矢印記号認識を用いたロボット操作
RTC マニュアル

１．本コンポーネントの概要

symbolrecognition は、テンプレートマッチング法による矢印記号認識を行い，認識した矢印記号の種類によってロボットに移動指令を出力する RTC です。この RTC は、OpenCV の中にあるアルゴリズムを用いて矢印記号を認識しており、矢印記号を認識した際には vel2D の出力ポートから、TimedVelocity2D 型でロボットに移動指令を出力します。

２．開発環境

本コンポーネントの開発環境は以下の通りです。

・OS：Windows10 Home (64bit)
・コンパイラ：Microsoft Visual Studio Community 2015
・RT ミドルウェア (C++)：OpenRTM-aist–1.1.2-RELESE
・Eclipse：Eclipse SDK – 4.4.2
・CMake：CMake - 3.5.2
・OpenCV version：OpenCV 3.1

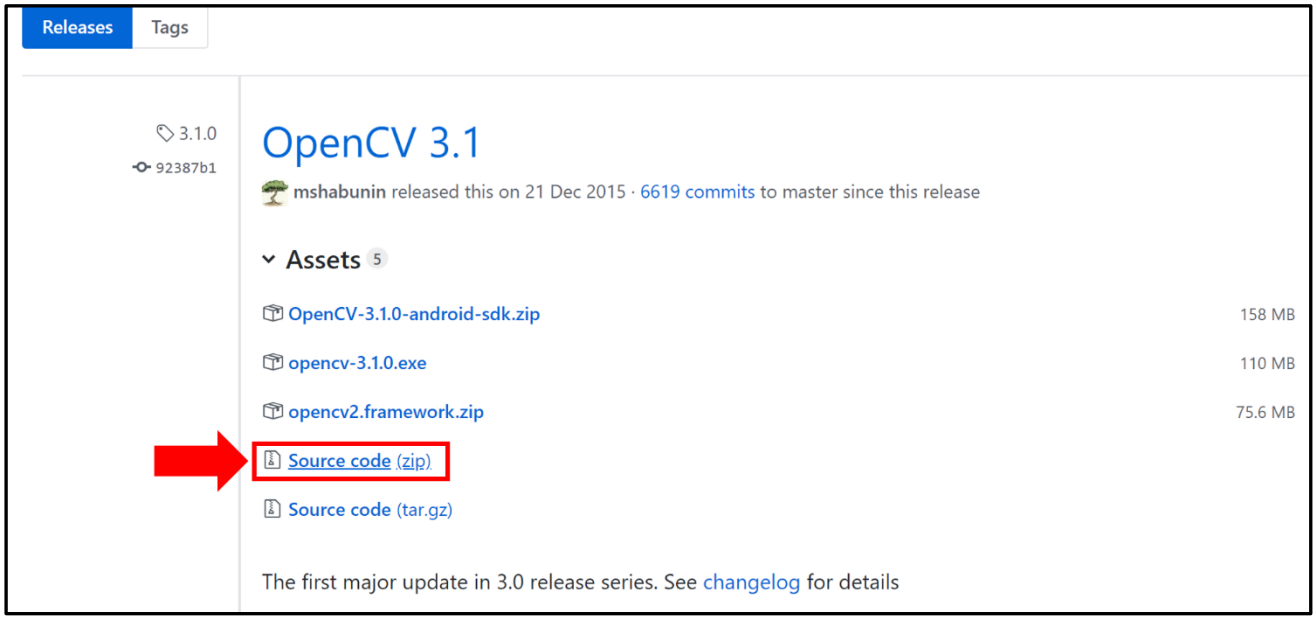
３．本コンポーネントを使用するまでの手順

注) OpenCV 3.1 がインストールされている方はこの項目は読み飛ばして頂いて結構です。

(1) OpenCV 3.1 のインストール

今回は CMake を用いて OpenCV のインストールを行います。

[1] 【https://github.com/opencv/opencv/releases/tag/3.1.0】のウェブページにアクセスし、『Source code (zip)』(=opencv-3.1.0.zip)をダウンロードします。



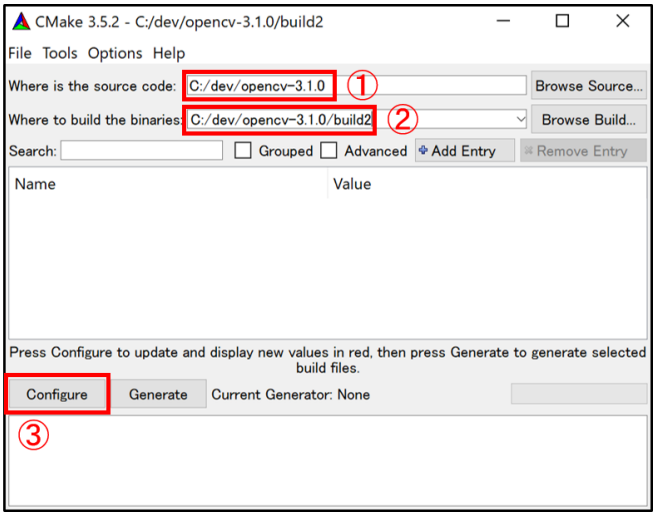
[2] ダウンロードした『opencv-3.1.0.zip』ファイルを展開して、自分の好きな場所に配置します。以降の説明は〔C:¥dev¥opencv-3.1.0〕に配置したものとして説明を行います。

[3] 次に OpenCV のライブラリをビルドするための Visual Studio ソリューションファイルを、以下の手順で作成します。

① CMake を開き、『Where is the source code』のテキストボックスに【C:/dev/opencv-3.1.0】と入力します。

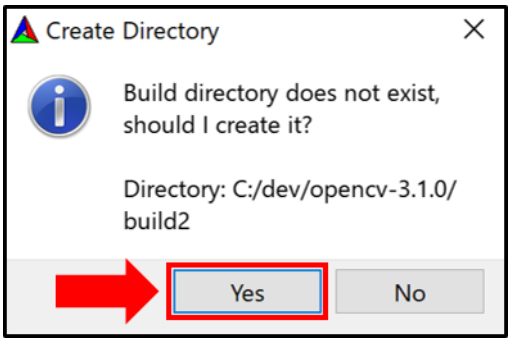
②『Where to build the binaries』のテキストボックスに【C:/dev/opencv-3.1.0/build】と入力します。

③『Configure』を押します。



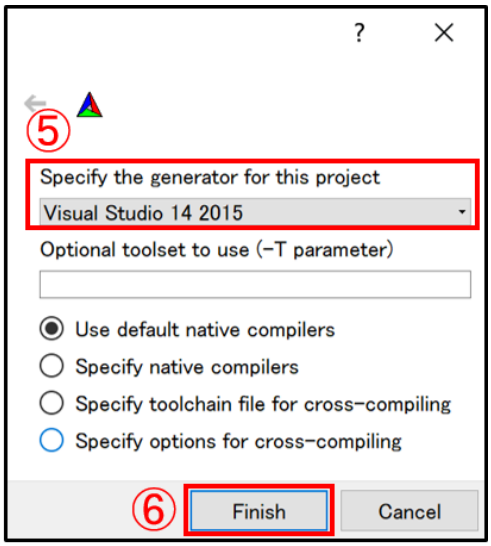
④『Configure』を押すと、Create Directory のウィンドウが表示されるので、『Yes』を押します。

注) 以下の画像から build2 となっている部分がありますが、手順に従っていると build となるはずです。

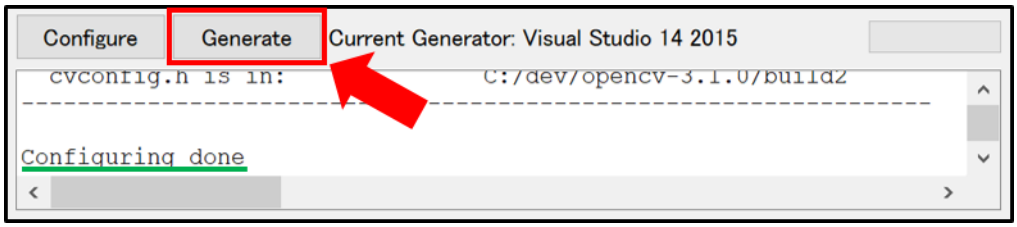


⑤『Specify the generator for this project』を選択し、自分が使用している Visual Studio のバージョンを選択します。

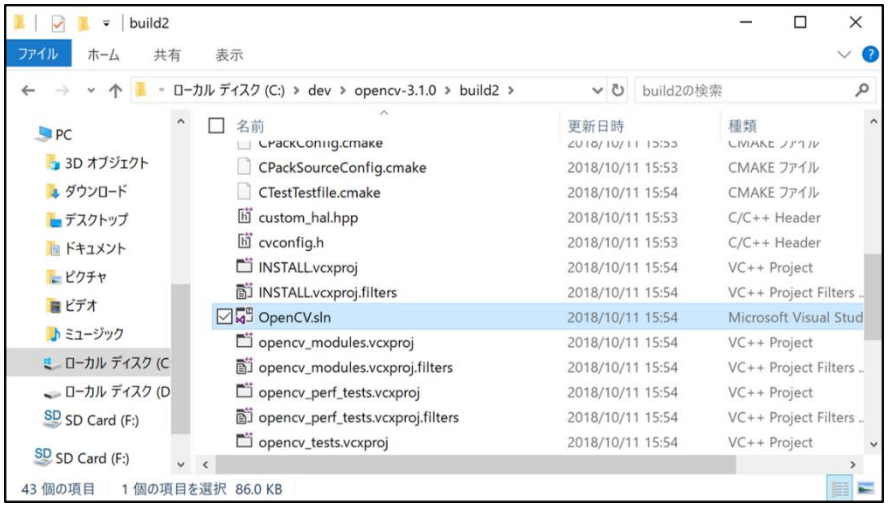
⑥『Finish』を押します。



⑦『Configure』の下のテキストボックスに〈Configuring done〉と表示されたら『Generate』を押します。



⑧『Generate』を押すと、〈Generating done〉と表示されたらソリューションファイルの作成が終了です。
ファイルは、〔C:¥dev¥opencv-3.1.0¥build〕フォルダの中に入っています。

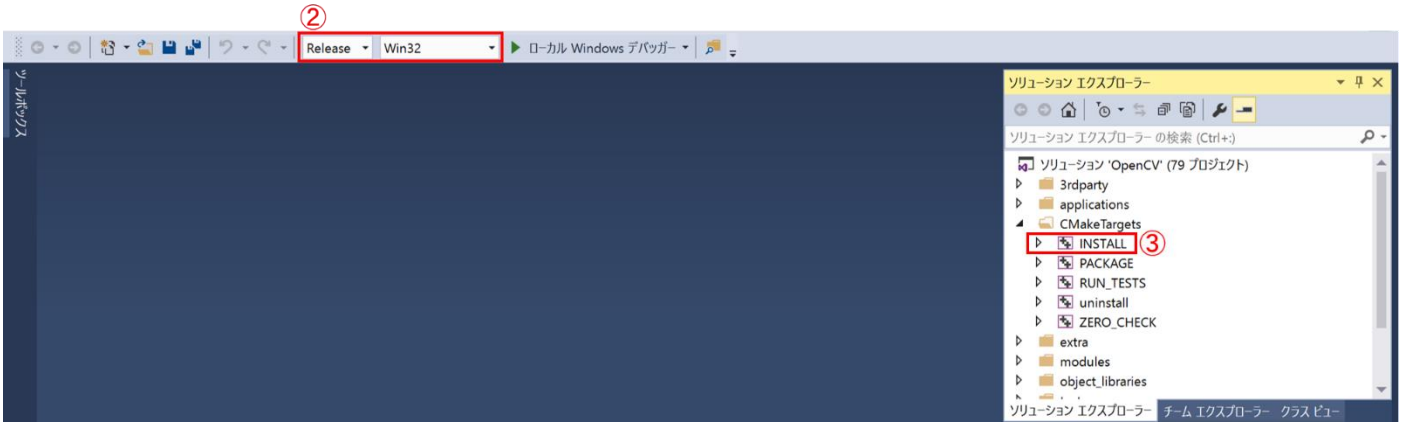


[4] 次はビルドを行っていきます。

① 先ほど作成した『OpenCV.sln』ファイルを開きます。

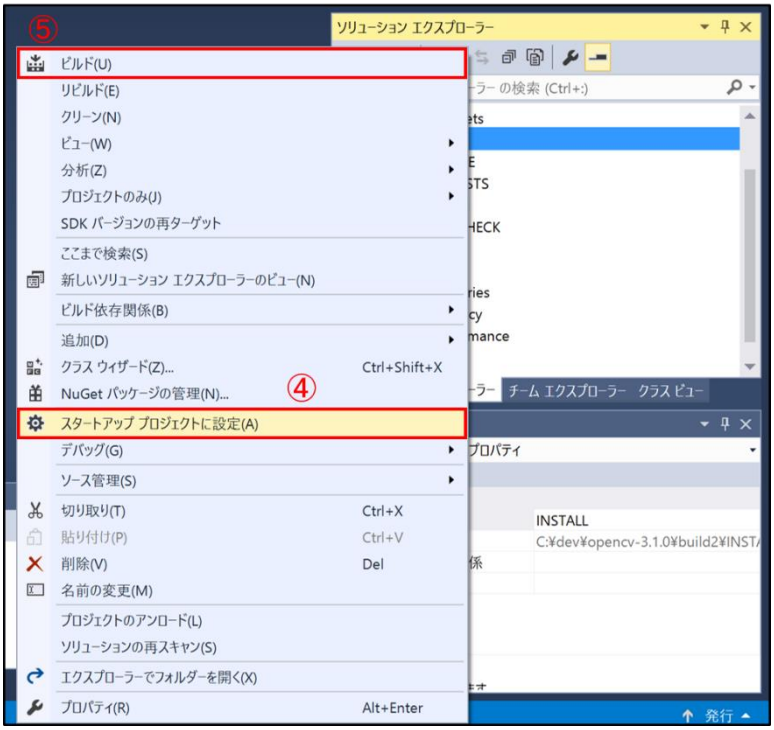
② ソリューション構成を「Debug」から「Release」に変更し、プラットフォームを指定します。

③ 次に『INSTALL』を右クリックします。



④「スタートアップ プロジェクトに設定」を押します。

⑤「ビルド」を押します。



[5] ビルドが完了すると、インクルードパス・ライブラリパス・DLL パスは以下のように配置されます。

インクルードパス：C:¥dev¥opencv-3.1.0¥build¥install¥include

ライブラリパス：C:¥dev¥opencv-3.1.0¥build¥install¥x86¥vc14¥lib

DLL パス：C:¥dev¥opencv-3.1.0¥build¥install¥x86¥vc14¥bin

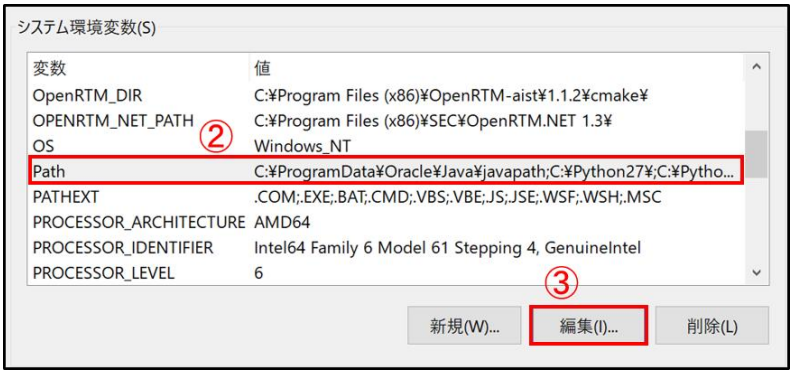
注) x86 は 32bit プラットフォームであること、vc14 は Visual Studio のバージョンが 2015 であることを示しています。x64 だと 64bit プラットフォーム、vc11 だと Visual Studio 2012、vc12 だと Visual Studio 2013 となるはずです。

[6] 次は環境変数設定を行います。

① Windows10 の場合、コントロールパネル→システムとセキュリティ→システム→システムの詳細設定→環境変数と進んでいきます。

② システム環境変数の中の『path』を選択します。

③『編集』を押します。



④ 環境変数名の編集というウィンドウが表示されると思うので、『新規』を選択し、
【C:¥dev¥opencv-3.1.0¥build¥install¥x86¥vc14¥bin】を追加します。

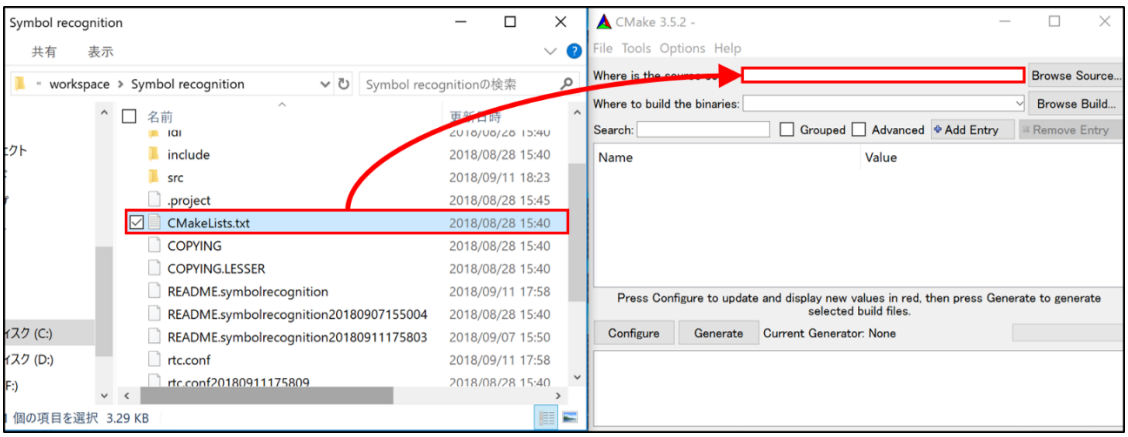
以上で OpenCV 3.1 のインストールは終了です。

(2) コンポーネントの準備

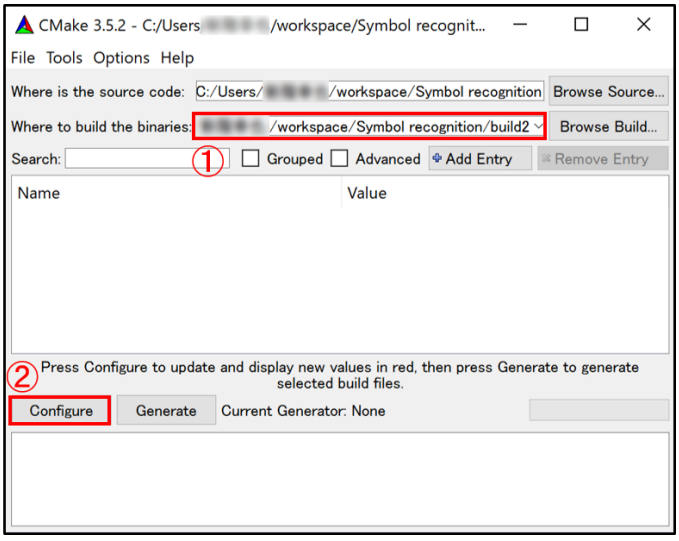
〇CMake について

[1] CMake とダウンロードした Symbol recognition のフォルダを開きます。

[2] ファイル内の CMakeList.txt を CMake の『Where is the source code』のテキストボックスに
ドラック＆ドロップします。



[3] ①『Where to build the binaries』のテキストボックスの最後に【/build2】を追加します。
②『Configure』を押します。

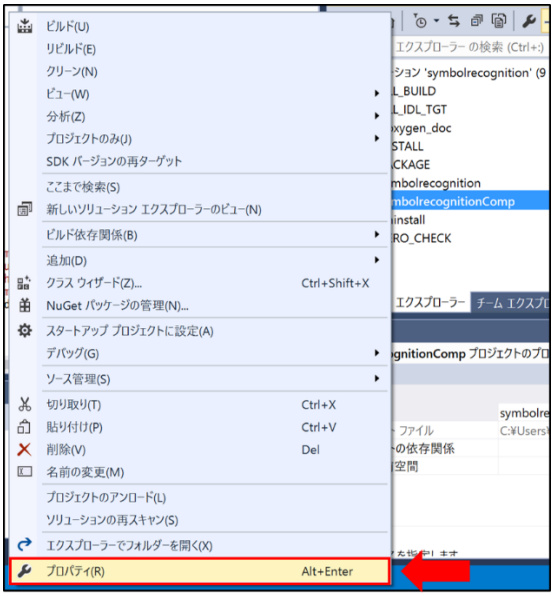


[4] この後は、３．の[3] ④～⑦と同様に進めていきます。すると CMakeLists.txt と同じ階層に build2 フォルダが作成されます。

[5] build2 フォルダの中にある『symbolrecognition.sln』を開きます。

○ソリューションファイルのプロパティ設定について

[1]『symbolrecognitionComp』を右クリックし、プロパティを押します。

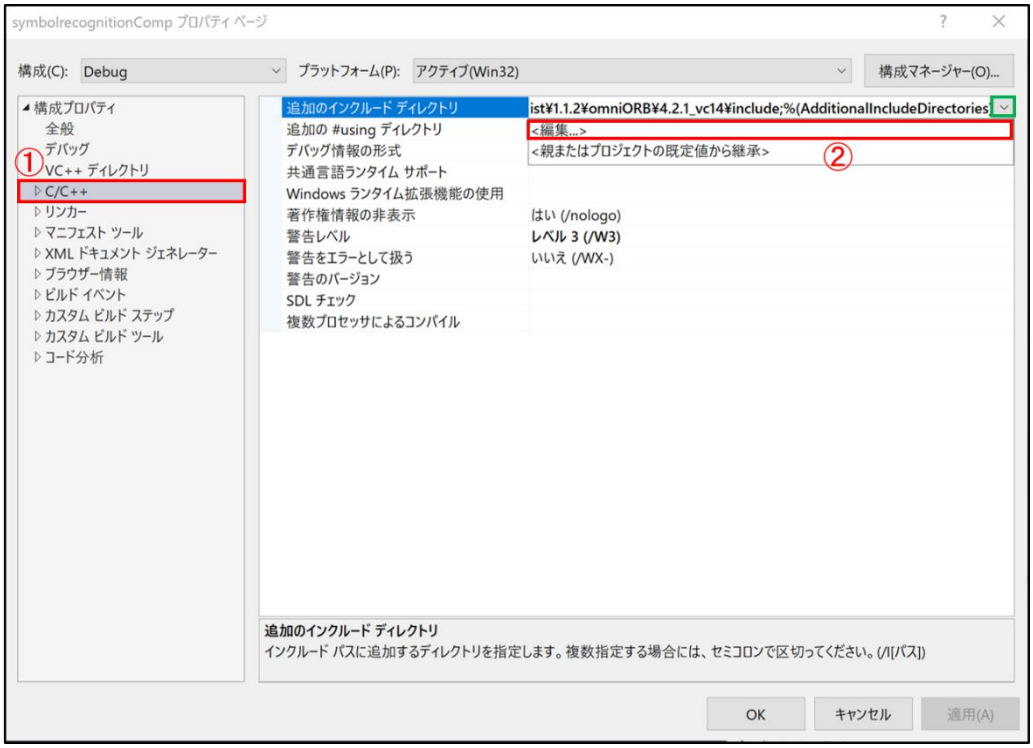


[2] symbolrecognitionComp プロパティページが開きます。

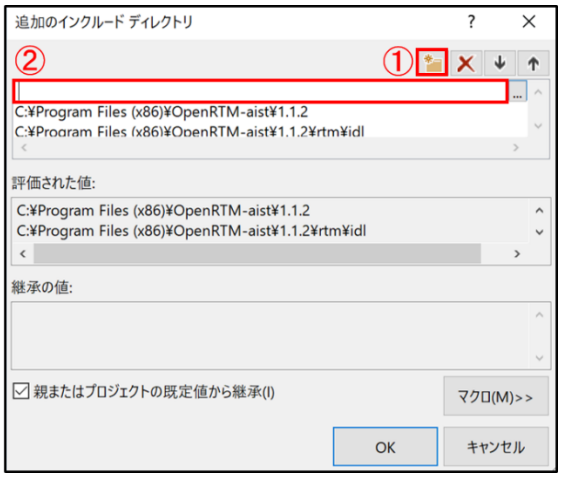
①『C/C++』を押します。

②『追加のインクルードディレクトリ』の緑の枠で囲まれた部分を押します。

次に、『<編集...>』を押します。すると追加のインクルードディレクトリのウィンドウが表示されます。



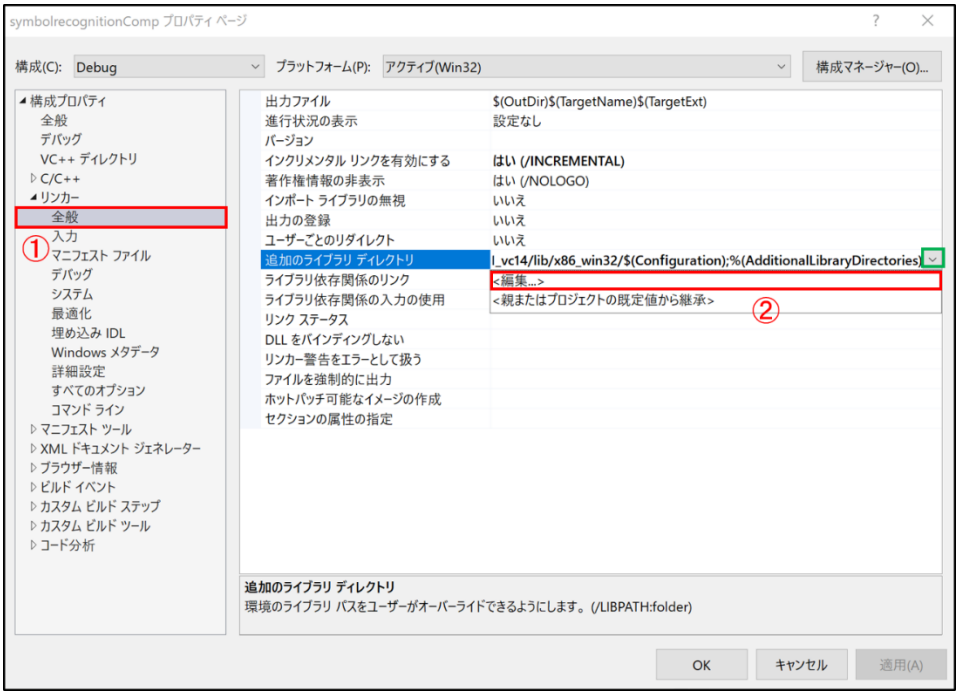
[3] ①のボタンを押すと、②の空欄があらわれます。②の空欄に、【C:¥dev¥opencv-3.1.0¥build¥install¥include】を入力します。入力が終わったら『OK』を押します。



[4]

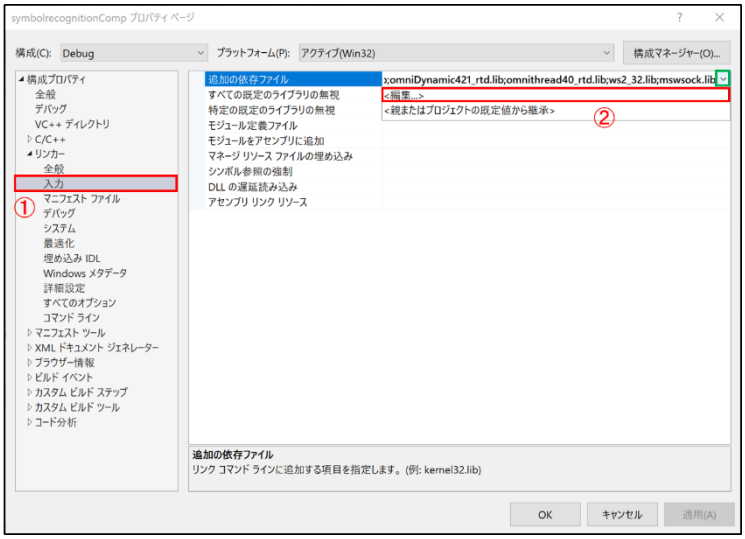
① 『リンカー』を選択し、その中にある『全般』を押します。

② [2]と同様に、『追加のライブラリディレクトリ』の緑の枠で囲まれた部分を押します。
そして、『<編集...>』を押します。すると追加のライブラリディレクトリのウィンドウが表示されます。

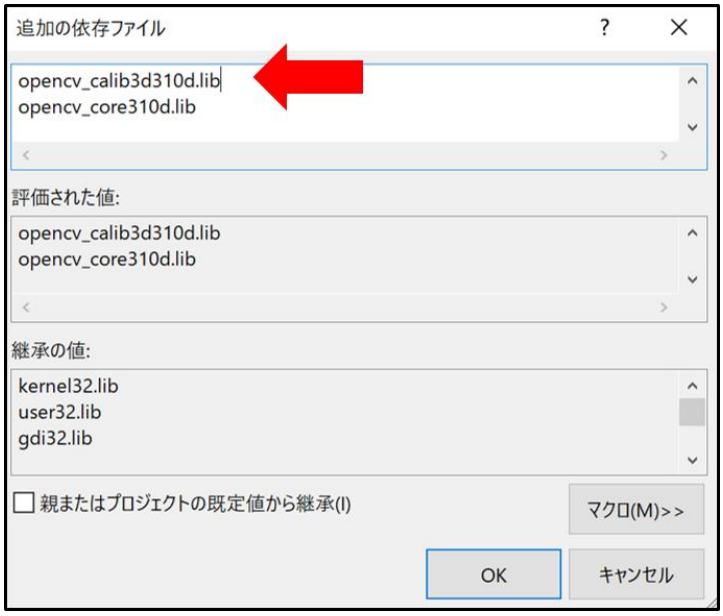


[5] [3]と同様にして、空欄に、【C:/dev/opencv-3.1.0/build/lib/Debug】を入力し、『OK』を押します。

[6]『リンカー』を選択し、『入力』を押します。今までと同様にして、追加の依存ファイルを選択し、『<編集…>』を押します。すると、追加の依存ファイルのウィンドウが表示されます。



[7] ウィンドウが表示されたら、テキストボックスに【opencv_calib3d310d.lib】、【opencv_core310d.lib】、【opencv_features2d310d.lib】、【opencv_flann310d.lib】、【opencv_highgui310d.lib】、【opencv_imgcodecs310d.lib】、【 opencv_imgproc310d.lib 】、【 opencv_ml310d.lib 】、【 opencv_objdetect310d.lib 】、【 opencv_photo310d.lib 】、【 opencv_shape310d.lib 】、【 opencv_stitching310d.lib 】、【 opencv_superres310d.lib 】、【 opencv_ts310d.lib 】、【opencv_video310d.lib】、【opencv_videoio310d.lib】、【opencv_videostab310d.lib】を追加します。追加したら『OK』を押します。



[8] symbolrecognitionComp プロパティページの『OK』を押します。

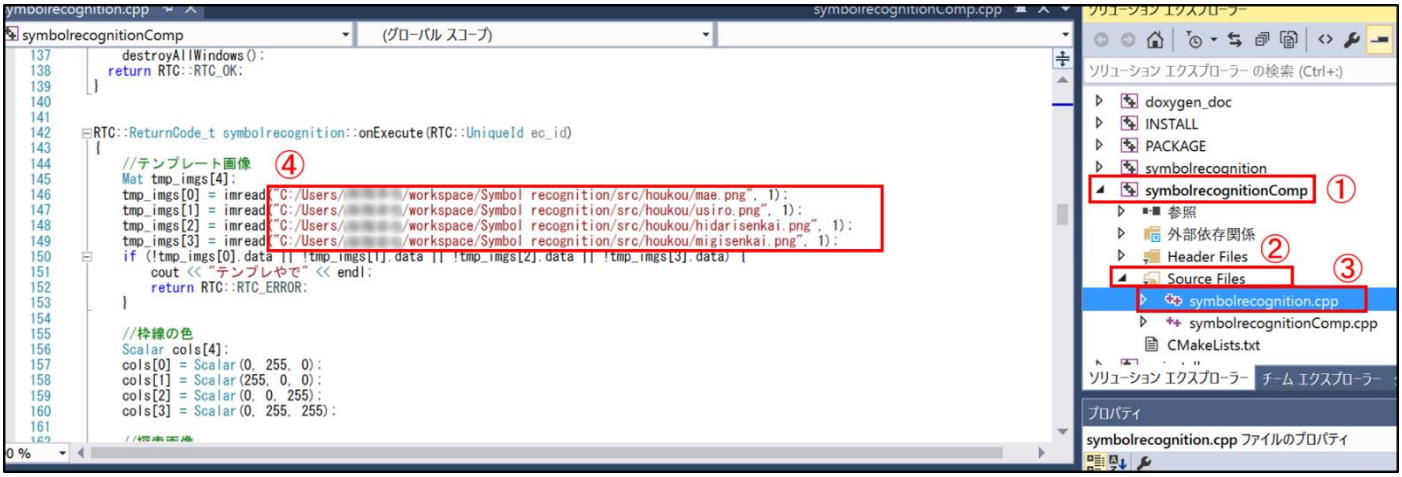
[9]

①『symbolrecognitionComp』の左にある矢印を押します。

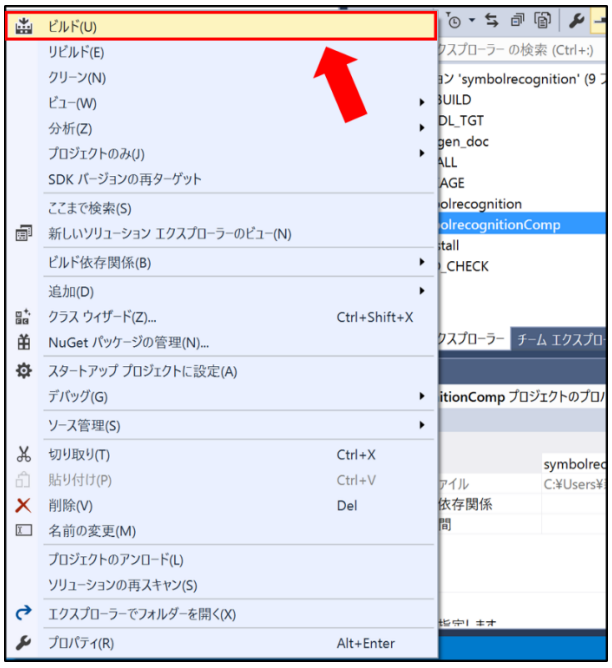
②『Source Files』の左にある矢印を押します。

③『symbolrecognition.cpp』を押します

④ コードが表示されると思うので、146～149 行目の参照されているファイルを変更します。
自分の workspace の中から読み込みができるように書き換えてください。

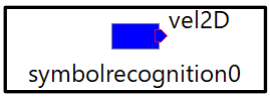


[10]『symbolrecognitionComp』を右クリックし、『ビルド』を押します。



[11] ビルドが終了したら、build2 フォルダ→src フォルダ→Debug フォルダと進んでいくと、フォルダの中に『symbolrecognitionComp.exe』が作成されます。

４．コンポーネントの使用方法



○データポートにつて

	名前	データ型
出力ポート	vel2D	TimedVelocity2D

○使用方法

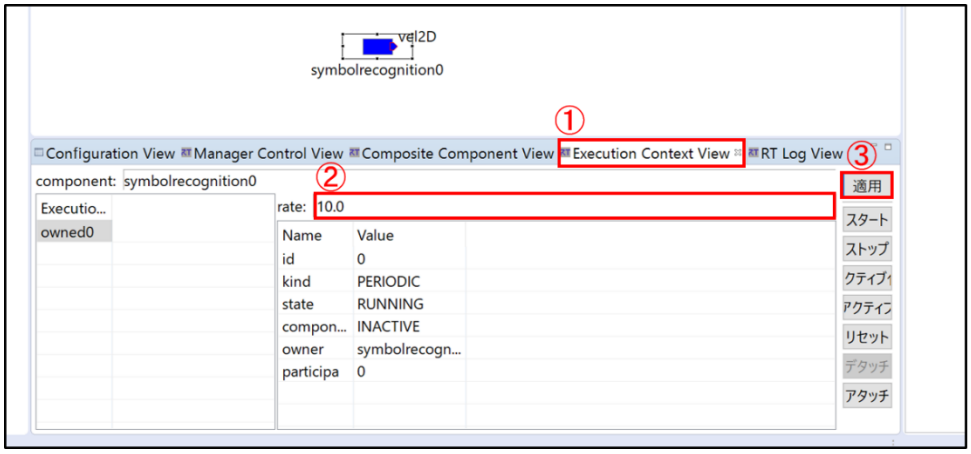
[1] symbolrecognitionRTC は矢印認識を行うためにカメラを用います。ノート pc などに内蔵されているカメラや、USB カメラをお使いください。

[2] 実行周期を自分が好きなように変更してください。

① まず、symbolrecognitionRTC と Eclipse を起動し、symbolrecognition を選択し、『Execution Context View』を押します。

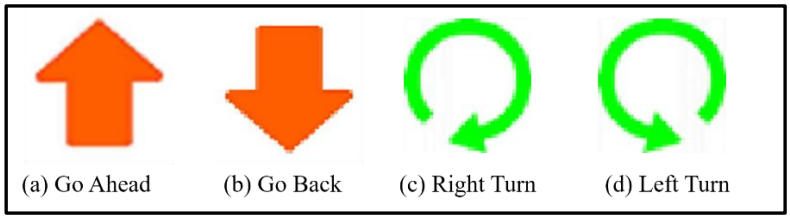
② 次に「rate」を自分が好きな値に変更します。写真では 10.0 としています。また、デフォルトの値は 1000.0 になっています。

③ 値を決めた後は適用を押します。

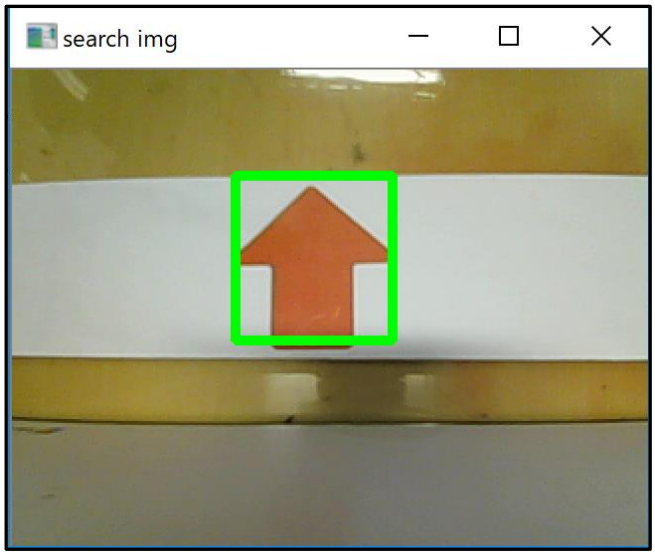


[3] symbolrecognitionRTC は以下の４つの矢印記号を認識するようになっています。

(a)を認識すると前進・(b)を認識すると後退・(c)を認識すると右旋回・(d)を認識すると左旋回の指令をロボットに送ります。矢印の画像は、Symbol recongnition ファイル→src ファイル→houkou ファイルの中にあります。



[4] 矢印記号を認識すると以下のように矢印記号が矩形で囲まれます。



５．問い合わせ先

本コンポーネントについての質問がございましたら、以下のメールアドレスまでご連絡ください。

東京理科大学 理工学部
機械工学科 4 年　新階 幸也
mail:7515060@ed.tus.ac.jp
