と Gy とし、各ポイントに水平と垂直の微分係数の近似値が含まれる場合、次のような計算になります。

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

これは、2 次元の畳み込み操作です。
x 軸は右方向に増加し、y 軸は下方向に増加します。イメージの各ポイントで、次の式を使用して、計算した勾配の近似値を組み合わせて勾配の程度を求めることができます。

$$\mathbf{G} = \sqrt{\mathbf{G}_x^{\,2} + \mathbf{G}_y^{\,2}}$$

G はイメージのピクセルごとに計算されます。

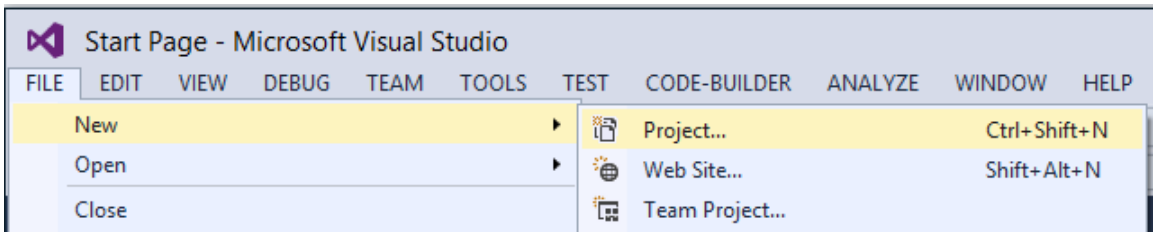
以下は、uchar バッファーを処理する Sobel カーネルの単純な実装です。

```
__kernel void Sobel_v1_uchar (__global uchar *pSrcImage, __global uchar
*pDstImage)
{
    uint dstYStride = get_global_size(0);
    uint dstIndex   = get_global_id(1) * dstYStride + get_global_id(0);
    uint srcYStride = dstYStride + 32;
    uint srcIndex   = get_global_id(1) * srcYStride + get_global_id(0) + 16;

    uint a,        b,        c;
    uint d,    /*中央*/  f;
    uint g,        h,        i;

    // データの読み取り
    a = pSrcImage[srcIndex-1];  b = pSrcImage[srcIndex];
    c = pSrcImage[srcIndex+1];

    srcIndex += srcYStride;
    d = pSrcImage[srcIndex-1];         /*中央*/
    f = pSrcImage[srcIndex+1];

    srcIndex += srcYStride;
    g = pSrcImage[srcIndex-1];  h = pSrcImage[srcIndex];
    i = pSrcImage[srcIndex+1];

    uint xVal =  a* 1 +                c*-1    +
                 d* 2 +    /*中央*/    f*-2    +
                 g* 1 +                i*-1;

    uint yVal =  a* 1 + b* 2 + c* 1 +
                      /*中央*/
                 g*-1 + h*-2 + i*-1;

    // データの書き込み
    pDstImage[dstIndex] =  min((uint)255, (uint)sqrt(xVal*xVal + yVal*yVal));
}
```

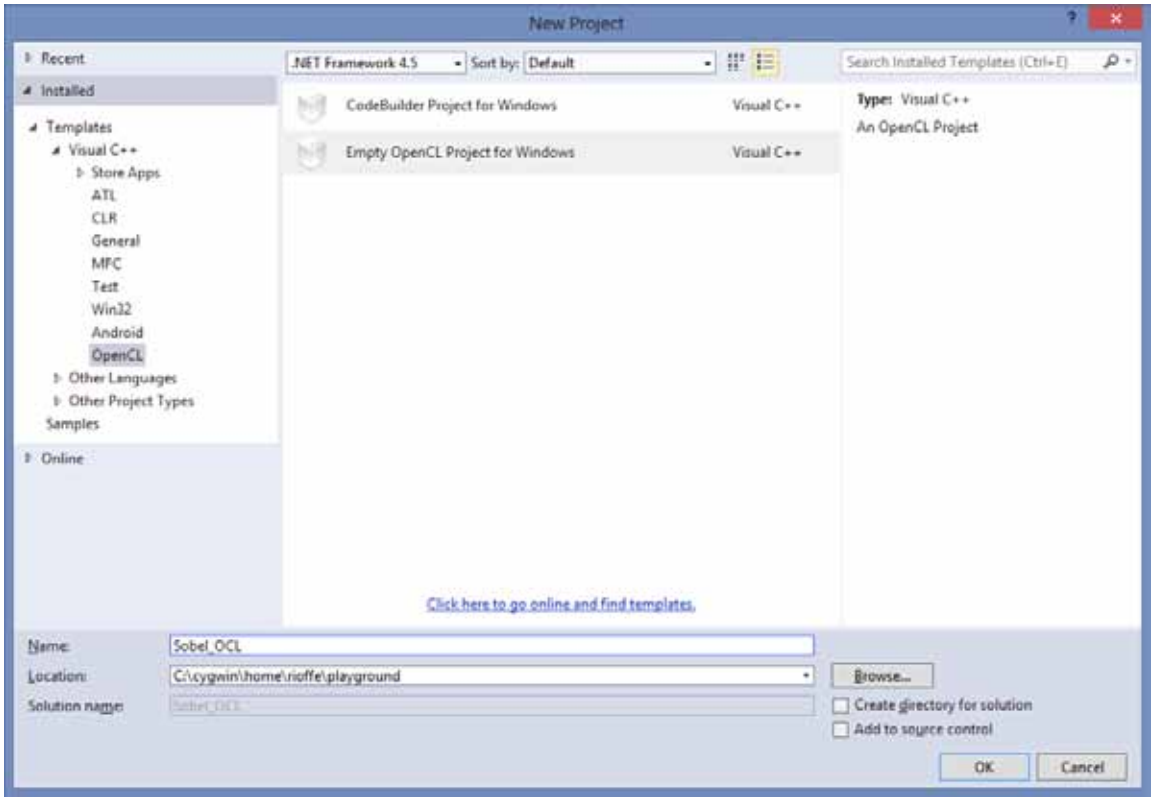
SobelKernels.cl ファイルには、このほかにもインテル® プロセッサー・グラフィックス向けに最適化された 3 つのカーネルが含まれています。それでは、チュートリアルを開始しましょう。

Microsoft* Visual Studio* で Sobel プロジェクトをセットアップする

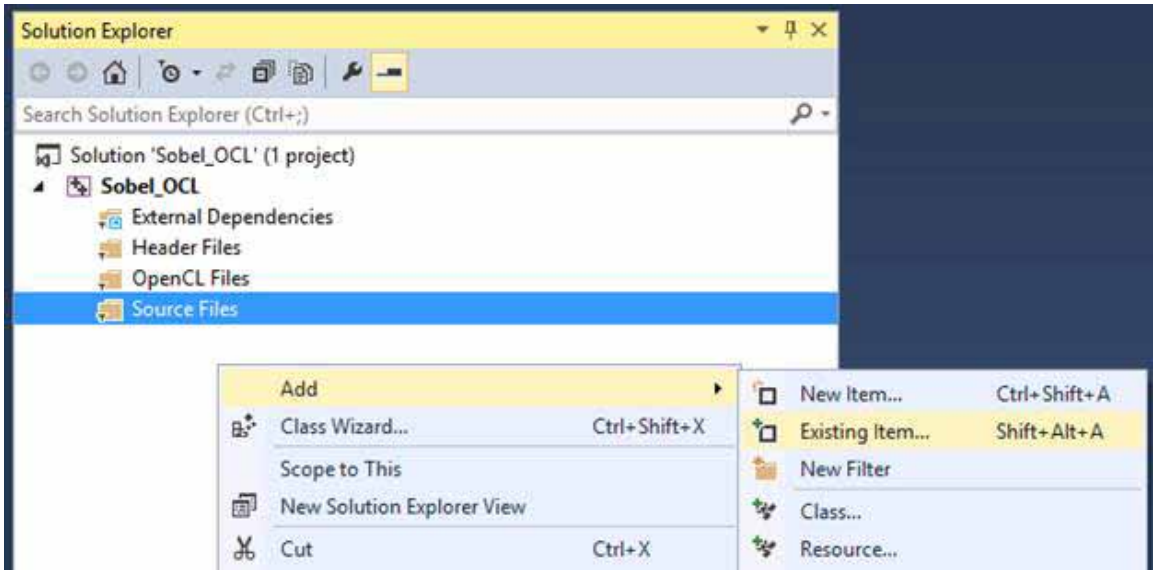
この記事の最後にあるリンクから Sobel_OCL.zip をダウンロードして展開します。Microsoft* Visual Studio* で [ファイル] > [新規作成] > [プロジェクト] を選択し、表示される [新しいプロジェクト] ウィンドウで [テンプレート] > [Visual C++] > [OpenCL] > [Empty OpenCL Project for Windows] を選択します。

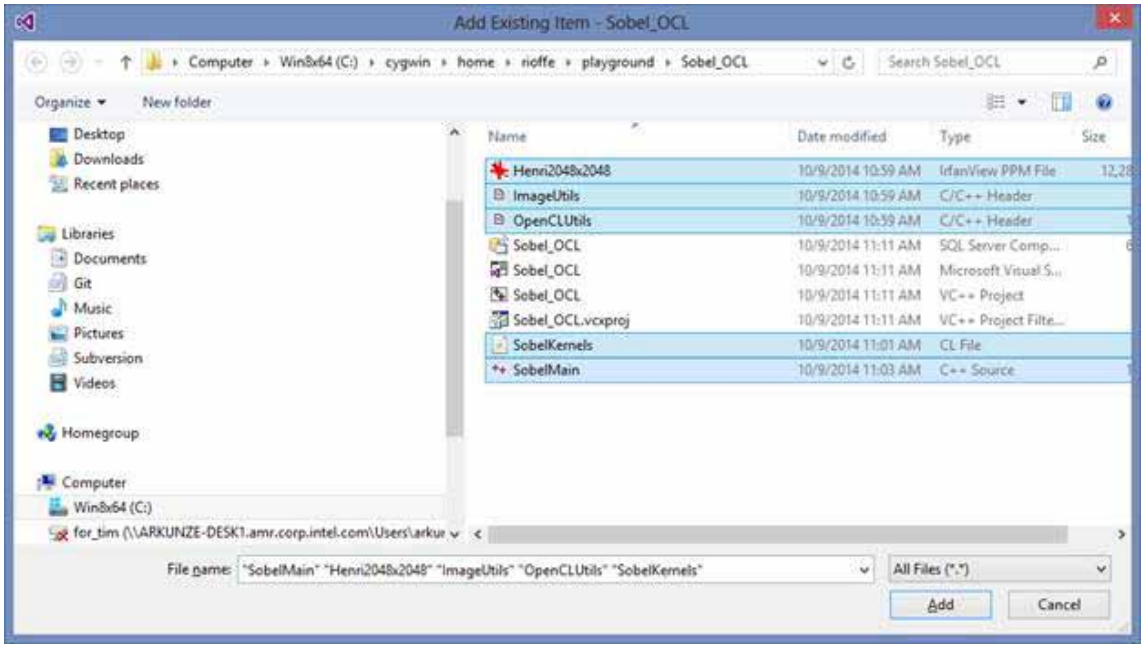


空の OpenCL* プロジェクトを作成します。プロジェクト名は Sobel_OCL とし、プロジェクト・ディレクトリーも Sobel_OCL ディレクトリーとします。必ず [ソリューションのディレクトリを作成] チェックボックスをオフにしてください。

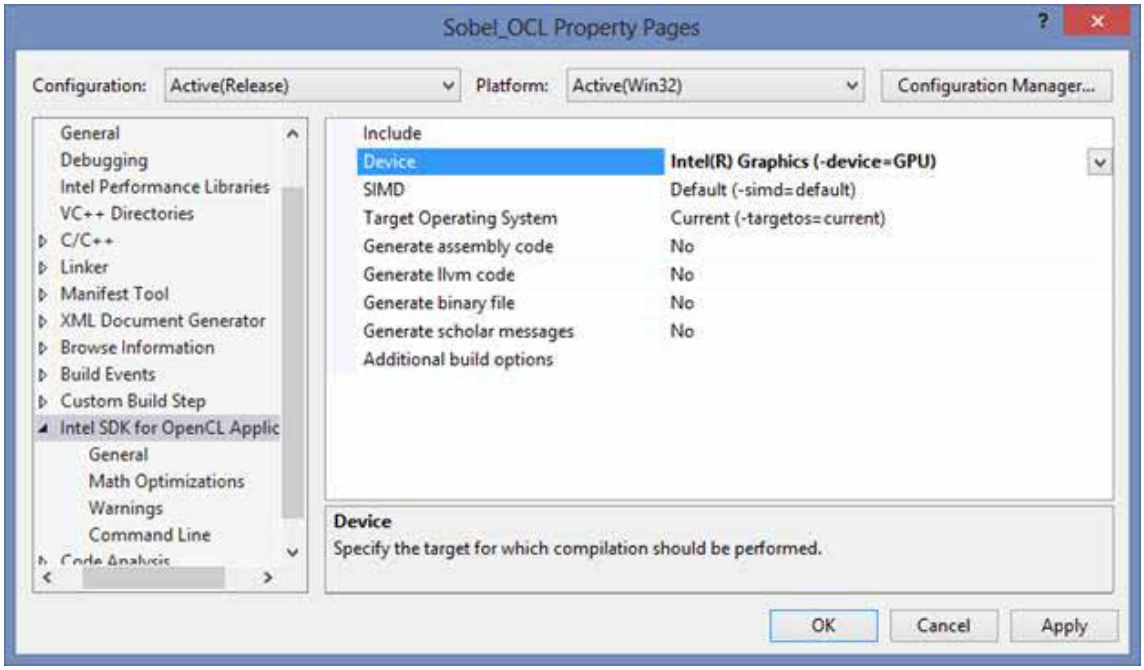


プロジェクトに既存のファイルを追加します。

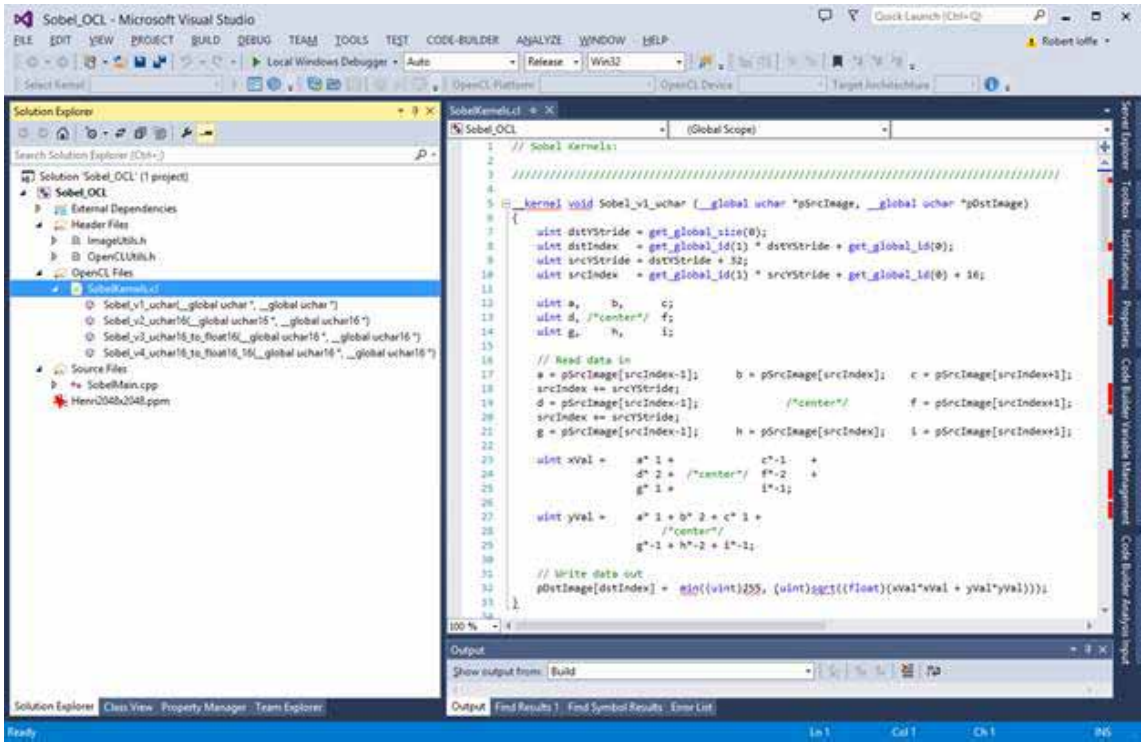




プロジェクト･プロパティーの [Intel SDK for OpenCL Applications] > [Device] **で** [Intel(R) Graphics (-device=GPU)] **を選択します。**



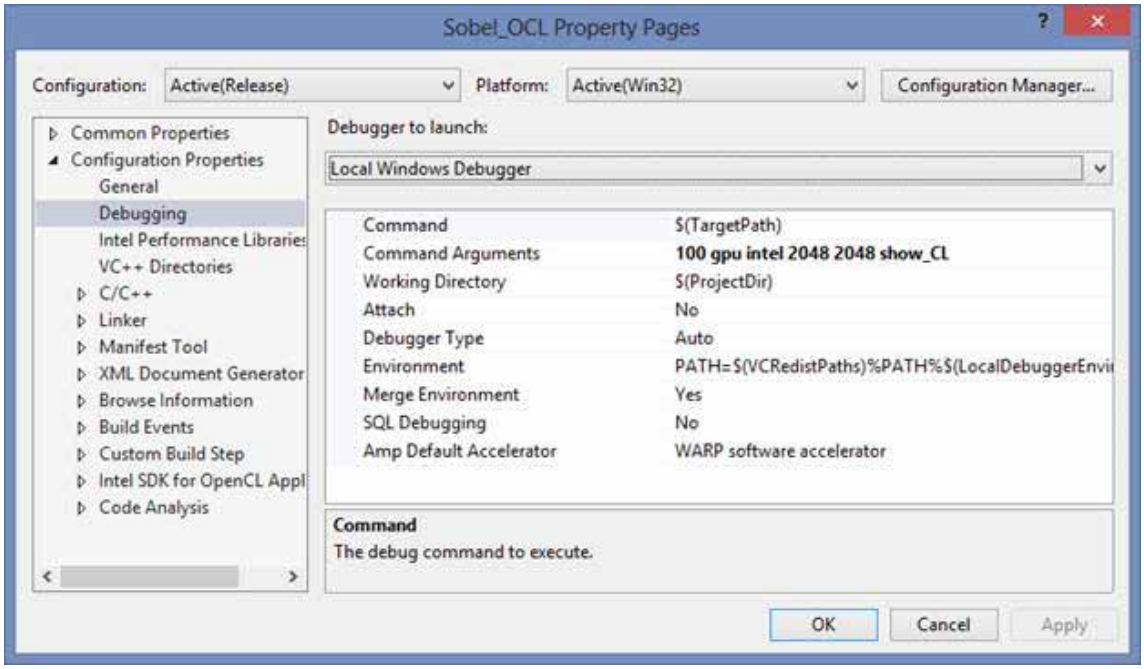
ソリューション エクスプローラーで SobelKernels.cl **を選択してカーネルを確認します。**



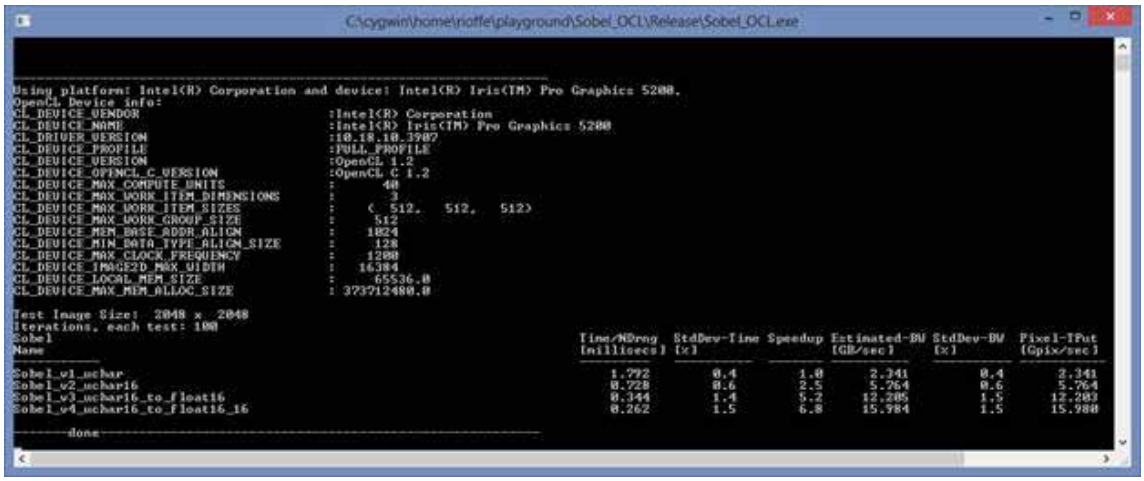
__kernel、get_global_id、uint **などのキーワードにホバー** (**マウスオーバー**) **して表示されるヒントを確認します。ソリューションの作成、ファイルの追加、適切なデバイスの選択が完了したので、**Release **構成でビルドしてみましょう** (SobelKernels.cl **は** OpenCL* API **オフライン・コンパイラーにより前処理されます**)。

```
1>------ Build started: Project: Sobel_OCL, Configuration: Release Win32 ------
1>  Preprocessing: SobelKernels.cl
1>  OpenCL Intel(R) Graphics device was found!
1>  Device name: Intel(R) Iris(TM) Pro Graphics 5200
1>  Device version: OpenCL 1.2
1>  Device vendor: Intel(R) Corporation
1>  Device profile: FULL_PROFILE
1>  fcl build 1 succeeded.
1>  fcl build 2 succeeded.
1>  bcl build succeeded.
1>
1>  Build succeeded!
1>
1>  SobelMain.cpp
1>  c:\cygwin\home\rioffe\playground\sobel_ocl\OpenCLUtils.h(149): warning C4996:
'clCreateCommandQueue': was declared deprecated
1>  C:\Intel\INDE\code_builder_4.6.0.118\include\cl/cl.h(1358) : see declaration
of 'clCreateCommandQueue'
1>  Generating code
1>  Finished generating code
1>  Sobel_OCL.vcxproj ->
C:\cygwin\home\rioffe\playground\Sobel_OCL\Release\Sobel_OCL.exe
1>  SobelKernels.cl
1>          1 file(s) copied.
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

プロジェクトのビルドが完了したら、実行ファイルのコマンドライン引数を設定します。



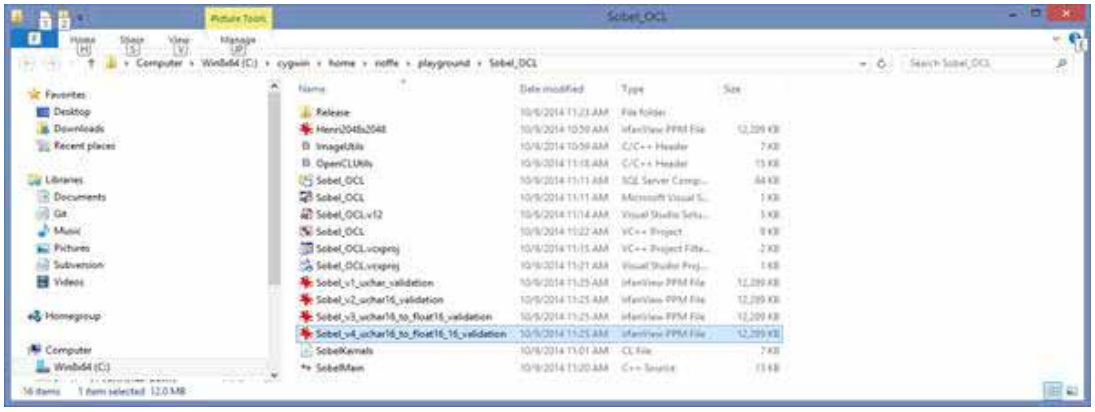
そして、アプリケーションを実行します。



後続のカーネルのほうがさらに最適化され、より高速になっていることが分かります。この最適化を利用して、皆さんのカーネルを最適化することができます。

`Sobel_v1_uchar`	**単純なスカラー実装**
`Sobel_v2_uchar16`	`uchar16` **を使用して行の** `16` **項目を処理**
`Sobel_v3_uchar16_to_float16`	`float16` **に変換して演算パフォーマンスを** `2` **倍に向上**
`Sobel_v4_uchar16_to_float16_16`	**一度に** `16` **行を処理**

最後に、4 **つの** *validation.ppm **ファイルに出力された結果を確認します。**



コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください

添付ファイル　サイズ

Sobel_OCL.zip 8.44MB