

実習の目的

- 教材用ロボットを用いて、実際のティーチング作業を学ぶ
- 実際のロボット活用事例にしばしば出てくるカメラとの協調システムに触れる
- 画像処理技術に触れる
- 実装・ハンズオンを通して技術的な提案・問題解決能力を養う

カメラとの協調システム

- ティーチングを実践した通り、ロボットは所定の位置に到達して決められた動作を行う装置
- ロボットやワークの位置が変わると同じタスクをこなすことができない

→カメラを使ってワークの位置をうまく抽出してワークの操作が行えないか

カメラとの協調における技術課題

1. カメラとロボットの座標の取り方と原点が異なる
 - 今回のロボットは天板中央が原点
 - 画像は一般的に左上が原点
2. 画像処理を用いた対象物の抽出
3. カメラ・処理系とロボットとの通信手段

今日は上記の課題をうまく解決する方法を紹介しながら
「ワークのパレタイジング作業」の実習を行います

実習課題 (ワークの自律パレタイジング作業)

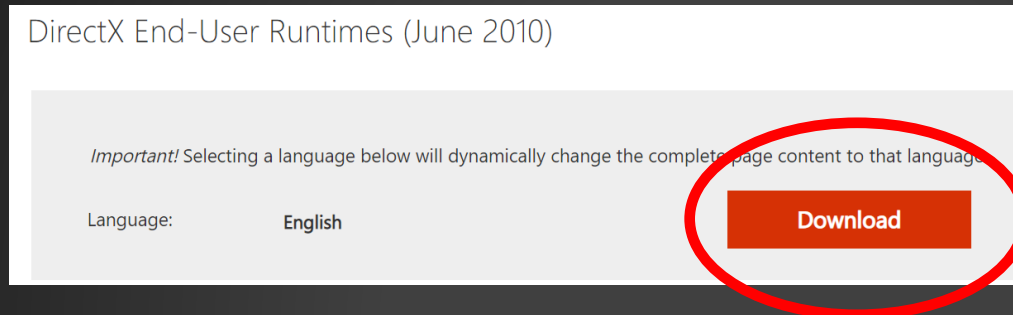
- カメラで撮影した把持対象物（ワーク）をロボットで把持して特定の場所に搬送する
 - カメラからの画像取得
 - 画像処理・対象物抽出技術
 - ※今回はOpenCVと呼ばれるライブラリを用いて、色による抽出を行う
 - 座標変換アルゴリズム（代数学）
 - カメラで撮影したワークの座標をロボットの座標系に変換する
 - 座標指定によるロボット制御
 - TCP/IP通信に関する知識
 - Python (Jupyter Lab) を用いたプログラミング技術

事前準備

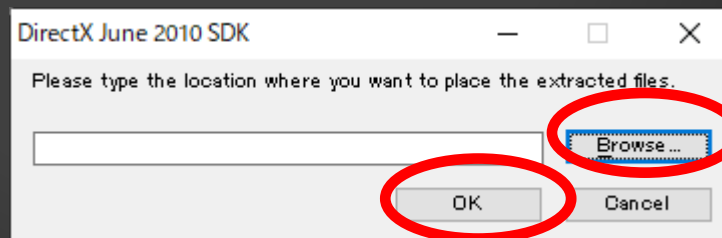
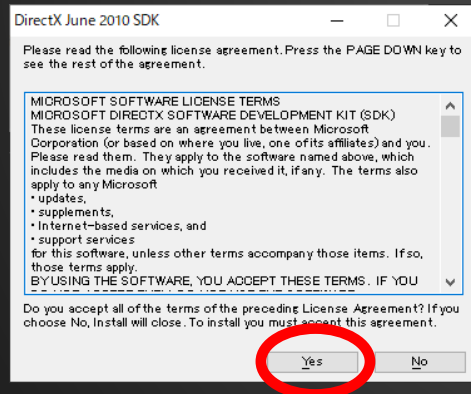
- オンライン実習では支給PCに事前セットアップを実施済みです
 - OpenCV-pythonのセットアップ
 - TCP/IPを利用してロボットを動作させるアプリケーション

DirectXライブラリのセットアップ

- 下記のURLからダウンロードを行います
 - <https://www.microsoft.com/en-us/download/details.aspx?id=8109>



- ダウンロードされた「directx_Jun2010_redist.exe」を実行します
- 利用規約に同意した後、展開先ディレクトリ（※）の指定画面に遷移するので、ディレクトリを選択して「OK」を押下します



- 展開先ディレクトリ内の「DXSETUP.EXE」を実行します
（※）セットアップ終了後、展開したディレクトリやファイルは削除して構いません

CP2110ソフトウェアのダウンロード

- 下記URLに遷移後、「ソフトウェアとツール」を選択します
 - <https://jp.silabs.com/products/development-tools/interface/cp2110ek-evaluation-kit>
- Windows版インストーラをダウンロードしてセットアップします



CP2110EK

CP2110 HID USB - UART ブリッジ開発キット

MSRP\$39.00

[🛒 今すぐ購入](#)

[📖 ユーザ・ガイド](#)

この CP2110EK HID USB - UART ブリッジ評価キットでは、すべての GPIO 機能、フロー制御、RS-485 トランシーバ制御および LED の送受信を含む、CP2110HID USB - UART ブリッジの完全な評価およびカスタマイズが可能です。A USB cable, a serial cable, Windows®, and Mac® application examples and full documentation are included.

[概要](#) [スタート・ガイド](#) [技術文書](#) [ソフトウェアとツール](#) [コミュニティとサポート](#)

CP2110 ソフトウェア

これらのパッケージには、CP2110/4 ファミリーに関して、ドキュメント、プログラミングサンプル、カスタマイズユーティリティが含まれています。

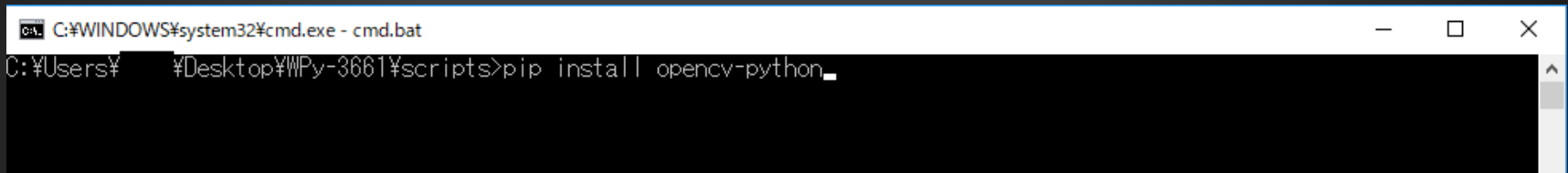
CP2110/4 HID USB-to-UART インターフェースライブラリには、CP2110 デバイスを設定、作動させるシンプル API が含まれています。ライブラリにはインターフェース・アブストラクションが含まれており、これによりユーザーは、USB HID コードを書かなくてもアプリケーションを開発できます。

プラットフォーム	ソフトウェアとツール	リリース・ノート
 Windows	ダウンロード	リリース・ノート
 Mac	ダウンロード	リリース・ノート
 Linux	ダウンロード	リリース・ノート

opencv-pythonのセットアップ

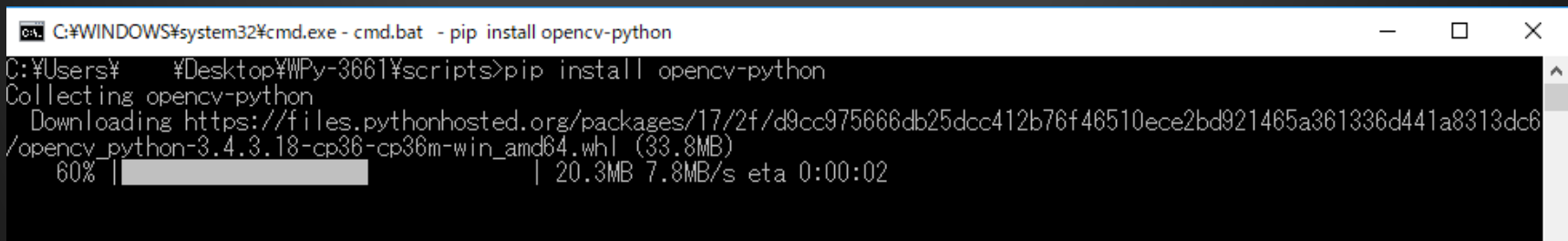
- WinPythonへのopencv-pythonのインストール手順

- (1) WinPythonを展開したディレクトリ内にある「WinPython Command Prompt.exe」を起動する
- (2) 「**pip install opencv-python**」 と入力しEnterキーを押下する



```
C:\WINDOWS\system32\cmd.exe - cmd.bat
C:\Users\%USER%\Desktop\WPY-3661\scripts>pip install opencv-python_
```

- (3) ダウンロードとインストールが始まるので暫く待つ



```
C:\WINDOWS\system32\cmd.exe - cmd.bat - pip install opencv-python
C:\Users\%USER%\Desktop\WPY-3661\scripts>pip install opencv-python
Collecting opencv-python
  Downloading https://files.pythonhosted.org/packages/17/2f/d9cc975666db25dcc412b76f46510ece2bd921465a361336d441a8313dc6/opencv_python-3.4.3.18-cp36-cp36m-win_amd64.whl (33.8MB)
    60% |██████████| 20.3MB 7.8MB/s eta 0:00:02
```

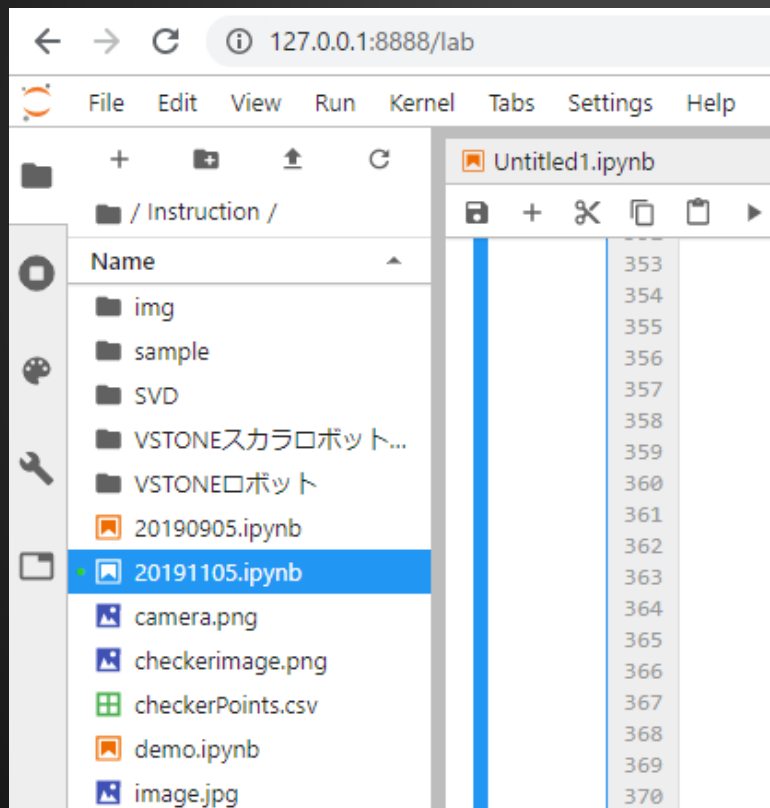
- (4) 「Successfully installed opencv-python-・・・」が表示されればインストール完了（表示されない場合は実習当日対応します）

opencv-pythonのセットアップ

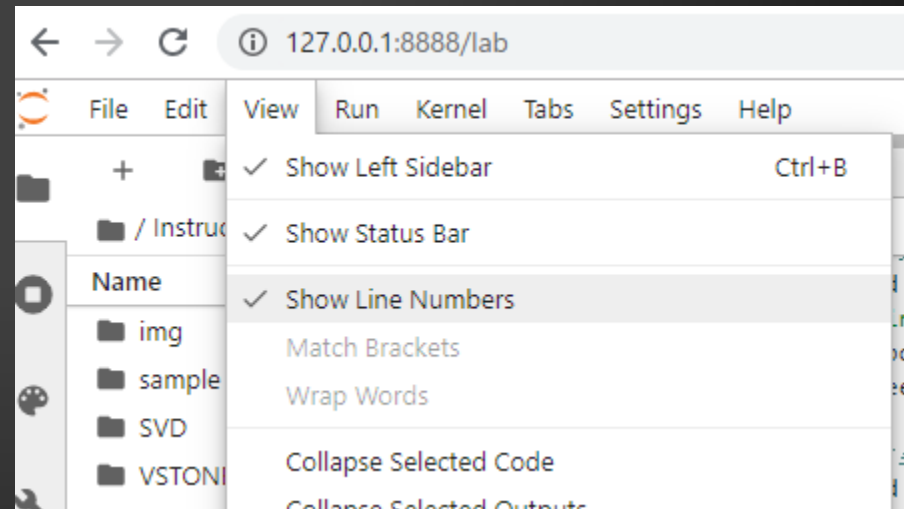
- OpenCV:画像処理ライブラリ
 - Intelが開発したオープンソースのライブラリでC/C++により記述
 - 様々なプラットフォーム, 開発言語への移植がなされている
- 画像処理 (Image Processing)
 - 勾配、エッジ、コーナー (Gradients, Edges and Corners)
 - サンプリング、補間、幾何変換 (Sampling, Interpolation and Geometrical Transforms)
 - モルフォロジー (英語版) 演算 (Morphological Operations)
 - フィルタと色変換 (Filters and Color Conversion)
 - ピラミッドとその応用 (Pyramids and the Applications)
 - 画像分割、領域結合、輪郭検出 (Image Segmentation, Connected Components and Contour Retrieval)
 - 画像と形状のモーメント (Image and Contour Moments)
 - 特殊な画像変換 (Special Image Transforms)
 - ヒストグラム (Histograms)
 - マッチング (Matching)
 - ラベリング (Labeling) : OpenCV 3.0以降
- 構造解析 (Structural Analysis)
 - 輪郭処理 (Contour Processing)
 - 計算幾何 (Computational Geometry)
 - 平面再分割 (Planar Subdivisions)
- モーション解析と物体追跡 (Motion Analysis and Object Tracking)
 - 背景統計量の累積 (Accumulation of Background Statistics)
 - モーションテンプレート (Motion Templates)
 - 物体追跡 (Object Tracking)
 - オプティカルフロー (Optical Flow)
 - 推定器 (Estimators)
- パターン認識 (Pattern Recognition)
 - 物体検出 (Object Detection)
- カメラキャリブレーションと3次元再構成 (Camera Calibration and 3D Reconstruction)
 - カメラキャリブレーション (Camera Calibration)
 - 姿勢推定 (Pose Estimation)
 - エピポーラ幾何 (Epipolar Geometry)
- 機械学習
 - 単純ベイズ分類器 (Naive Bayes Classifier)
 - k近傍法 (K Nearest Neighbors)
 - サポートベクターマシン (SVM)
 - 決定木 (Decision Trees)
 - ブースティング (Boosting)
 - Random forest (Random forest)
 - EMアルゴリズム (Expectation-Maximization)
 - ニューラルネットワーク (Neural Networks)
- ユーザインタフェース
 - シンプルGUI (Simple GUI)
 - 画像の読み込みと保存 (Loading and Saving Images)
 - ビデオ入出力 (Video I/O)
 - OpenGL/Direct3D相互運用

実習ソース等のセットアップ

- 実習サイトからダウンロード済みのファイルをWinPythonのインストールディレクトリ直下の「notebook」ディレクトリに移動する

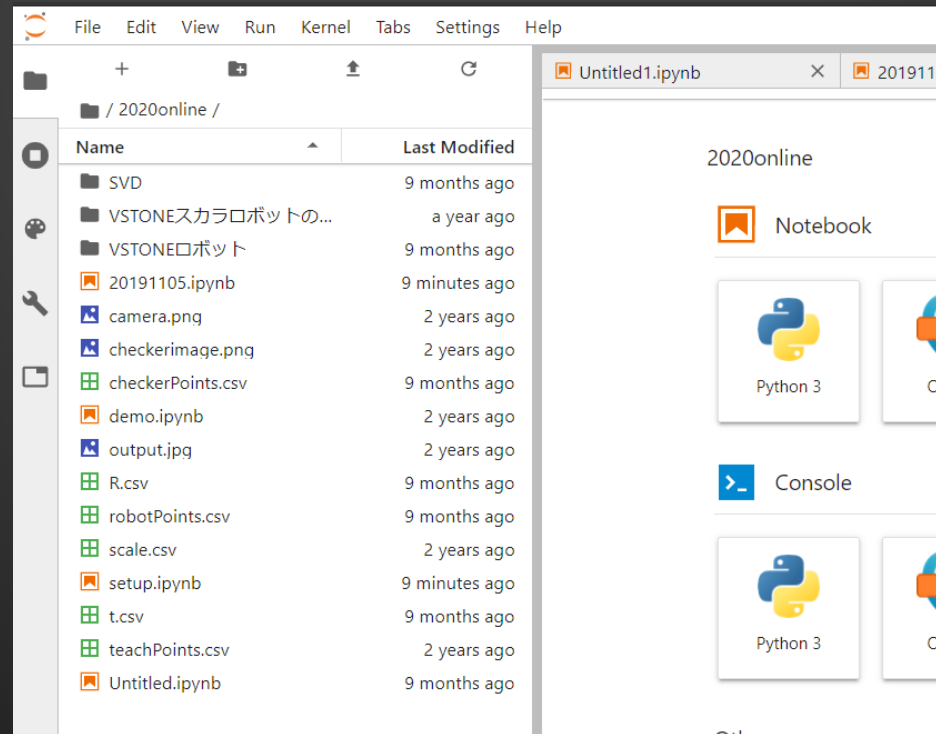


- ダブルクリックすると編集画面が開く
- [View]→[Show Line Number]にチェックを入れて、エディタに行番号が表示されるようにする



配布したプログラムの配置

- デスクトップの「ロボット実習WinPython」を開きます
- 「scripts¥winjupyter_lab.bat」を起動します
- 展開したファイルが左ペインのファイルブラウザに表示されるます

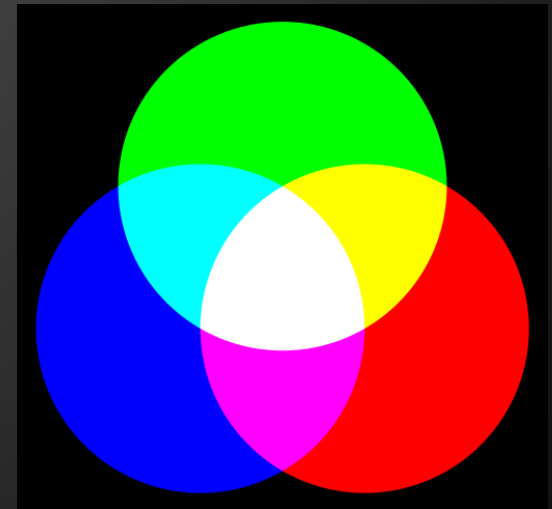
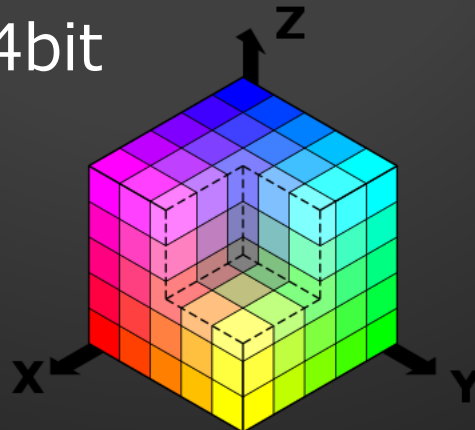


画像処理で対象物を抽出する

- 色に基づいて対象物を抽出する
 - 色空間を用いたセグメンテーション
 - 今回扱う方法
- 形状に基づいて対象物を抽出する
 - 形状マッチング（楕円・円・矩形・凸包）
 - 機械学習

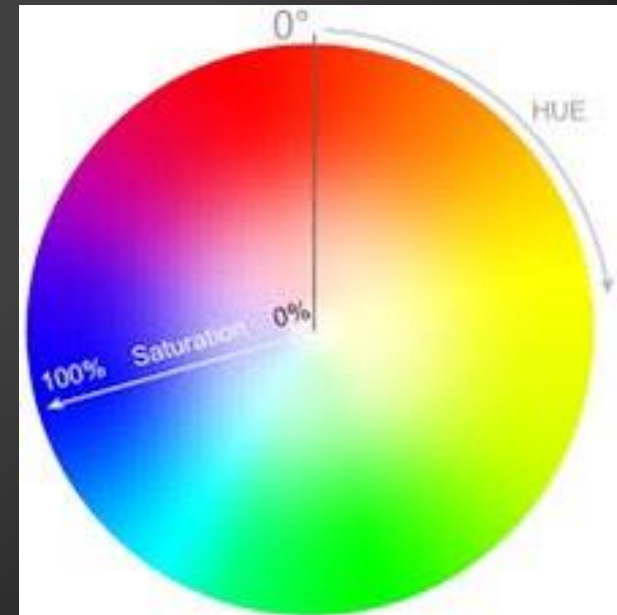
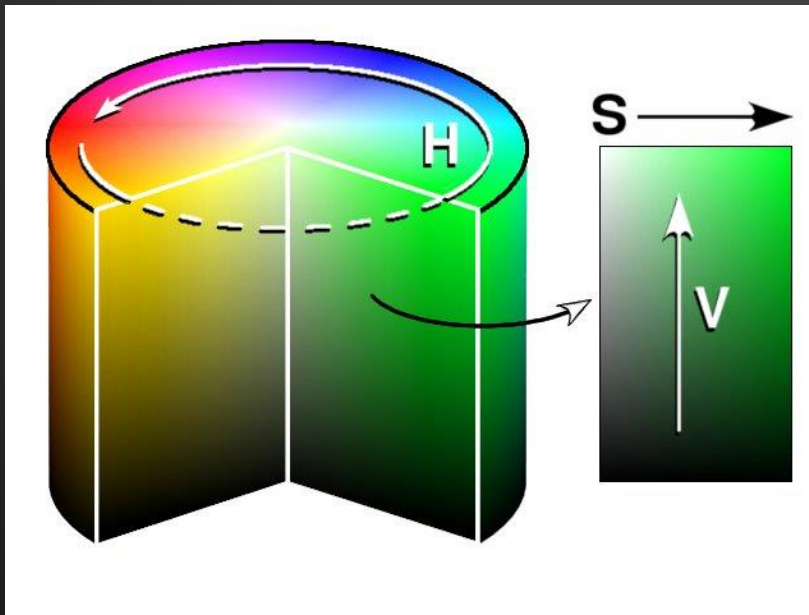
色空間の話

- RGB色空間
 - 1画素に定義される色をRGBの3要素により表現する
 - 色は全部足すと白になる（黒は光がないことを意味する）
 - 1ch/8bitの場合は0～255(0x00～0xFF) の数値で表現
 - RGBの3chで24bit



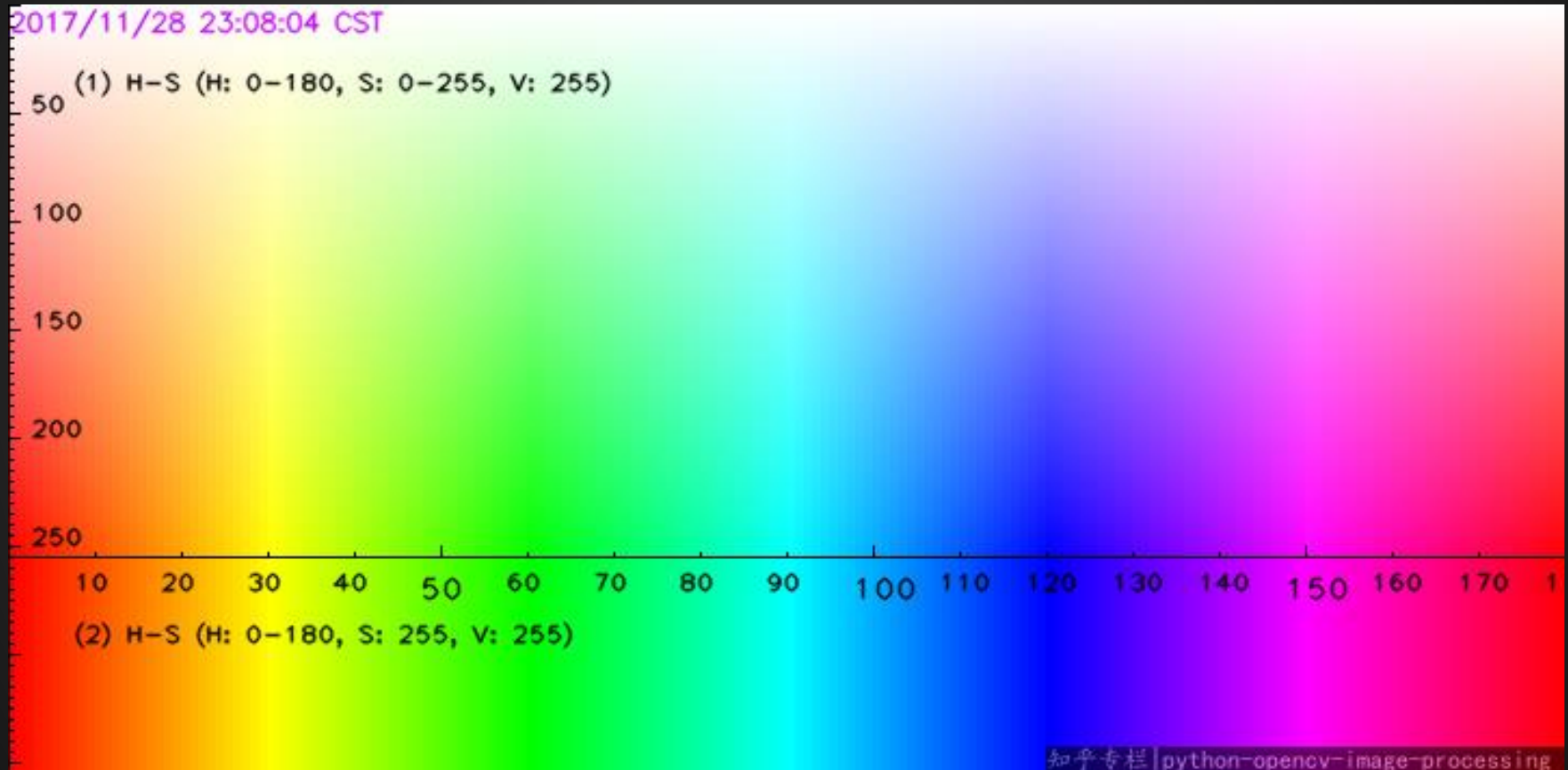
色空間の話

- HSV色空間
 - 色相(Hue)、彩度(Saturation)、明度(Value · Lightness · Brightness)の3要素で色を表現する方法
 - 円筒または円錐状の座標系に定義される



opencv-pythonにおけるHSV色空間マップ

- 横軸H, 縦軸S



カメラとロボットの協調

- ロボットの原点とカメラの原点が異なる
 - ロボットは土台中央, 画像は左上
 - 場合によっては軸のとり方も逆
- カメラの視野は鉛直下向きと限らない
 - ロボット座標の軸とカメラ座標の軸の関係を記述したいが厳密に合わせるのが難しい
 - カメラを真下に向けたつもりでも, 視線ベクトルが画像の中心を貫いているとは限らない
 - そもそも合っているかどうかをどう評価するか
 - ロボットの座標系をカメラ座標系に変換する行列を決定する
 - 同次変換行列による回転・並進行列を用いてカメラ座標系をロボット座標系に変換

ワークが置かれる平面の推定

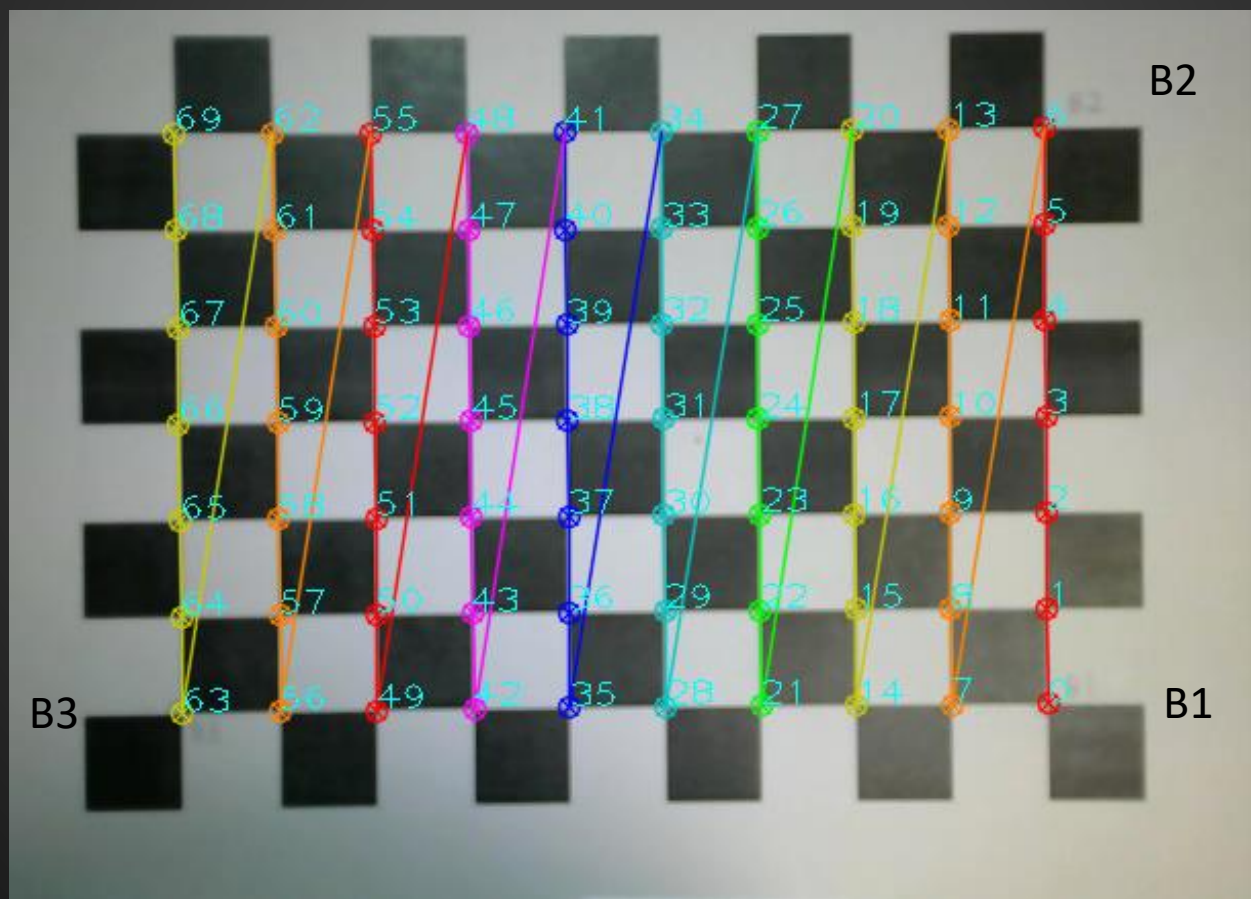
- 平面上3点で決定できる
 - 平面上に存在する3点決めるとはじめて一意に定義できる
- ワークを置く平面と画像平面の傾きを補正する行列を得ることで変換可能となる

手順

1. カメラとロボットを固定する
 - 位置関係が変わらないようにテープなどで固定する
2. ロボットの作業領域にキャリブレーションプレート（チェッカーボード）を固定する
3. カメラでキャリブレーションプレートを撮影する
 - 「(2) カメラからチェッカーボードを検出するプログラム」を参照
4. ロボットの手先でキャリブレーションプレート状の3点にティーチングを行い、その座標の値を記録する
 - 「(4) 画像処理（色抽出）およびロボットの制御を行うプログラム」を参照

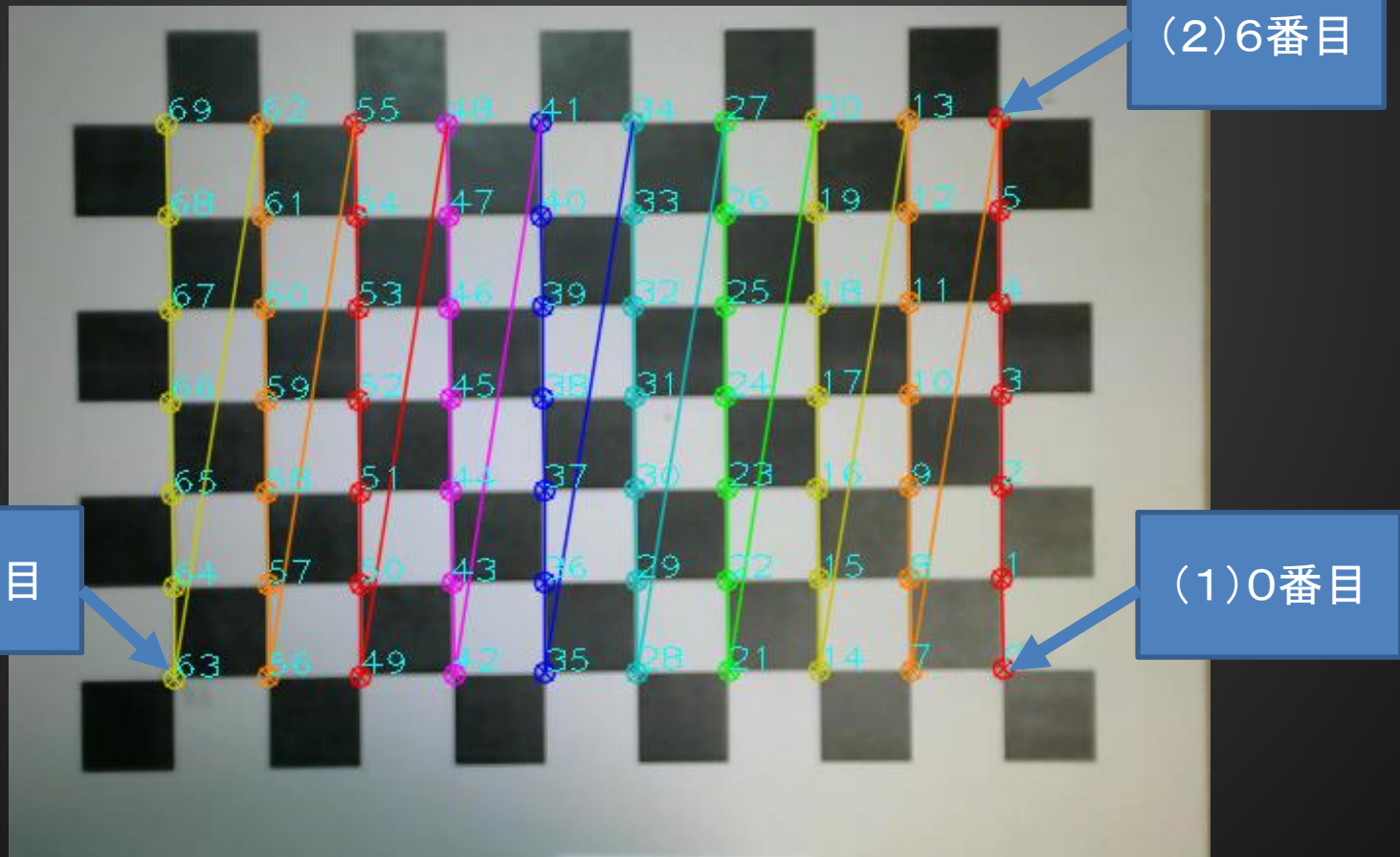
撮像した画像のチェッカー交点抽出

- 右下の点から左上にインデックスのついた点群が得られる



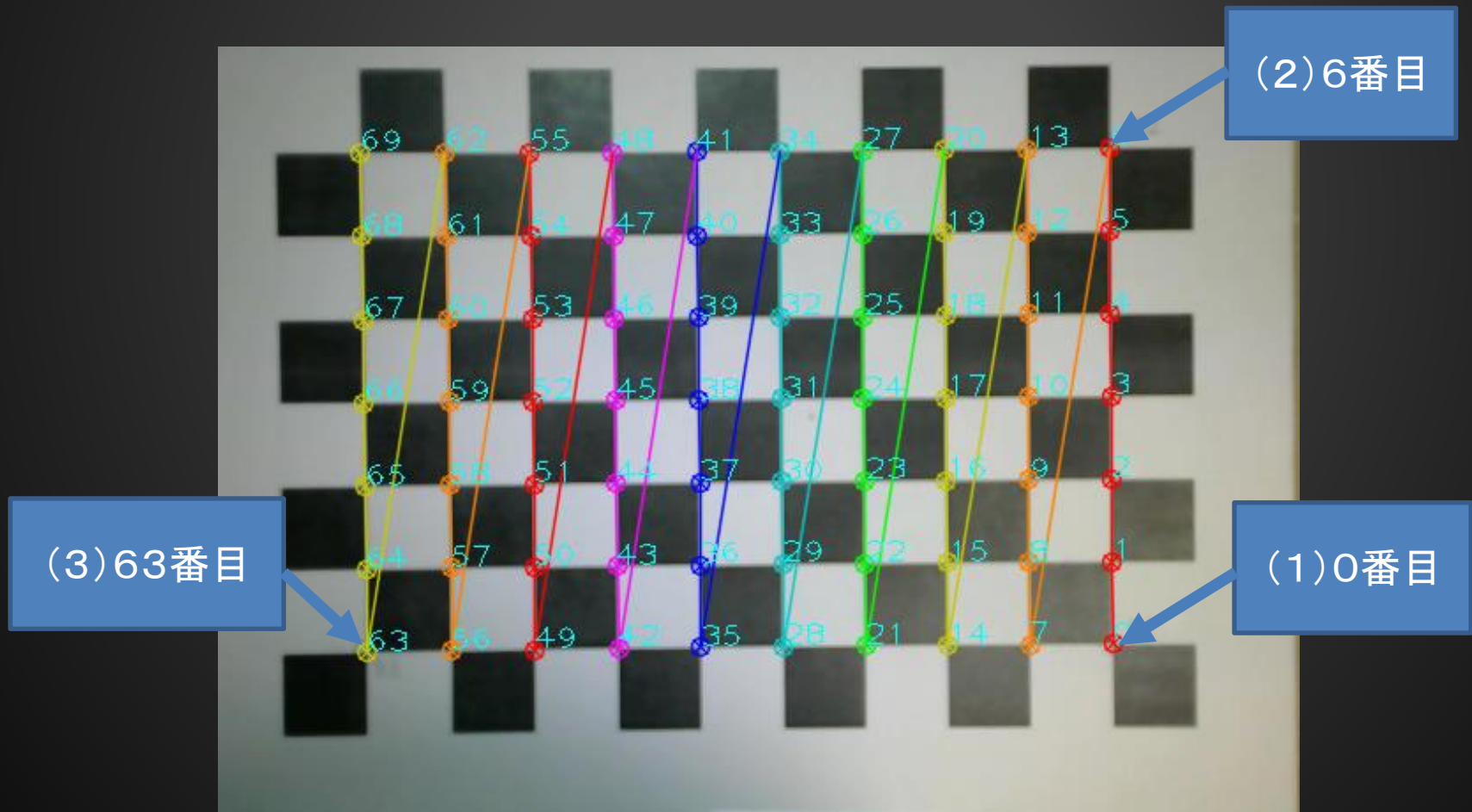
ロボットのティーチングによる 手先座標の取得

- 下記の3点に対しロボットをティーチングして手先を移動し, ロボットの手先座標 (XYZ) を記録する
 - teachPoints.csvにカンマ区切りで3点を記入する



ロボットのティーチングによる 手先座標の取得

- 次のスライドで述べるTCP/IP制御プログラム, または前回の実習で使ったシミュレータを用いる



ロボット-カメラ座標変換

- カメラ座標上の撮影した**3点**のチェッカーの位置にロボットのハンドを実際に動かすことでロボット座標との対応をとる
 - 2点だと1軸しか一意に決まらない。
 - 3点だと2軸が一意に決まる。2軸で外積とると1軸が決まる。
- 対応をとったカメラとロボットの座標を特異値分解することにより、カメラ座標上の点とロボット座標上の点で以下の関係が成り立つようにする

$$\begin{bmatrix} X_r \\ Y_r \\ Z_r \end{bmatrix} = \begin{bmatrix} R_{11} & R_{21} & R_{31} \\ R_{12} & R_{22} & R_{32} \\ R_{13} & R_{23} & R_{33} \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}$$

ロボット座標系 回転行列 カメラ座標系 並進行列

→ 特異値分解により回転行列と並進行列を決める

実習手順

1. カメラの画角を確認する

- 「setup.ipynb」の「(1) カメラから画像をキャプチャするプログラム」を実行するとカメラプレビューが見れるので、ロボットの天板が視野内に収まるよう調整する

2. カメラとロボット座標系の紐付けを行う

- チェッカーボードのB1~B3の座標にロボット手先を実際に移動させ、座標をteachpoints.csvに書き込む
- setup.pyの(2)を実行する

3. ロボットのTCP/IP通信による制御を行う

- サーバアプリケーションを立ち上げてsetup.ipunbの(3)を実行する

4. 色抽出の設定を行う

ロボットの設置（１）

- 完成図は下記の通りです



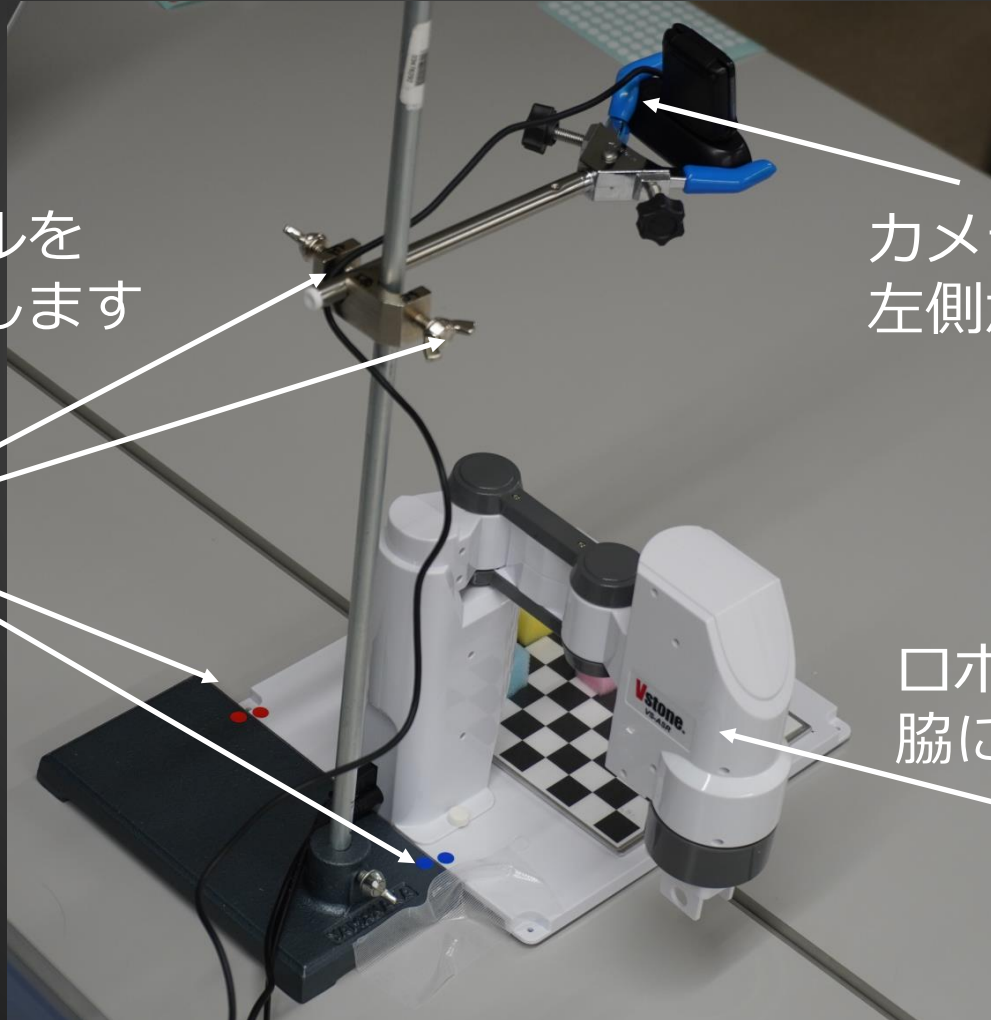
ロボットの設置（１）

- 設置のポイント

位置決めシールを
合わせて固定します

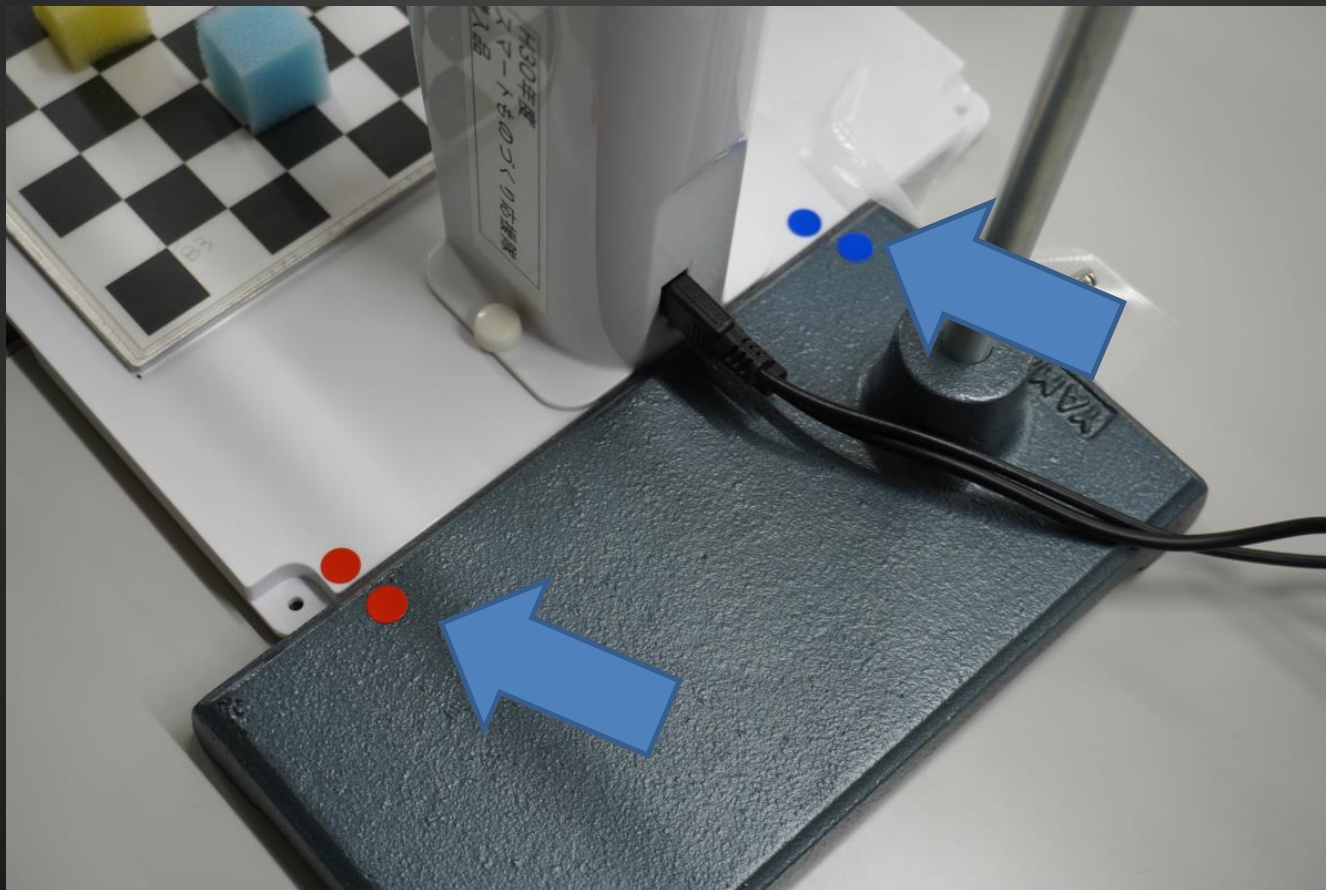
カメラのケーブルは
左側から出します

ロボットアームは
脇に待避します



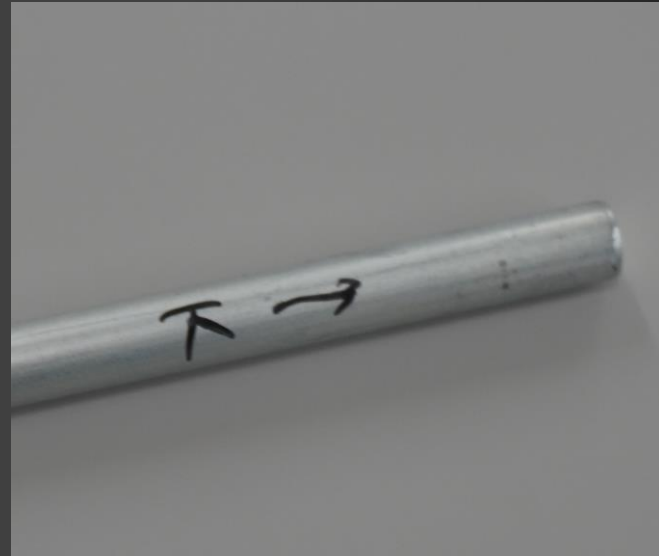
ロボットの設置（２）

- スタンドを設置します
 - スタンド側のシールとロボット天板側のシールが横並びになるように設置を行います



ロボットの設置（3）

- ポールの設置
 - ポールには上下の記載があります.
 - 「下」側をスタンドに差し込んで立てます



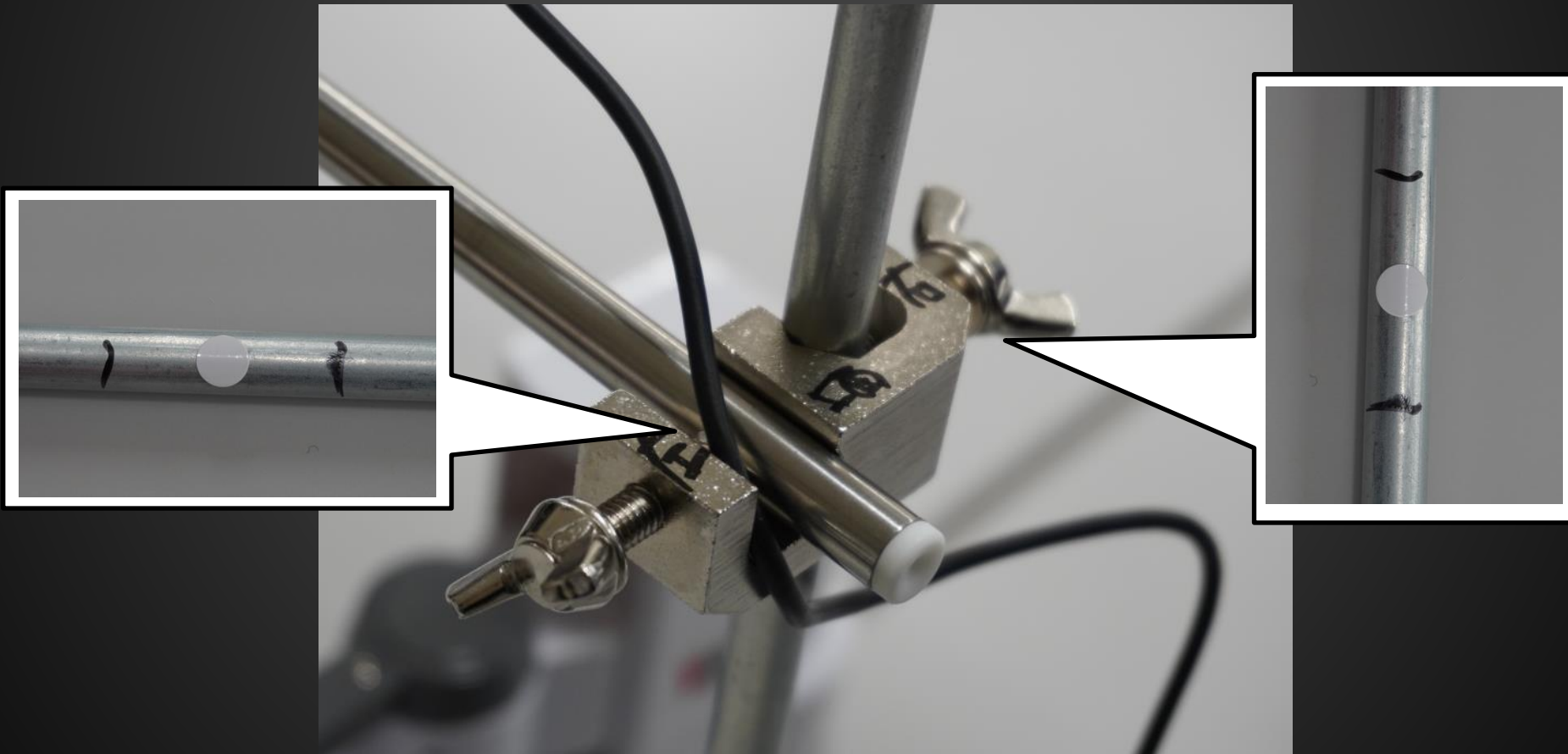
ロボットの設置（４）

- クランプの設置
 - クランプを文字が入った面が上になるように取り付けます
 - ポールとクランプに白い位置決めシール貼ってあるので，その部分を蝶ねじで固定します



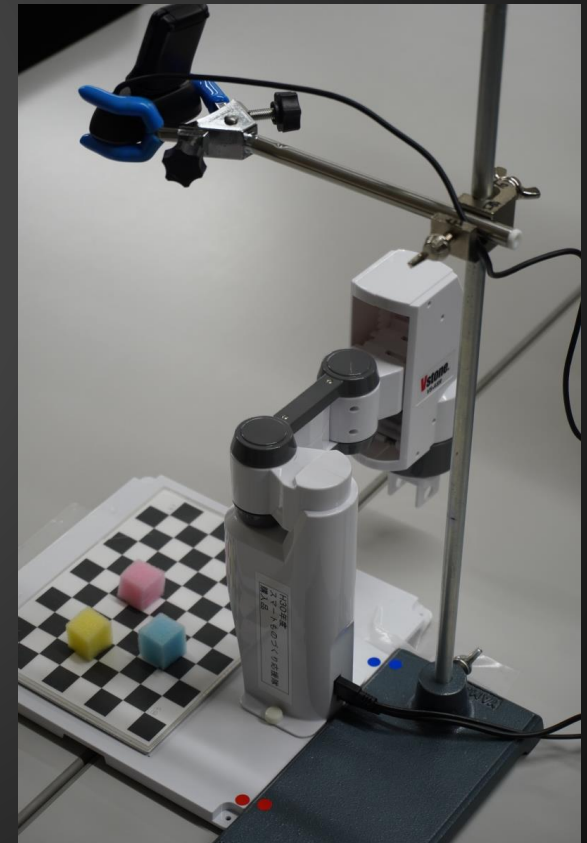
ロボットの設置（４）

- クランプの設置
 - クランプを文字が入った面が上になるように取り付けます
 - ポールとクランプに白い位置決めシール貼ってあるので、その部分を蝶ねじで固定します



ロボットの設置（5）

- カメラの設置
 - カメラのケーブルが左側から出るようにカメラをクランプの爪に固定します



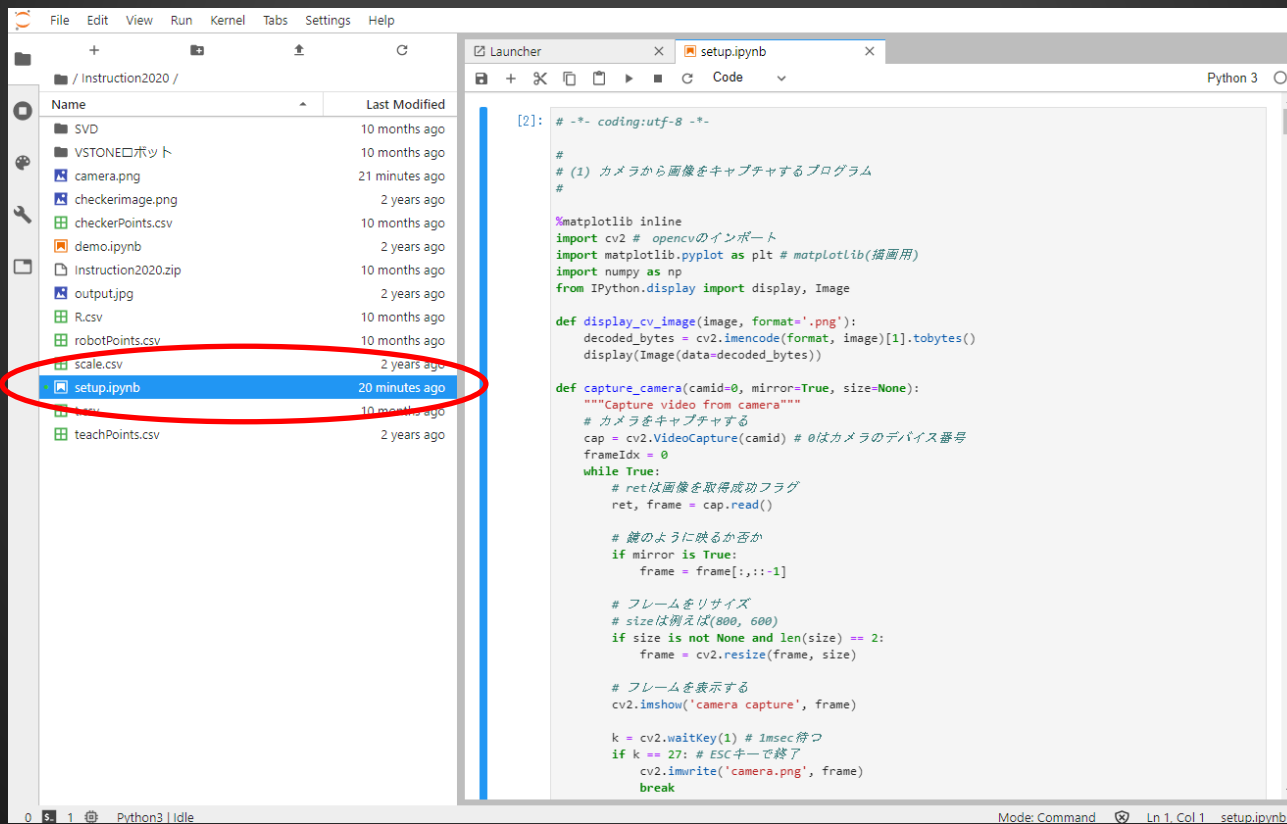
WinPython (Jupyter Lab) の起動

1. デスクトップの「Winpython」ショートカットをクリックします
2. 「scripts」ディレクトリに移動して「winjupyter_lab.bat」を実行します
3. コマンドプロンプトとブラウザが立ち上がり、ブラウザにコード編集画面が表示されます。

※Jupyter Lab利用中は、この時に起動したコマンドプロンプトウィンドウを閉じないでください

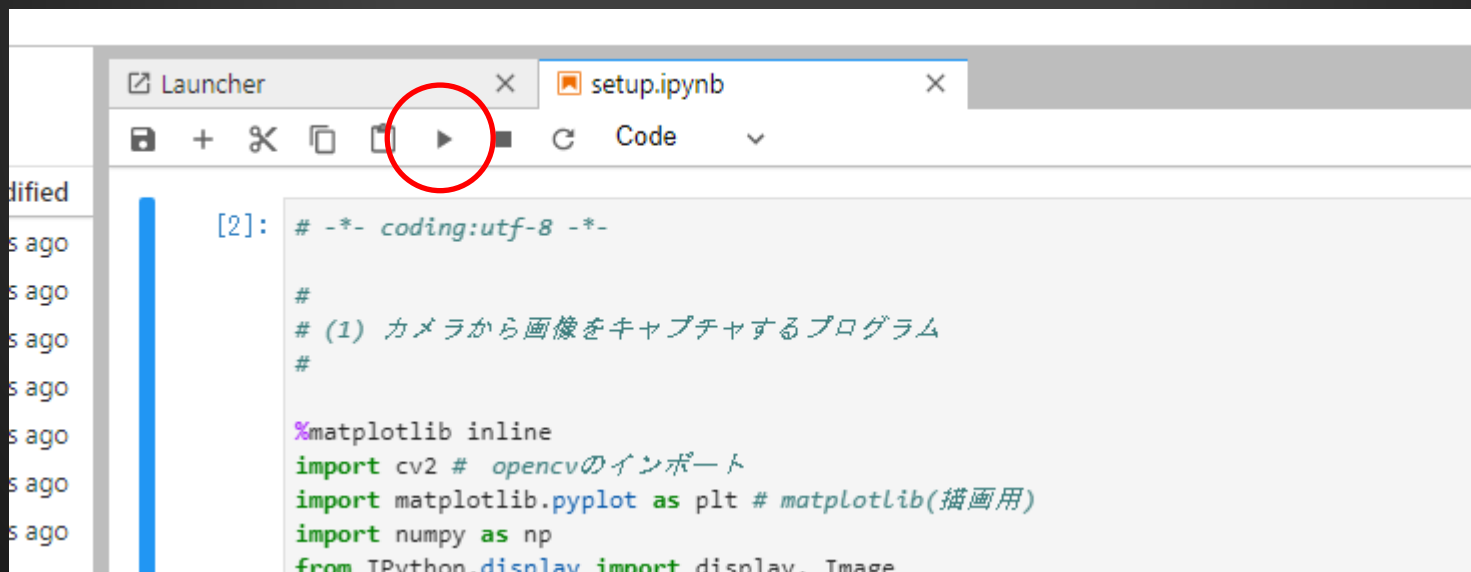
WinPython (Jupyter Lab) の起動

- 左ペインにファイルブラウザが表示されます
- 「setup.ipynb」をクリックするとエディタが開きます



Jupyter Labでのプログラム実行方法について

- 右ペインにコードブロックに別れたエディタが表示されます
1. 実行するコードブロックをフォーカスします（クリックします）
 2. エディタ上部の「▶」ボタンを押下すると、選択したコードブロックが実行されます
 - 「□」ボタンを押下すると実行中断できます

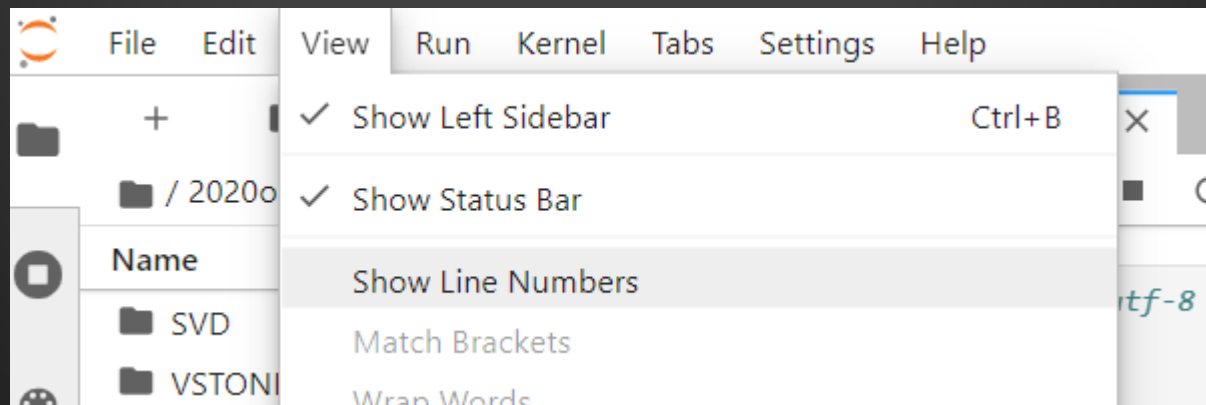


エディタ画面における行数表示方法について

- 下記の手順でソースコードに行数を降るようにします

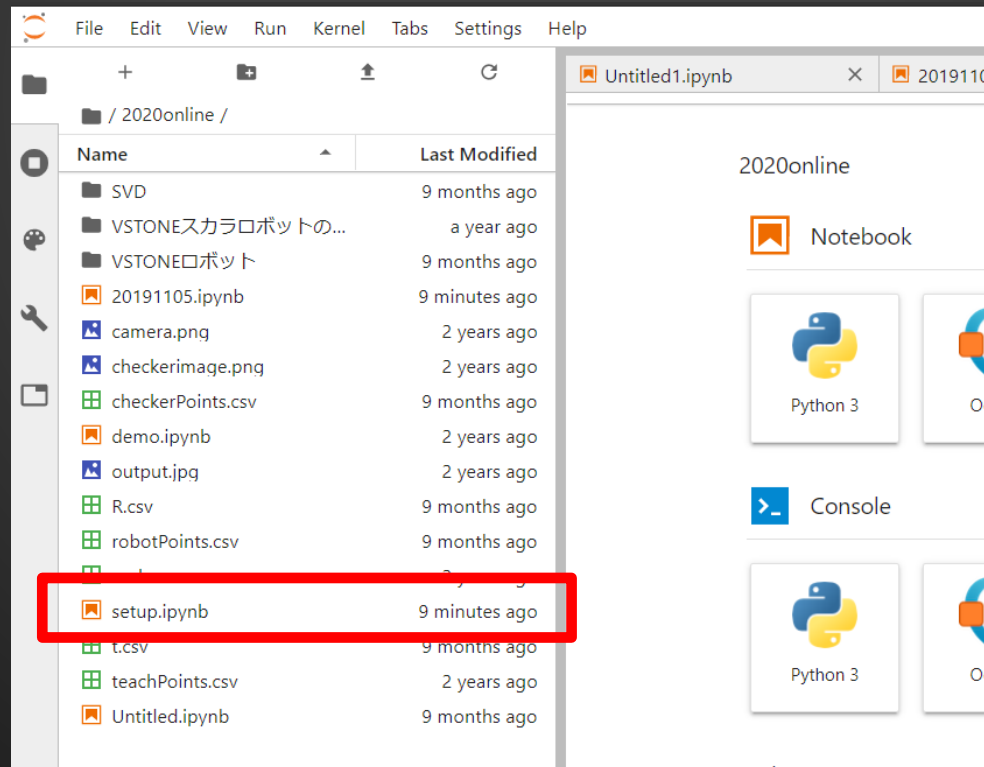
1. ソースコードを開きます

2. 「View」メニューから「Show Line Numbers」を選択します



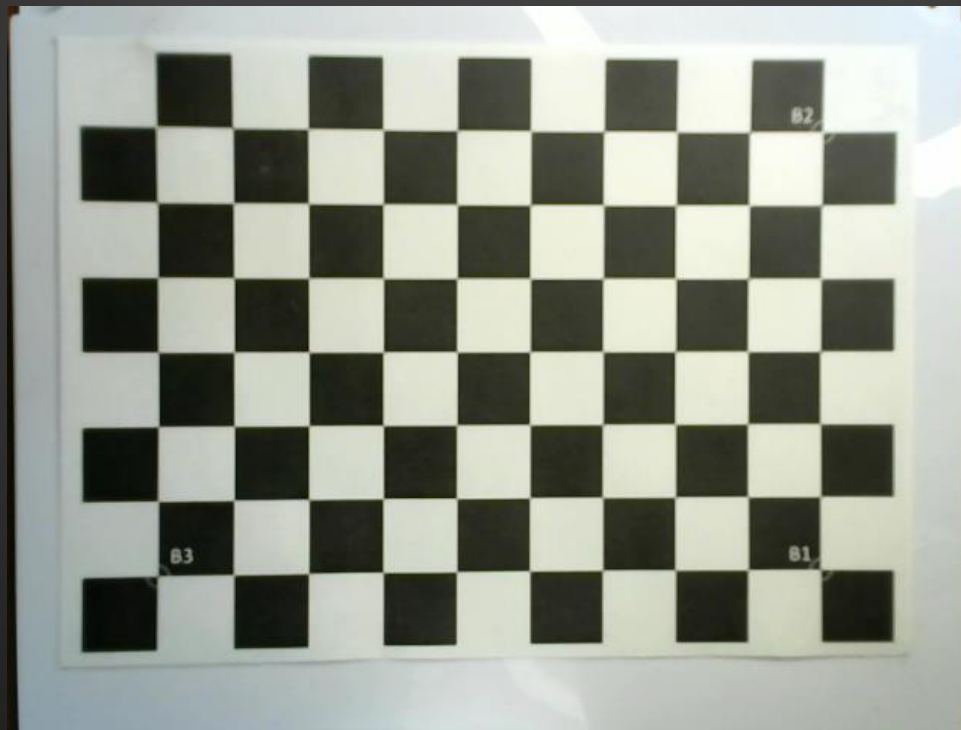
カメラの視野調整

- ブラウザのWinPythonの画面に移動します
1. ロボットのアームを天板外に退避します（カメラに映り込まないようにする）
 2. カメラをPCに接続します
 3. 「setup.ipynb」を開いて実行します
 4. 別のウィンドウでキャプチャ画像が表示されます
 5. カメラの視野角を調整します
 6. 「ESC」ボタンを押下すると画面が閉じます（※「X」で閉じても再度開きます）



カメラの視野調整

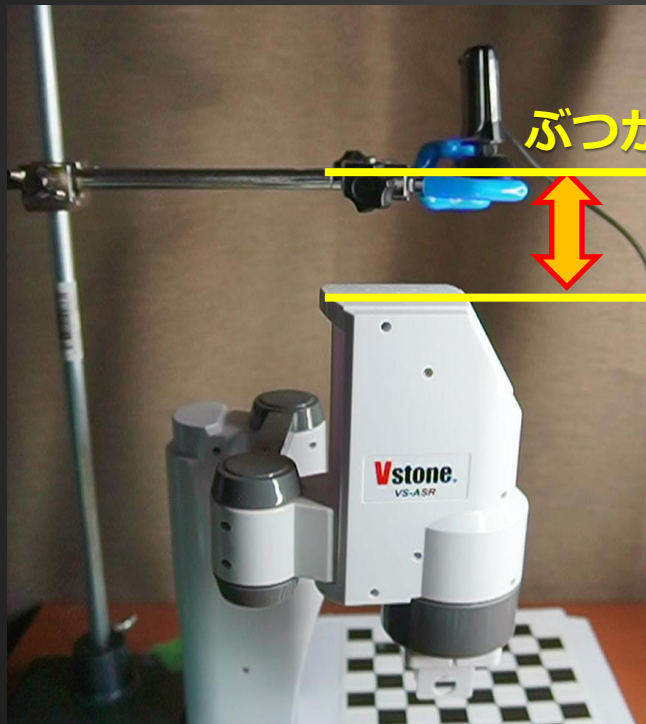
- 下記のような画像が撮影できるように下記の手段で視野の調整を行います
 - カメラスタンドの移動・高さ調整
 - カメラクランプの角度調整
 - ロボット本体の移動



カメラの視野調整

- 注意事項

- カメラクランプの高さは、ロボットの3軸目の可動範囲の外になる高さに設定してください



- ロボットの3軸目は動かしにくいので、右の画像のように2軸目を支えながら動かしてください

カメラの視野調整

- カメラが開かない場合

「setup.ipynb」 (1) の5 6行目

「capture_camera(camid=0, mirror=False)」

のcamidに適当な数字を入れてください

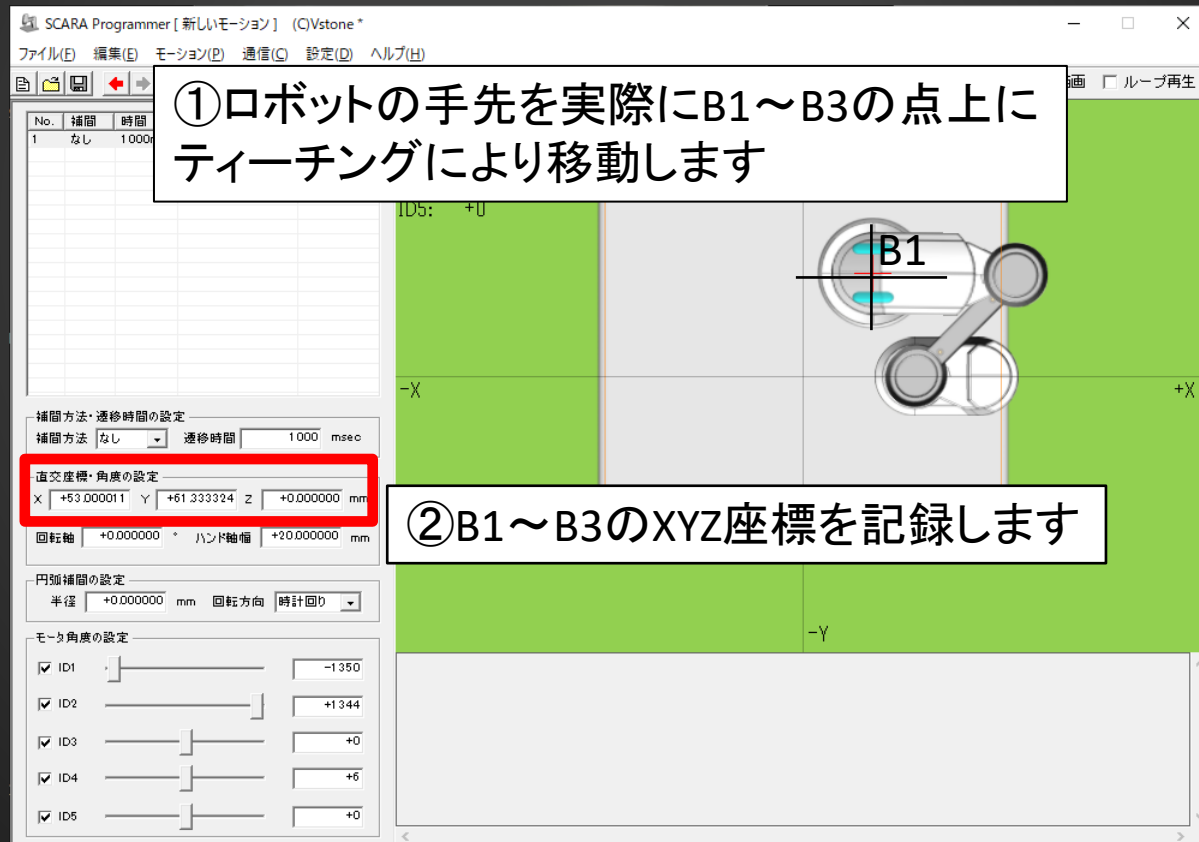
(0 オリジンでカメラの台数分だけIDが割り振られています)

以上でロボットの設置作業は終了です

以降スタンドとロボットの位置関係がズレないように
テープなどで固定します

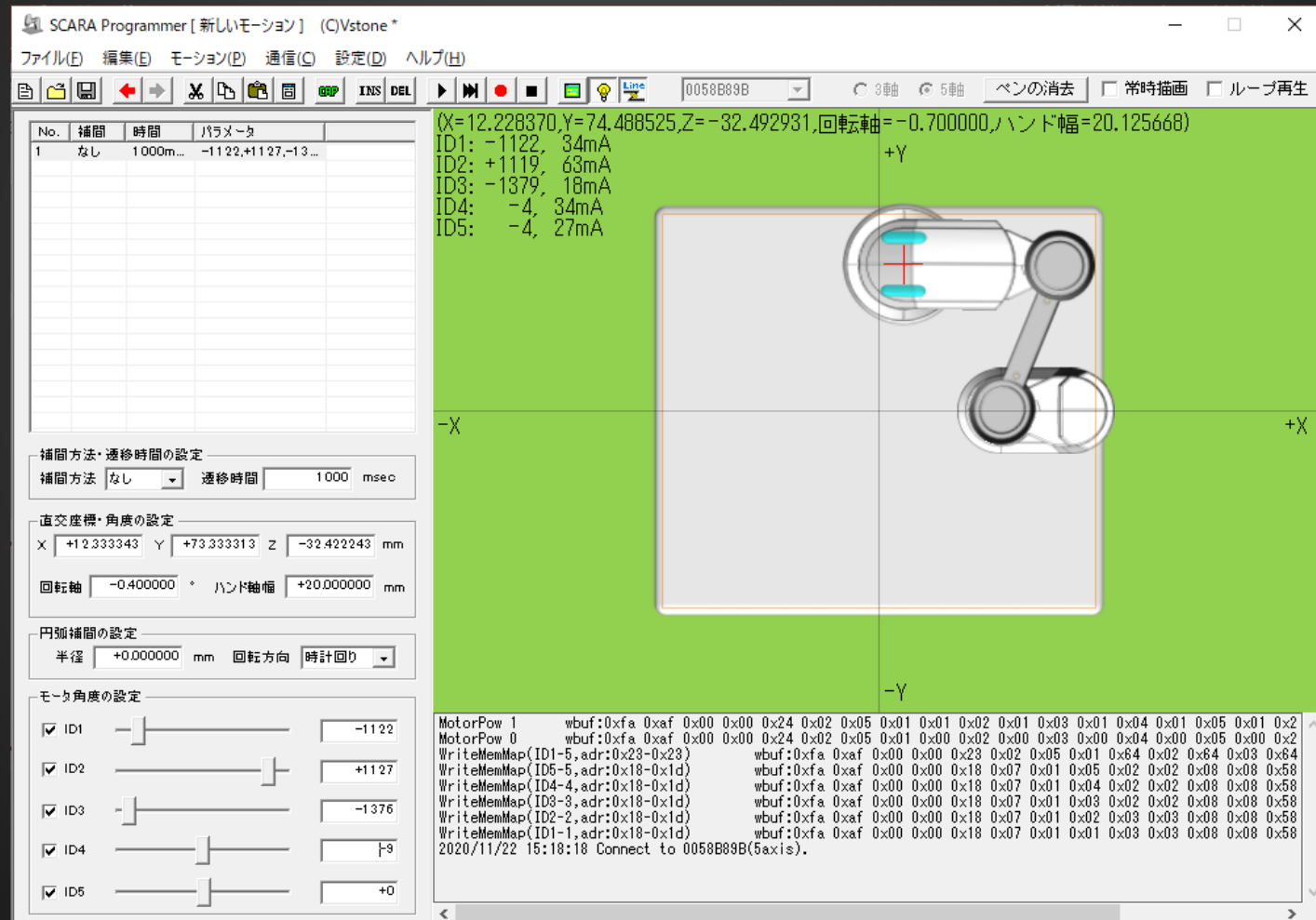
ロボット座標とカメラ座標の紐付け作業（１）

- チェッカーボード上の「B1, B2, B3」の３点にロボットの先端を移動して座標を記録します



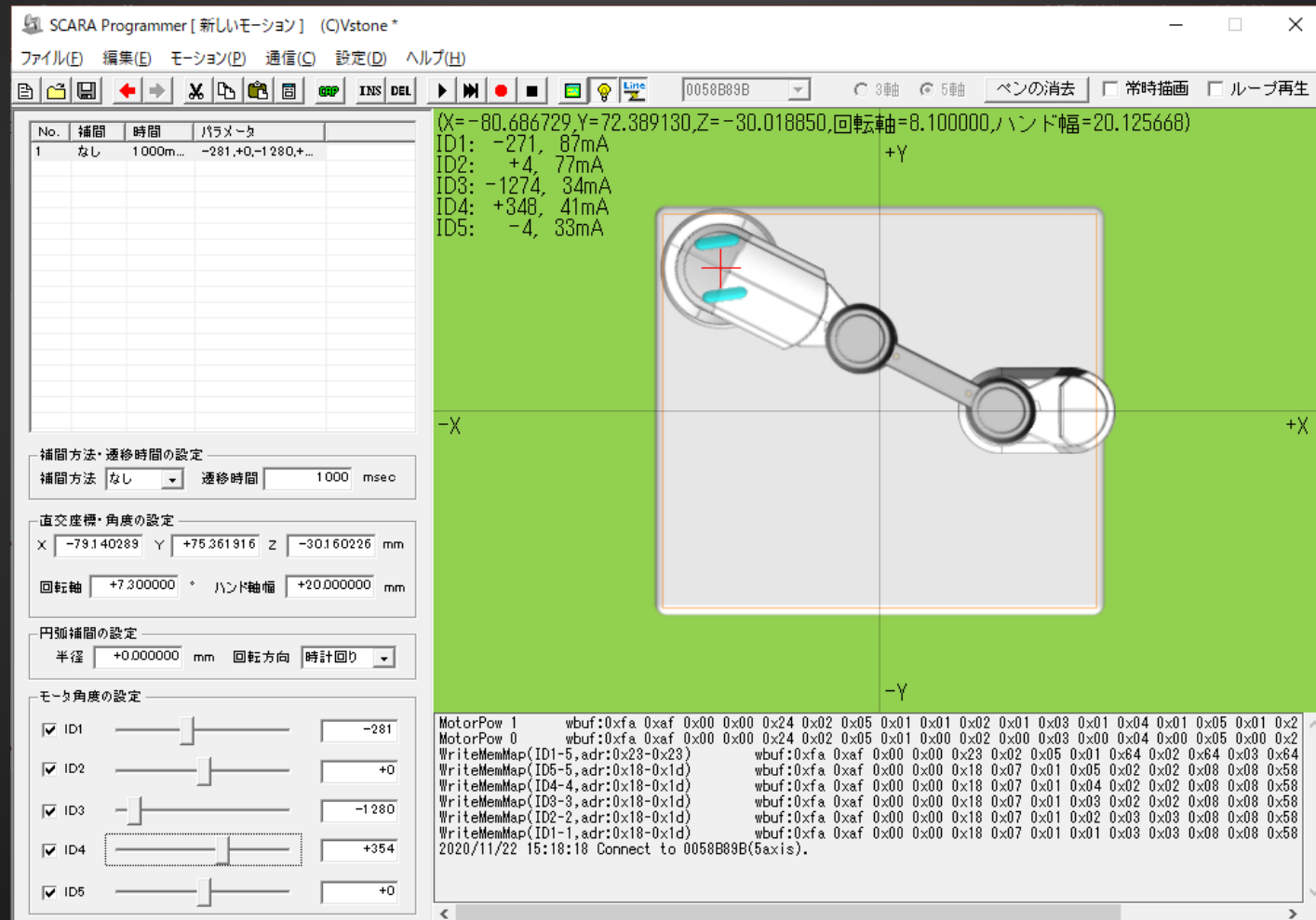
ロボット座標とカメラ座標の紐付け作業（１）

- B1



ロボット座標とカメラ座標の紐付け作業（１）

- B2



ロボット座標とカメラ座標の紐付け作業（１）

- B3

SCARA Programmer [新しいモーション] (C)Vstone *

ファイル(F) 編集(E) モーション(P) 通信(C) 設定(D) ヘルプ(H)

0058B89B 3軸 5軸 ペンの消去 常時描画 ループ再生

No.	補間	時間	パラメータ
1	なし	1000m...	+44,+1128,-1354...

補間方法・遷移時間の設定
補間方法 なし 遷移時間 1000 msec

直交座標・角度の設定
X +18.803600 Y -77.290833 Z -31.903864 mm
回転軸 +3.800000 ° ハンド軸幅 +20.000000 mm

円弧補間の設定
半径 +0.000000 mm 回転方向 時計回り

モータ角度の設定

<input checked="" type="checkbox"/> ID1	<input type="text" value="44"/>
<input checked="" type="checkbox"/> ID2	<input type="text" value="1128"/>
<input checked="" type="checkbox"/> ID3	<input type="text" value="-1354"/>
<input checked="" type="checkbox"/> ID4	<input type="text" value="-1134"/>
<input checked="" type="checkbox"/> ID5	<input type="text" value="+0"/>

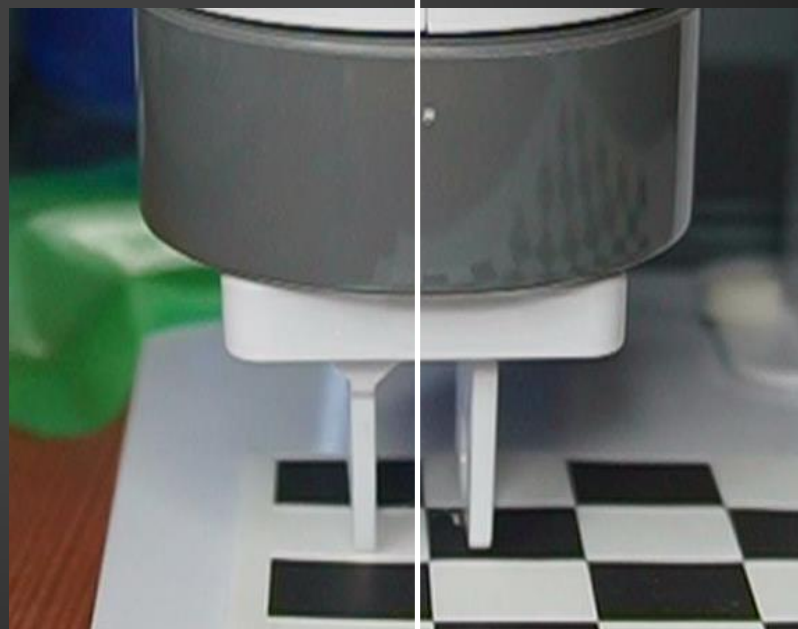
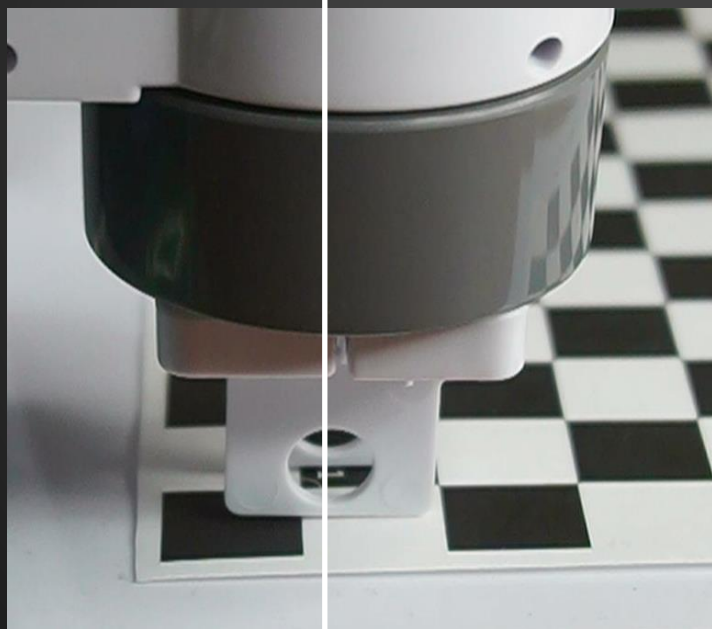
(X=17.335934,Y=-76.586716,Z=-32.257804,回転軸=2.200000,ハンド幅=20.125668)
ID1: +34, 84mA
ID2: +1127, 23mA
ID3: -1369, 60mA
ID4: -1139, 33mA
ID5: -4, 34mA

+Y
-X
+X
-Y

MotorPow 1 wbuf:0xfa 0xaf 0x00 0x00 0x24 0x02 0x05 0x01 0x01 0x02 0x01 0x03 0x01 0x04 0x01 0x05 0x01 0x2
MotorPow 0 wbuf:0xfa 0xaf 0x00 0x00 0x24 0x02 0x05 0x01 0x00 0x02 0x00 0x03 0x00 0x04 0x00 0x05 0x00 0x2
WriteMemMap(ID1-5,adr:0x28-0x23) wbuf:0xfa 0xaf 0x00 0x00 0x28 0x02 0x05 0x01 0x64 0x02 0x64 0x03 0x64
WriteMemMap(ID5-5,adr:0x18-0x1d) wbuf:0xfa 0xaf 0x00 0x00 0x18 0x07 0x01 0x05 0x02 0x02 0x08 0x08 0x58
WriteMemMap(ID4-4,adr:0x18-0x1d) wbuf:0xfa 0xaf 0x00 0x00 0x18 0x07 0x01 0x04 0x02 0x02 0x08 0x08 0x58
WriteMemMap(ID3-3,adr:0x18-0x1d) wbuf:0xfa 0xaf 0x00 0x00 0x18 0x07 0x01 0x03 0x02 0x02 0x08 0x08 0x58
WriteMemMap(ID2-2,adr:0x18-0x1d) wbuf:0xfa 0xaf 0x00 0x00 0x18 0x07 0x01 0x02 0x03 0x03 0x08 0x08 0x58
WriteMemMap(ID1-1,adr:0x18-0x1d) wbuf:0xfa 0xaf 0x00 0x00 0x18 0x07 0x01 0x01 0x03 0x03 0x08 0x08 0x58
2020/11/22 15:18:18 Connect to 0058B89B(5axis).

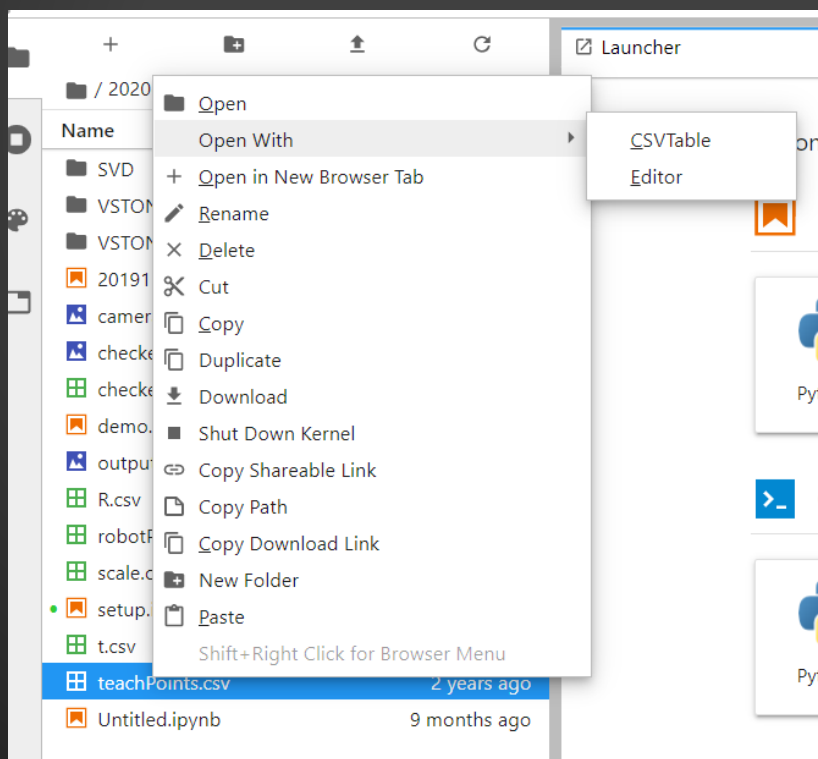
ロボット座標とカメラ座標の紐付け作業（2）

- ロボットハンドの先端は天板に可能な限り近づけます
- ロボットのJ5（5軸目）の回転軸が点上に来るようにロボットを移動します



ロボット座標とカメラ座標の紐付け作業（2）

- チェッカーボード上の「B1, B2, B3」の3点にロボットの手先を移動して座標を記録します
- WinPythonに移動し, 「teachPoints.csv」を右クリックから「Editor」で開きます



ロボット座標とカメラ座標の紐付け作業（3）

- チェッカーボード上の「B1, B2, B3」の3点にロボットの手先を移動して座標を記録します
- WinPythonに移動し, 「teachPoints.csv」を右クリックから「Editor」で開きます
- 開いたCSVファイルは下記の通り座標が記録されているので, 実際にロボットの手先を動かして得た座標に書き換え・保存を行います

teachPoints.csv	
1	+19.704960, +74.912987, -24.081056
2	+70.318939, -71.693878, -24.081056
3	-28.905252, +74.682541, -24.081056
4	

← B1x, B1y, B1z

← B2x, B2y, B2z

← B3x, B3y, B3z

ロボット座標とカメラ座標の紐付け作業（４）

1. ロボットのアームを天板上から退避します
 - カメラに写り込まないようにします
 - ロボット制御アプリケーションからロボットを制御して対比するか，モータOFF状態でダイレクトティーチにより動かしてください。
2. 「setup.ipynb」の（１）を実行してチェッカーボードの画像をカメラから取得します
3. 「setup.ipynb」の（２）を実行してロボット座標とカメラ座標の紐付けを行います
 - 下記の通り，チェッカーボードの交点が検出できれば成功です



ここからの作業

1. TCP/IPを用いたロボット制御アプリケーションの動作確認を行う
2. カメラで撮影した色抽出により把持対象物を画像から抽出する
3. 把持対象物を把持して所定の場所に移動する
 - 色によるワークの選り分け

TCP/IP通信によるロボットの制御

- 添付のアプリケーションを用いてTCP/IP通信によるロボットの制御が可能であるので併せて確認する
 - ロボットをUSB接続してから「VS-ASR_Console.exe」を起動する
 - 「VSTONEスカラロボットの制御サーバ」→「VSRのTCP通信制御ソフトウェア」
 - Windowsファイアウォールのアラートが表示されたら許可する
 - コマンド体系については添付資料を参照
 - 下記のソースコードにより検証可能（setup.ipynbの（3））

```
# -*- coding:utf-8 -*-
import socket

host = "127.0.0.1"
port = 5000

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect((host, port))
client.send(b"J,900,0,0,0,0,300¥n") #コマンドを送信する
response = client.recv(4096)
print(response)
```


TCP/IPによるロボットの制御（2）

- TCP制御サーバの起動

- コンソールが表示され、下記のメッセージが表示されれば制御サーバが正常起動しています

```
1個のデバイスが見つかりました
1番目のデバイスに接続中
1      -898      252      126
2      0         0       0
3     -13      255      243
4      2         0       2
5     -6      255      250
-----
server start
Listenを開始しました(0.0.0.0:5000).
```

- 「SCARA_Programmer」が起動中でロボットと通信している場合、同時に起動できません。

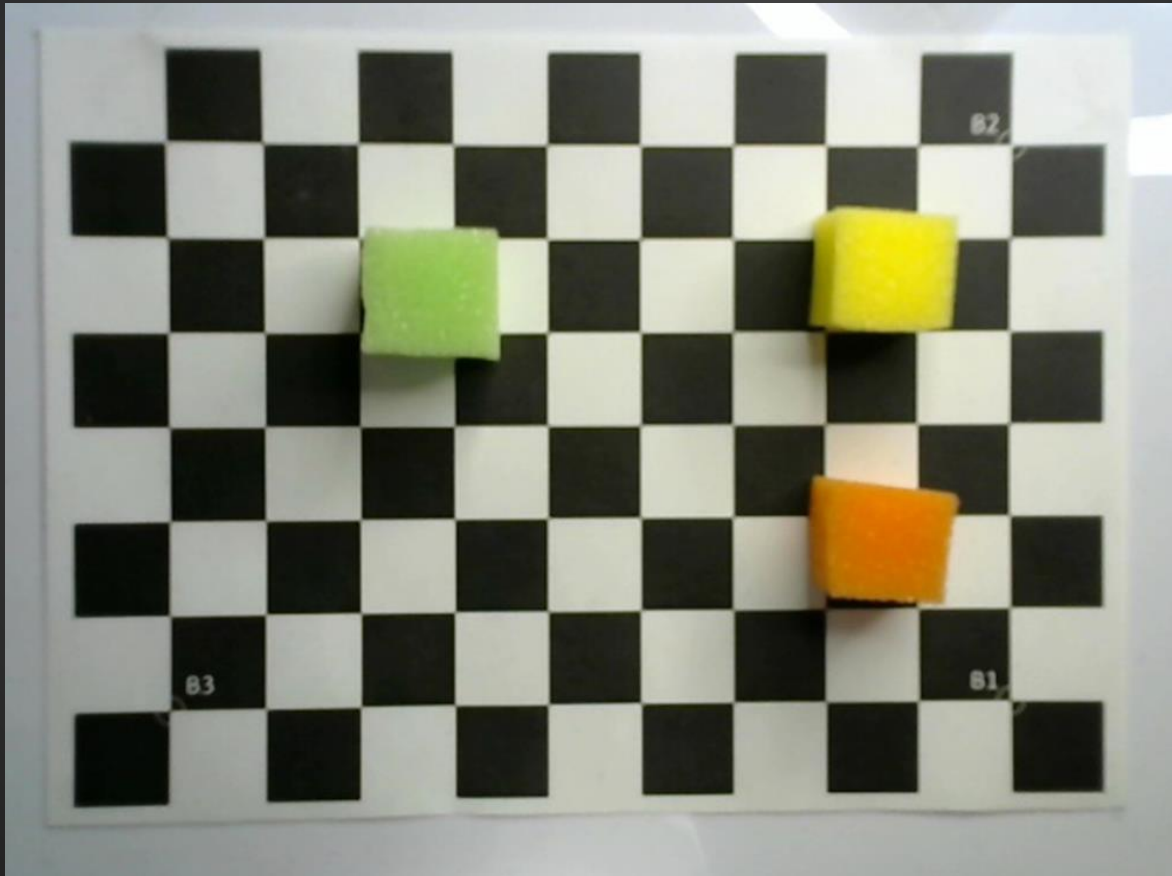
TCP/IPによるロボットの制御（3）

- 「setup.ipynb」を開き，「(3) ロボットのリモート制御を行うプログラム」を実行する
 - ロボットが動作すれば正常起動済み
 - エラーが返る場合，TCP制御ソフトが起動していることを確認する.

```
# -*- coding:utf-8 -*-  
  
#  
# (3) ロボットのリモート制御を行うプログラム  
#  
  
import socket  
  
host = "127.0.0.1"  
port = 5000  
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
client.connect((host, port)) #これでサーバーに接続します  
client.send(b"J,900,0,0,0,0,300¥n")  
response = client.recv(4096) #適当な2の累乗にします(大きすぎるとダメ)  
  
print(response)
```

色抽出による物体認識（１）

- 3色のワークの色抽出を行う



色抽出による物体認識（2）

1. 「20201122.ipynb」を開く
2. L.73 に3色のHSV上下限が記述されている

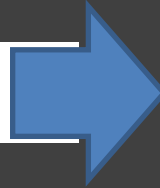
```
70 def extractcolorHSV(img):  
71     # 取得する色の範囲を指定する  
72     lower_blue = np.array([ 39, 92, 104]) #緑HSV下限  
73     upper_blue = np.array([ 71, 255, 255]) #緑HSV上限  
74  
75     lower_red = np.array([ 7, 216, 153]) #橙HSV下限  
76     upper_red = np.array([ 23, 255, 255]) #橙HSV上限  
77  
78     lower_yellow = np.array([ 22, 156, 204]) #黄HSV下限  
79     upper_yellow = np.array([ 50, 255, 255]) #黄HSV上限
```

3. コードブロックを実行して物体検出する
 - ロボットも動くので注意する
 - カメラが開かない場合, L.206を確認する
 - `capture_camera(camid=X, mirror=False)`

色抽出による物体認識（3）

- ロボットを動かしたくない場合, L.189を修正する
 - 行頭の「#」を消す

```
189 def RobotRef(command):  
190     #return
```



```
189 def RobotRef(command):  
190     return
```

パレタイジングの実施

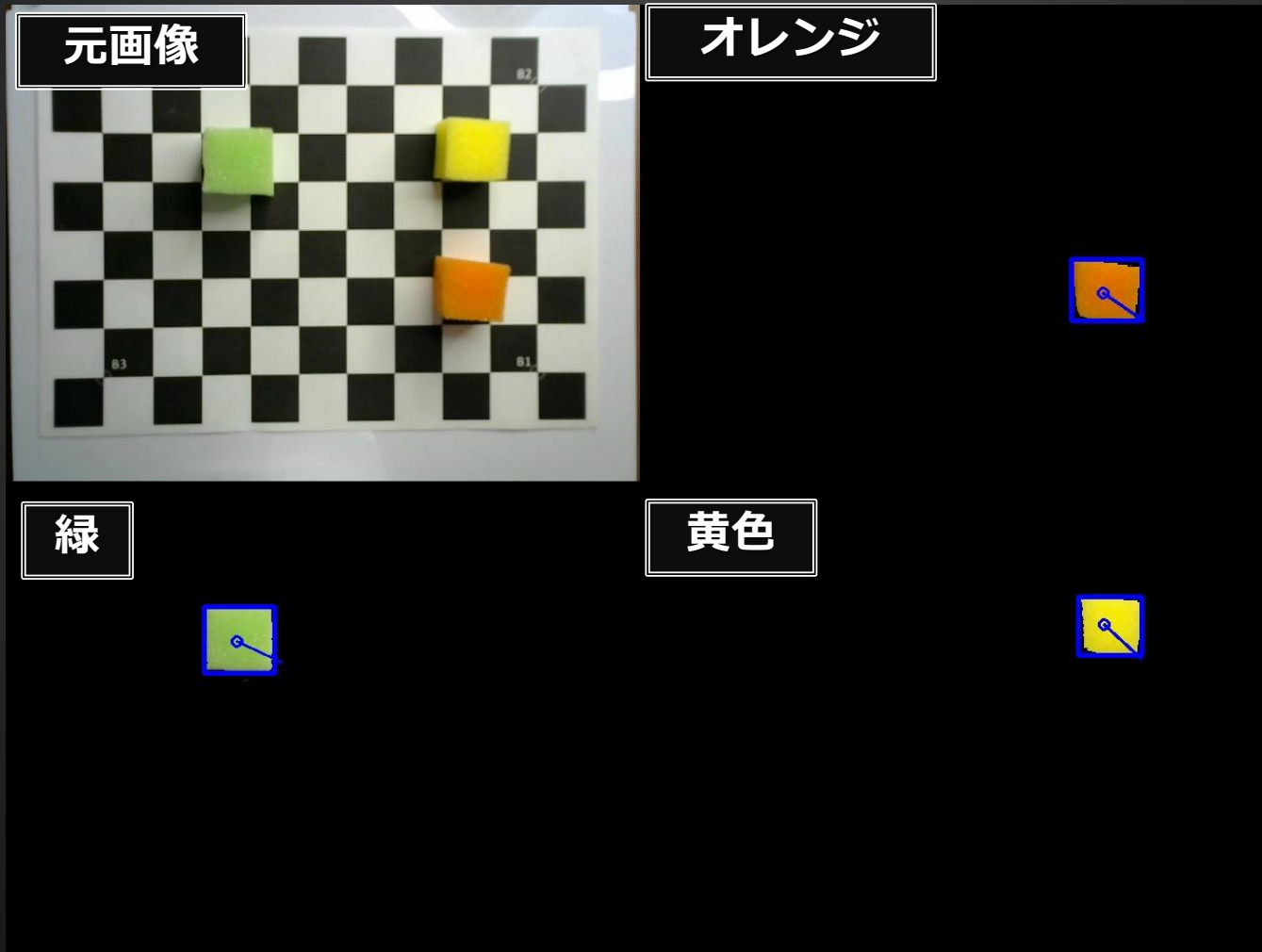
1. ロボットのTCP制御アプリケーションを起動します
 2. 「20191105.ipynb」を開き, (4) のコードブロックを実行します
- 動かない場合のトラブルシューティング
 - L.202が下記の通りコメントアウトしてあることを確認します（行頭「#」でコメントアウト状態）

```
200  
201 def RobotRef(command):  
202     #return  
203
```

- カメラとPCの接続を確認します
- 色抽出により, 把持位置が検出されていることを確認します
- TCP制御アプリケーションの起動状態を確認する

パレタイジングの実施

- 画像認識結果が表示され、ロボットが動作します



ロボット実習のまとめ

- ティーチング作業を実習した
 - ロボット単体が行える作業はこの程度
 - 付帯装置がないと有用な作業はできない
- 画像処理
 - 現場でも役に立つ頑強な色抽出方法について実習した
 - RGBよりもHSV表色系のほうが色抽出は容易
 - 照明環境により色の見えかたが異なるため、照明環境の整備は画像処理の適用においては重要な問題である
- カメラを視覚としたロボットの活用
 - ロボットとセンサ類を組み合わせることではじめて自動機として役に立つ
 - 知らなければならぬ要素技術が多い
 - SIerとしてロボットシステムを構築する場合…
 - 上流工程では実装の必要がない場合も多いが、要素技術に関する知識は必要。
 - どのように何を利用して実現するか。誰が／どこができるのか
 - 結局実装を知らないと工程引けない→値付けもできない
 - 一気通貫で立ち上げ、操業を行う場合には幅広い知識が必要
 - 自社で得意な分野はきちんとかなす。できないことは外にお願いするのも大事
 - できる人を捕まえる能力やネットワークも重要
 - 予算とマンパワーのトレードオフ