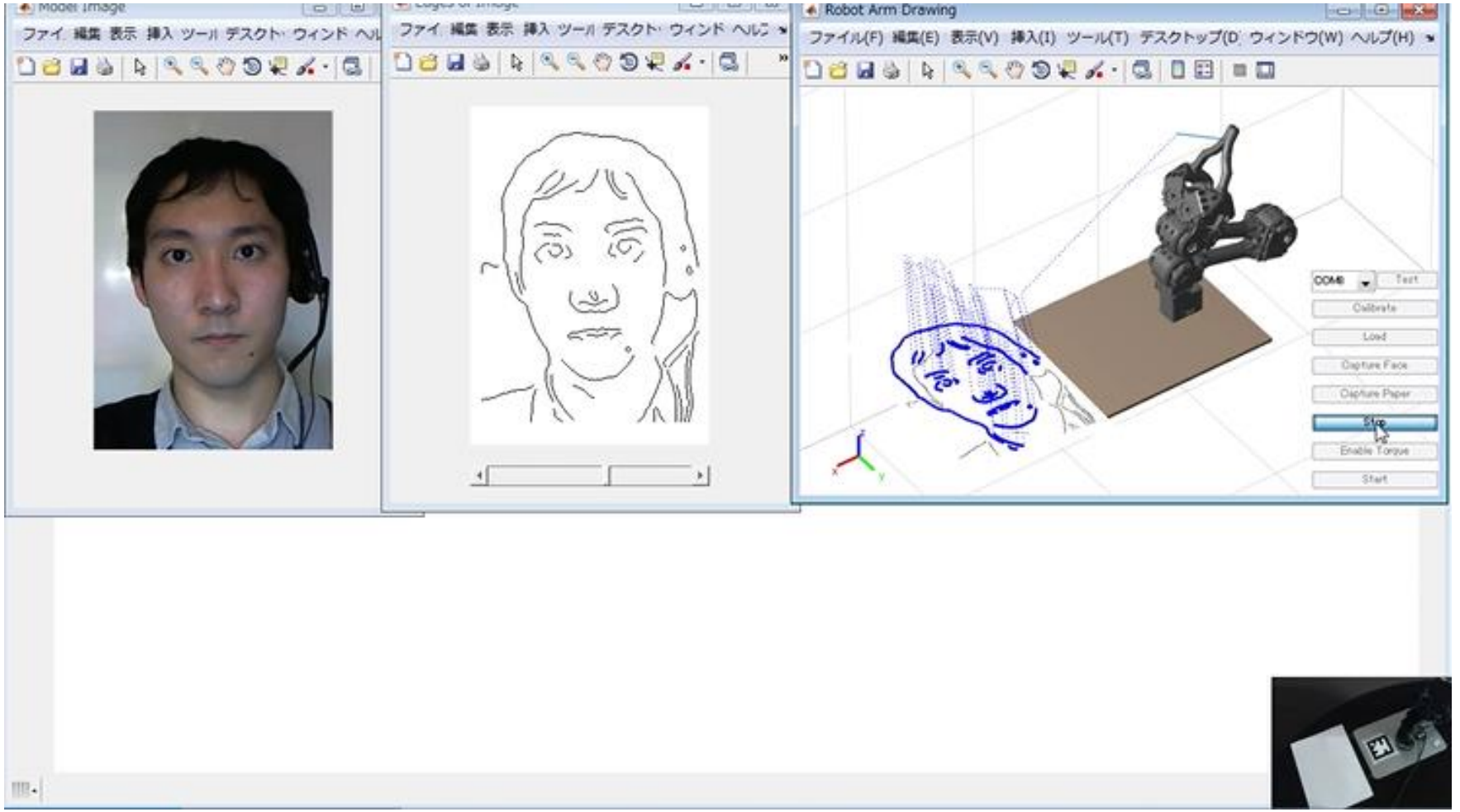
似顔絵ロボットのアルゴリズム設計
～顔認識とロボットアームの経路計画～

MathWorks Japan

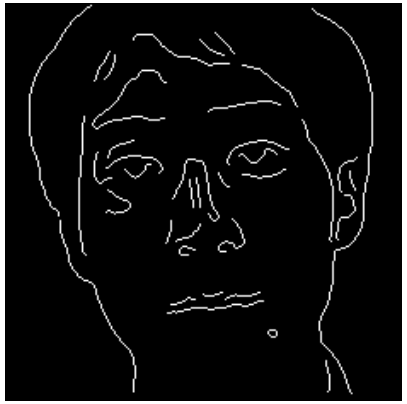
実機デモ:ロボットアームによる似顔絵作成



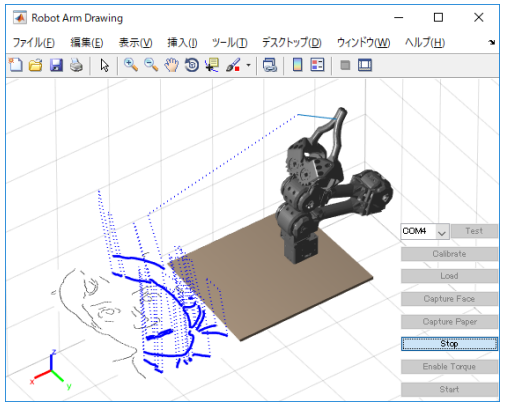
ロボットアームによる似顔絵作成：https://jp.mathworks.com/videos/portrait-drawing-robot-manipulator-1521840275440.html

デモの流れ

画像の取り込み



画像処理

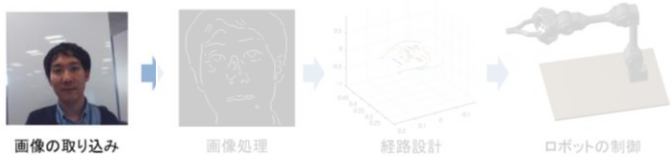


経路計画

実機制御

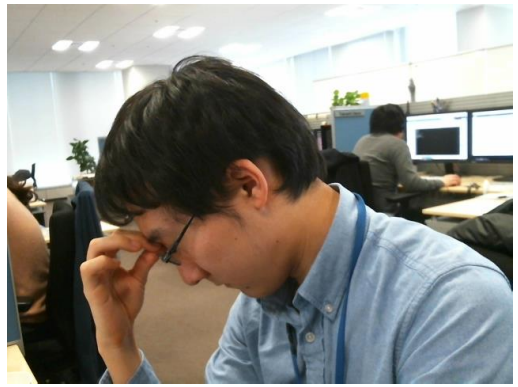
- キャリブレーション機能
- アプリケーション開発

1. 画像の取り込み（Webカメラ or 画像ファイル）



- Webカメラ：「MATLAB Support Package for USB Webcams」
  - わずか2行でWebカメラの画像を取り込み可能

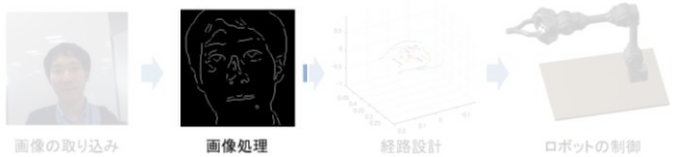
```
>> mycam = webcam(2);
>> I = mycam.snapshot;
```



- 画像ファイル：uigetfile + imread

```
>> [filename,pathname] = uigetfile(...
>>          {'*.bmp;*.png;*.jpg',…
>>          'Image Files (*.bmp,*.png,*.jpg)'},...
>>          'Select an Image File');
>> if filename == 0 %キャンセルに注意
>>   return
>> end
>> I = imread(fullfile(pathname,filename));
```

2. 画像処理①　顔の検出、切り出し

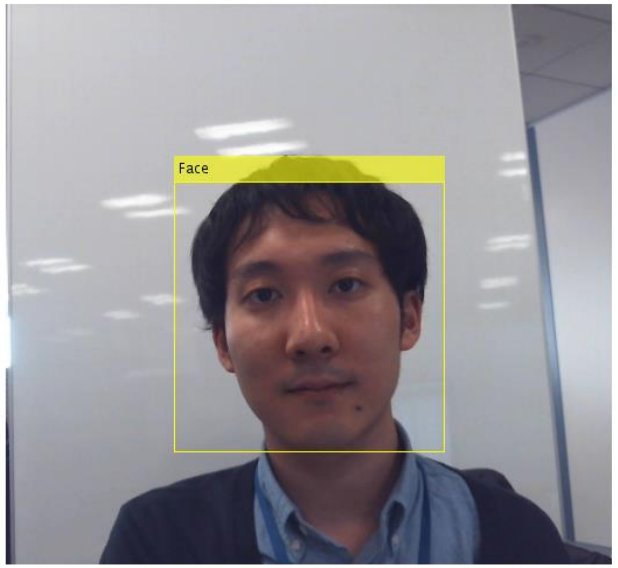


- 顔の検出
  - 「Computer Vision System Toolbox」のカスケード検出器を使用

```
>> faceDetector = vision.CascadeObjectDetector;
>> bboxes = step(faceDetector, I);
```

・向きが一定のもの
・色に依存しないもの

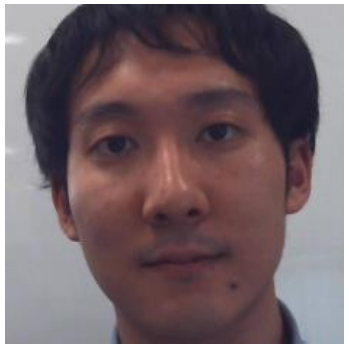
⇒ 顔検出が可能！



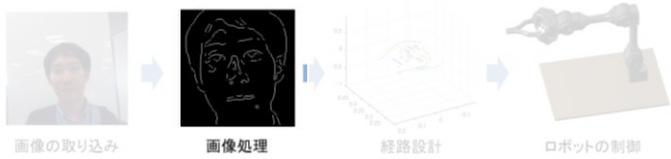
- 顔の切り出し
  - 検出結果をそのまま使用可能

```
>> I = imcrop(I,bboxes(1,:))
```

（実際はA4サイズと同じ比率で切り出す）



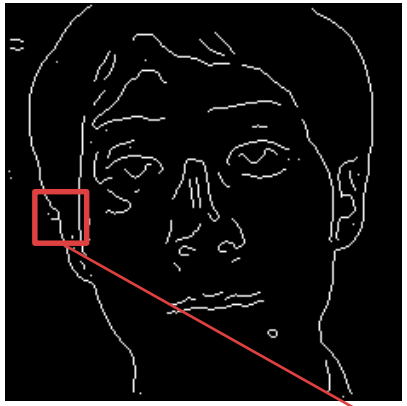
2. 画像処理②　線画の作成



エッジ検出

```
>> edge(rgb2gray(I),'Canny');
```

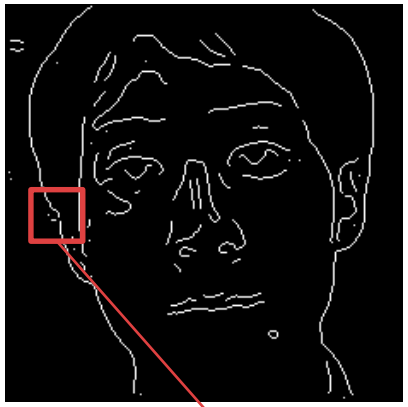
スケルトン化、
細かい枝木の除去



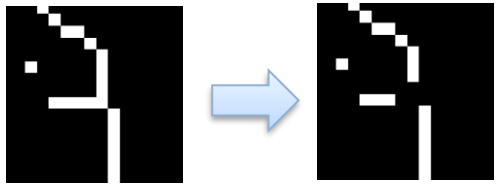
```
>> bwmorph(BW,'skel',Inf);
>> bwmorph(BW2,'spur',3);
```

1ピクセル幅にスケルトン化

枝分かれ部分の切断



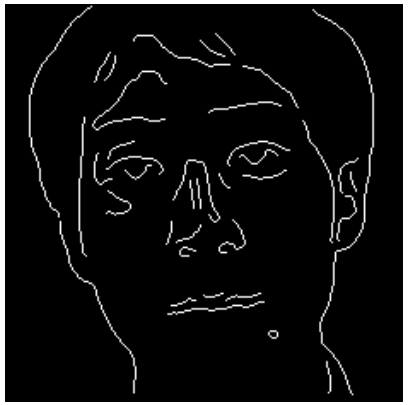
```
>> bwmorph(BW3,'branch',1);
```



一筆書き
できない

一筆書き×3

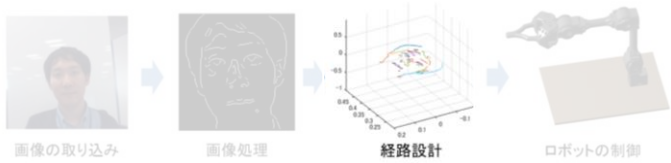
オープン処理
(小さい領域除去)



```
>> bwareaopen(BW3,10);
```

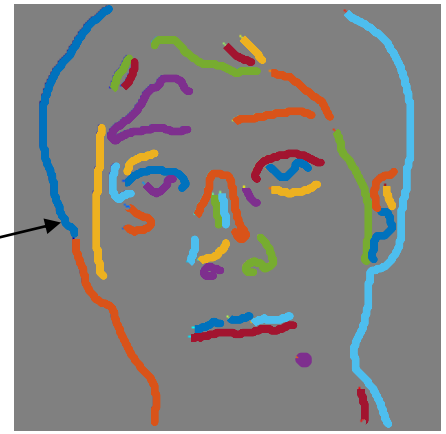
完成！

3. 経路設計①　画像から経路への計算



- 画像を経路に変換:
  Image Processing Toolboxの「**bwboundaries**」
  - オブジェクト（線）毎に境界をトレース

```
>> [B,L] = bwboundaries(BWseg,'noholes');
```



plot

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

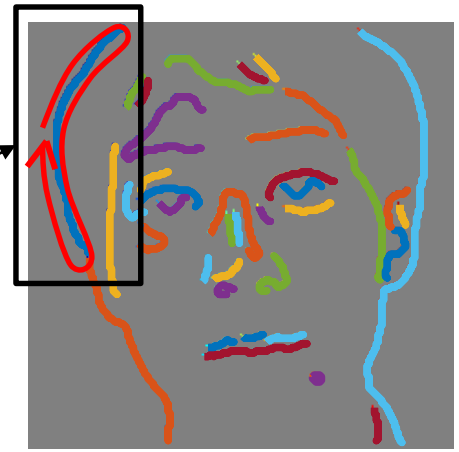
画像データ

$$\begin{bmatrix} 2 & 1 \\ 3 & 2 \\ 2 & 3 \\ 1 & 2 \\ 2 & 1 \end{bmatrix}$$

経路

インデックス配列

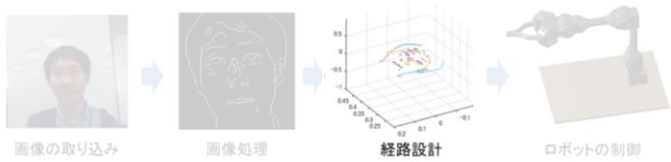
- 問題点
  - トレース開始点が端点ではない
  - 線上を往復してしまう



現実

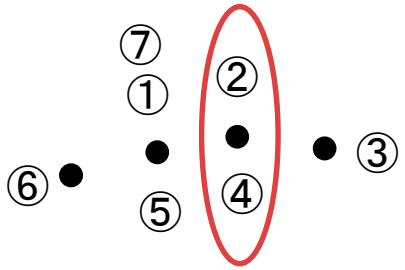
理想

3. 経路設計② 経路の整形



- 整形方法
  - 端点のインデックスを導出

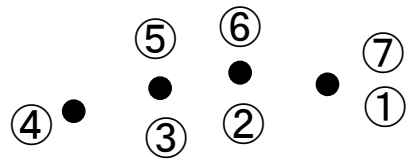
```
>> edgeind = find(all(circshift(boundary,1)==circshift(boundary,-1),2),1);
```



端点の前後は同じ座標

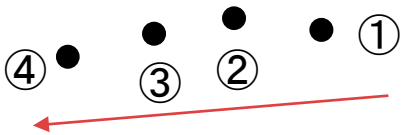
  - 端点があれば、それが開始点にくるように並び替え

```
>> if ~isempty(edgeind)
>>         boundary = circshift(boundary,-edgeind+1);
```

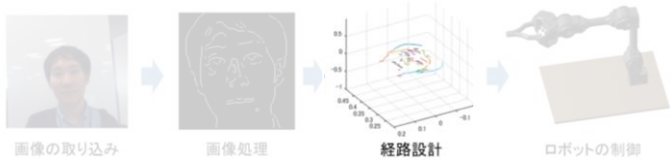


  - 半分のみを抽出

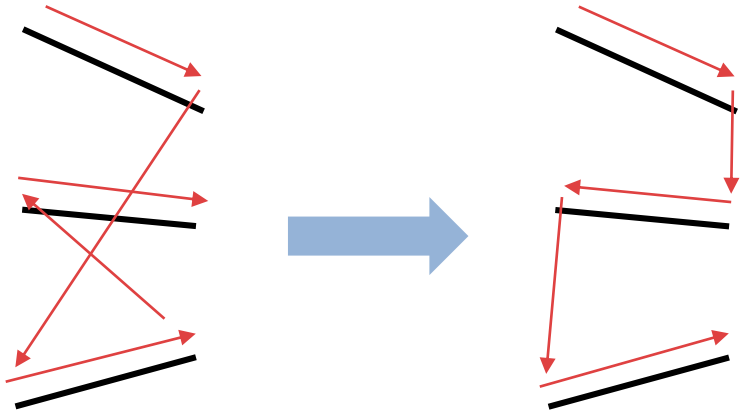
```
>>         boundary = boundary(1:ceil(end/2),:);
>> end
```



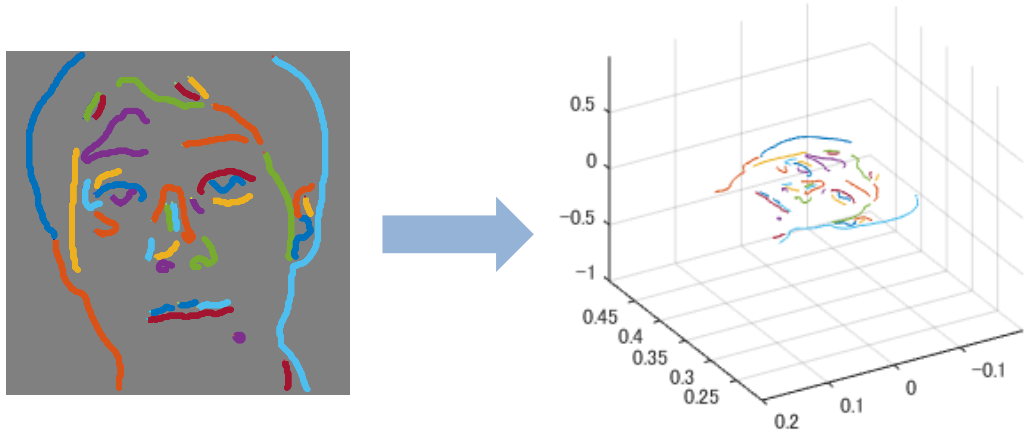
3. 経路設計③　並び替え、3次元経路への変換



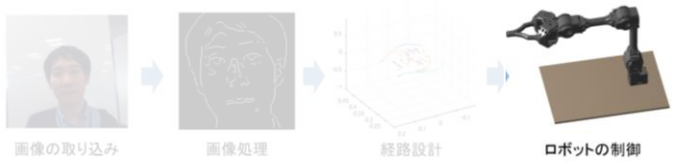
- 並び替え
  - デフォルト：x座標が小さい順
    → 近いオブジェクトをたどるように並び替え



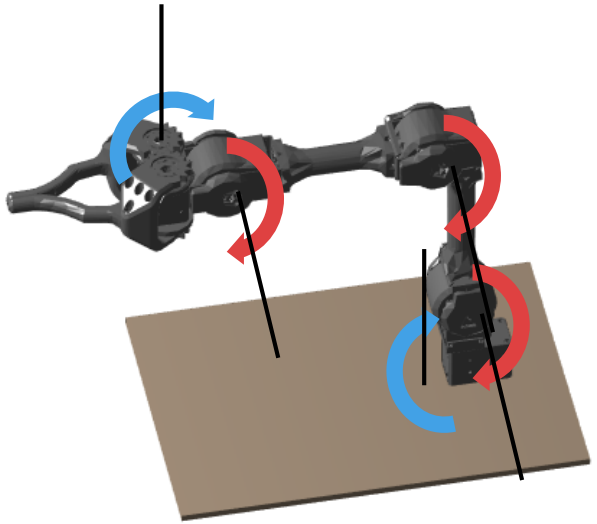
- 3次元経路への変換
  - 画像上の経路から、ペン先が動く3次元座標に変換
  - 紙の位置と1ピクセルの長さから計算（高さは0と仮定）



4. ロボットの制御①　概要



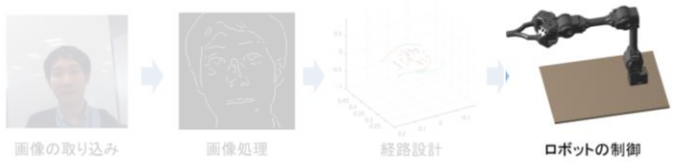
- 使用するロボット：ROBOTIS Mikata Arm



- 4アクチュエータ（＋グリッパー）
  - 3次元位置＋姿勢1つが独立に制御可能
- 各サーボモータが制御器を搭載
  - 目標角度を送ると自動でその角度まで回転

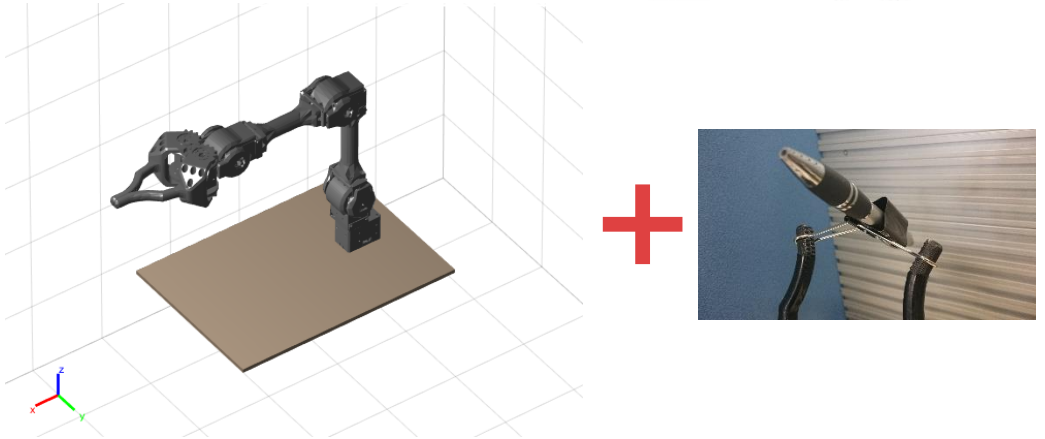
- 制御の手順
  - ロボットオブジェクトの作成
  - 逆運動学を解き、ジョイントの経路を計算
  - 経路に沿ってロボットを動かす

4. ロボットの制御② ロボットオブジェクトの作成



- ロボットオブジェクトの作成
  - URDFファイルからロボットアームのデータを読み込み

```
>> robot = importrobot('mikata_arm_4.urdf');
```



- 新しいリンク（描画用のペン）の追加
  - リンクとジョイントを作成

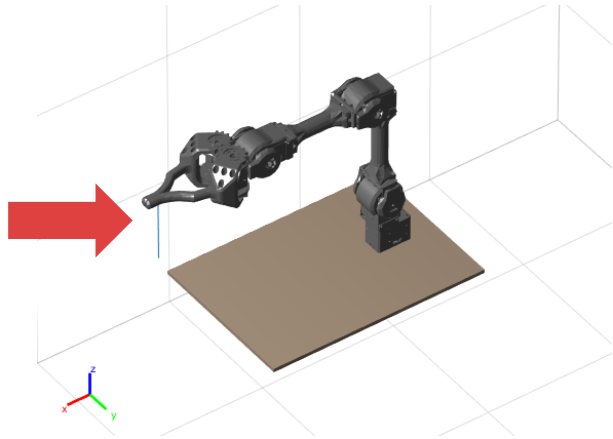
```
>> pen = robotics.RigidBody('pen');
>> penjnt = robotics.Joint('penjnt','fixed');
```

  - DHパラメータを設定

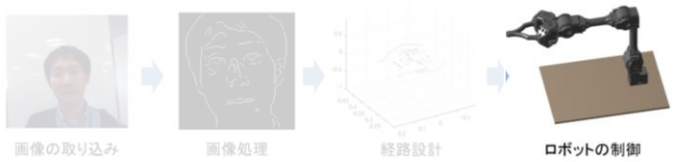
```
>> dhparam = [0 0 -0.09 0];
>> setFixedTransform(penjnt,dhparam,'dh');
```

  - リンクをロボットに追加

```
>> pen.Joint = penjnt;
>> addBody(robot,pen,'target_link');
```



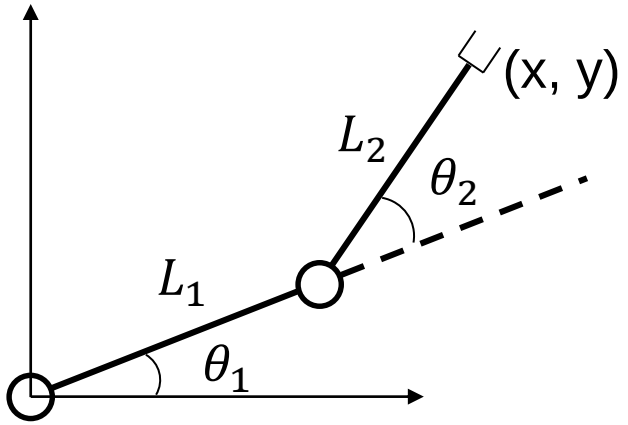
4. ロボットの制御③　ジョイント角度の経路設計



- 「逆運動学」を解いてジョイント角度を計算

- 逆運動学とは
  - 順運動学：ジョイントの角度から手先の座標を計算
  - 逆運動学：手先の座標からジョイントの角度を計算

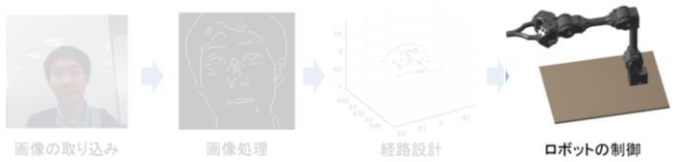
- 例：平面2リンクマニピュレータ



$$\theta_1 = \tan^{-1}\frac{y}{x} - \cos^{-1}\frac{x^2 + y^2 - L_1^2 L_2^2}{2L_1 L_2}$$

$$\theta_2 = \cos^{-1}\frac{x^2 + y^2 - L_1^2 L_2^2}{2L_1 L_2}$$

4. ロボットの制御④　4行でできる逆運動学



- Robotics System Toolbox を用いた逆運動学の計算
  - 手先の目標位置・姿勢から同次行列を作成

```
>> tf = makehgtform('translate',b1);
```

  - 逆運動学のソルバーを作成

```
>> ik = robotics.InverseKinematics('RigidBodyTree',robot);
```

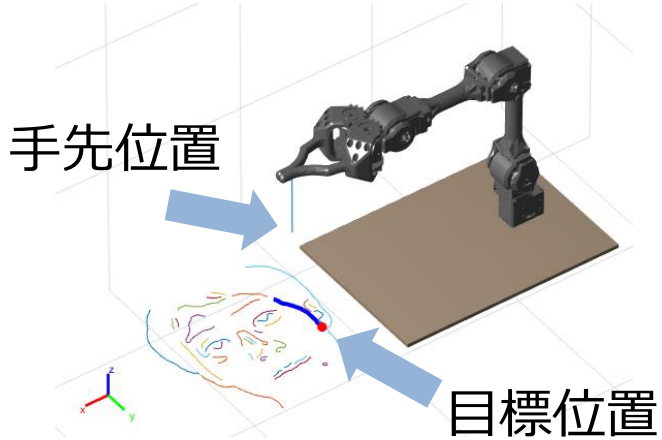
  - ソルバーで逆運動学を解く

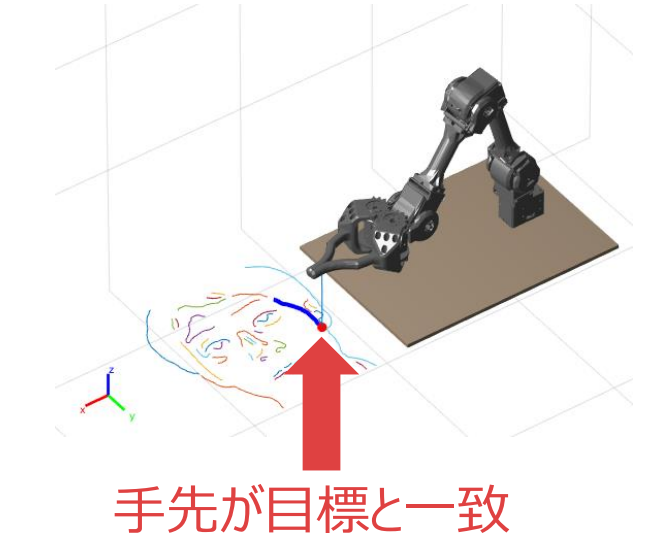
```
>> [Q,~] = ik('pen', tf, [1 1 0 1 1 1], Q);
```

リンク名　目標位置・姿勢　重み　初期値

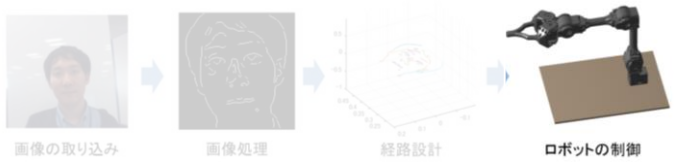
  - 計算結果からジョイント角度を取得

```
>> bJoint = arrayfun(@(x) x.JointPosition,Q);
```

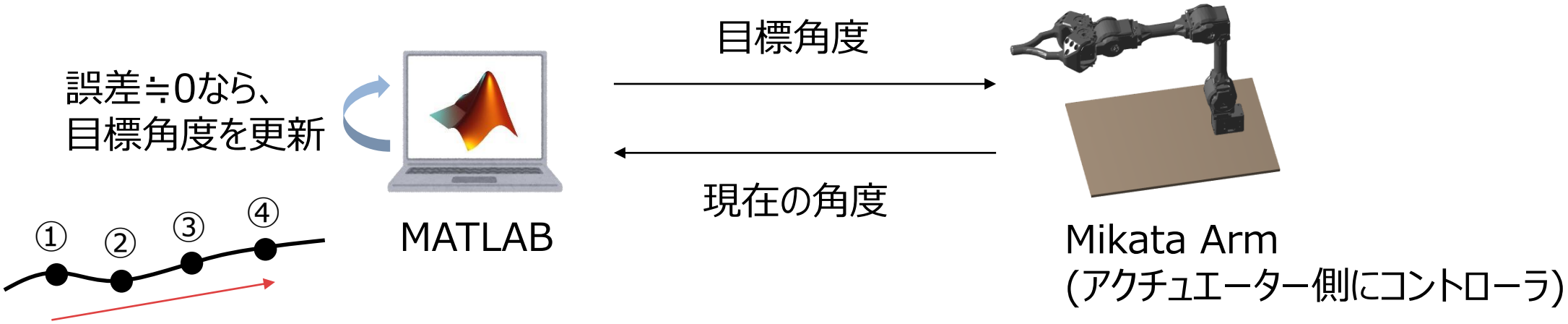




4. ロボットの制御④　ロボットの動かし方



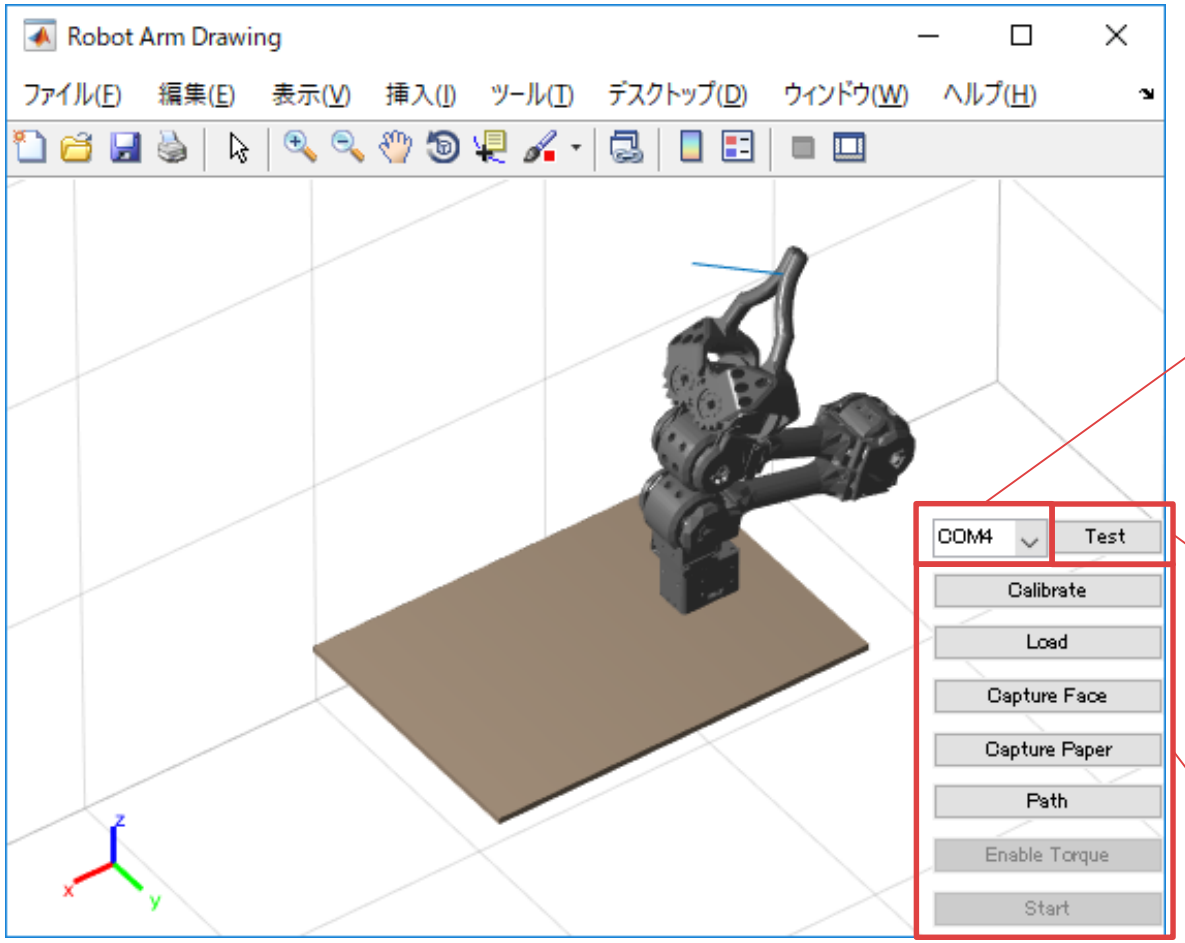
- 経路追従制御



- データのやり取りはROBOTIS Mikata Arm専用のSDKをラッピングした関数を使用

```
>> myDxl.writePositions(ref);          %目標角度の送信
>> myDxl.readPresentPositions';        %現在の角度の取得
```

アプリケーション製作①　メイン画面



シリアルポートの選択

起動時にデバイスが接続されている
シリアルポートを検出

接続の確認

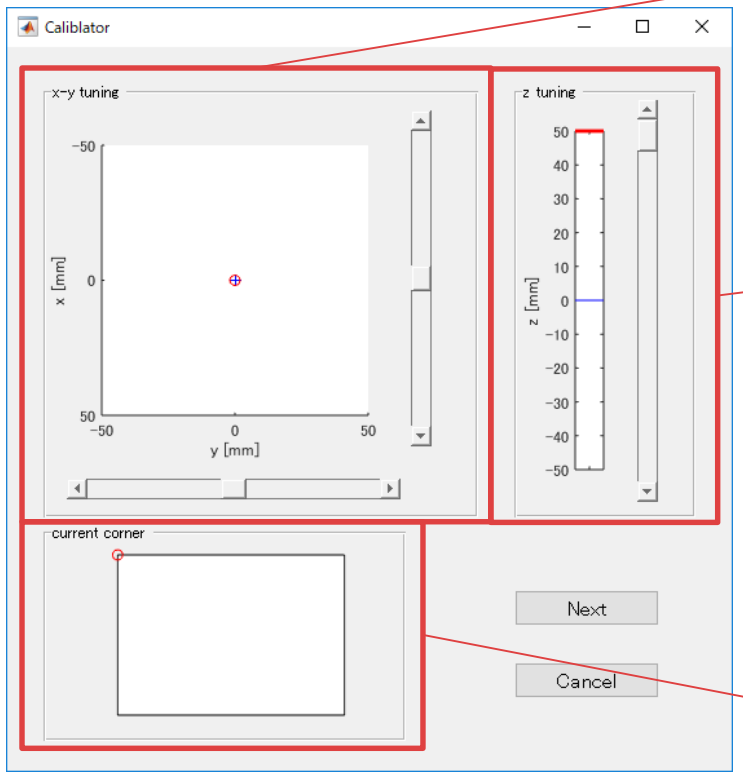
接続に成功するとポート番号が.matファイルに保存され、
次回以降はそれをデフォルト値として使用

各種操作

誤作動を防ぐため、下2つは
経路計算後のみ選択可能

アプリケーション製作①　キャリブレーション機能

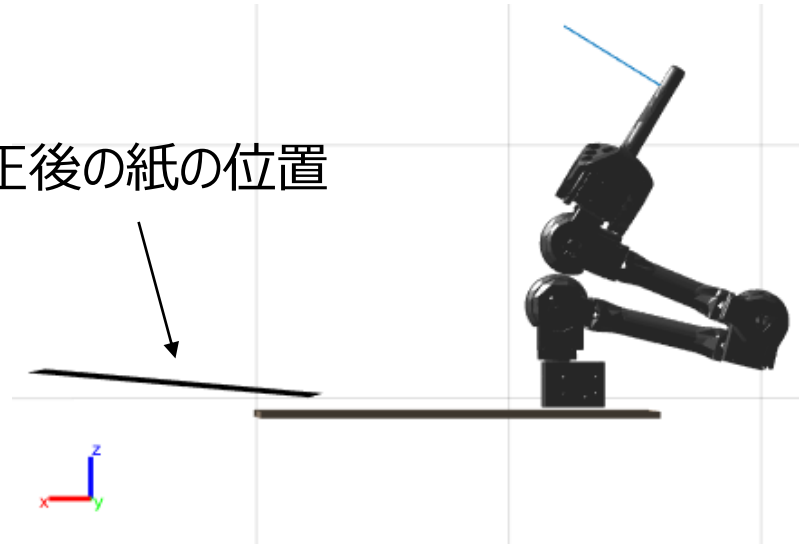
- 画像の座標系とロボットアームの座標系の位置合わせ
  - スライダーを用いてアームを手動で操作
  - 紙の実際の位置を測定し、経路を補正する



x,y方向の誤差の修正

z方向の誤差の修正

紙面上の目標位置

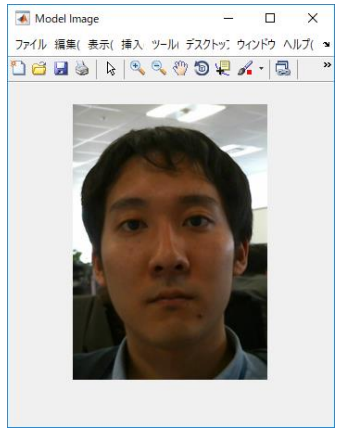


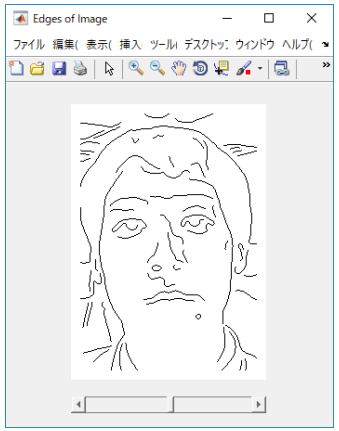
3次元経路の計算の際に
補正をかける

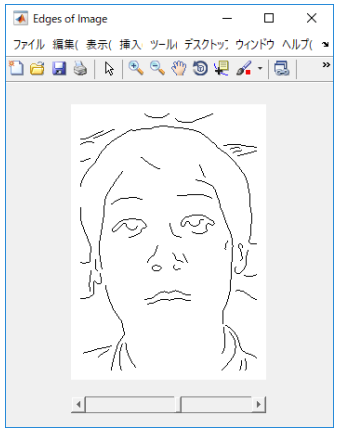
アプリケーション製作②　エッジ検出の閾値の調整

- エッジ検出の閾値をスライダーで調整可能

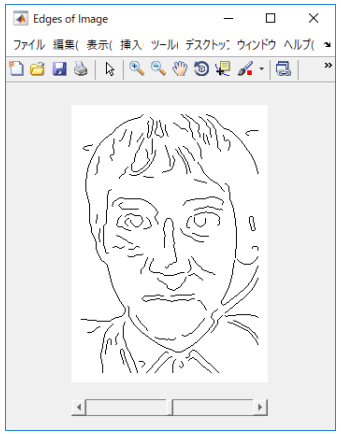
<顔が暗い場合>

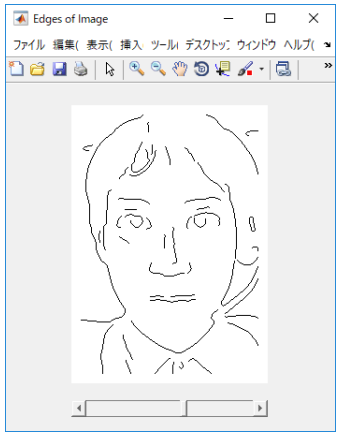






<顔が明るい場合>

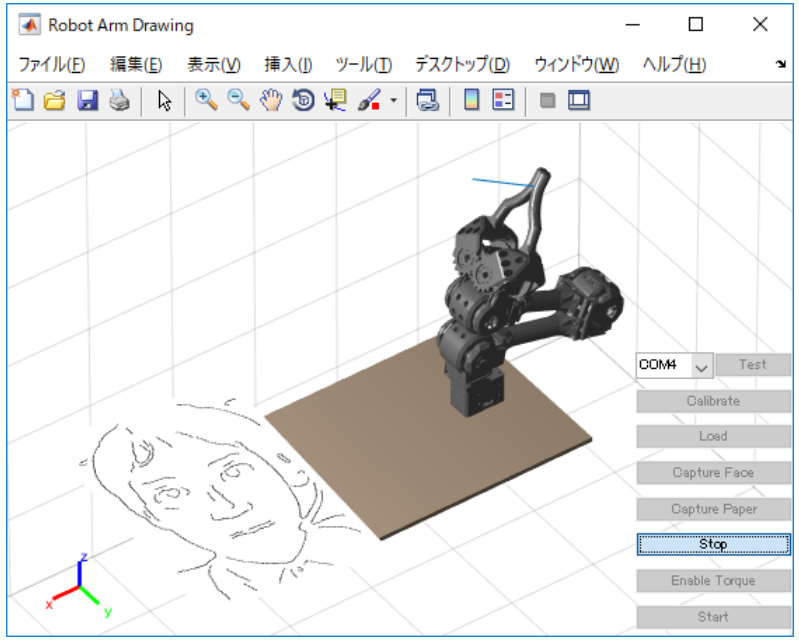




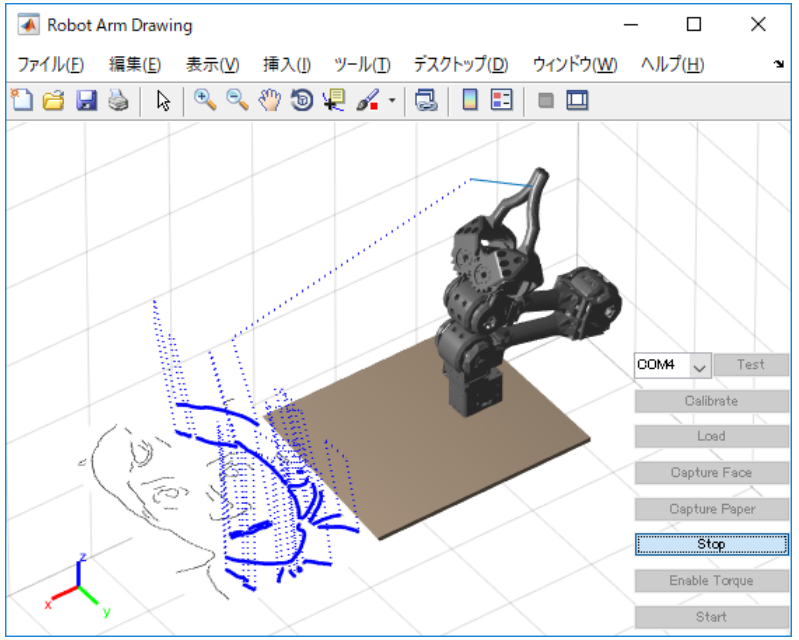
外乱光の影響があっても
適切なしきい値を設定可能！

アプリケーション製作③　経路計算の待ち時間

- 経路の計算状況を可視化
  - 進捗状況がわかり、体感速度が上がる



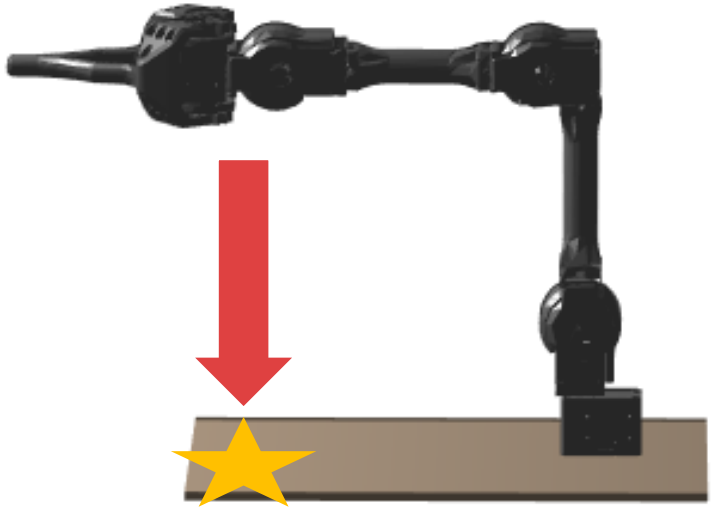
×変化なし
×いつ終わるかわからない



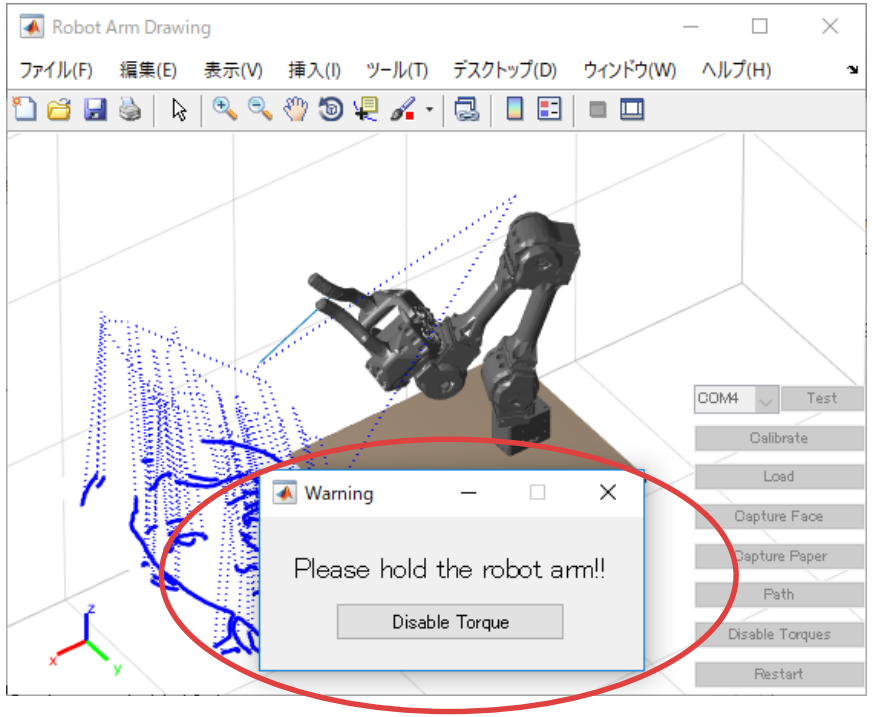
○経路の計算状況を表示
○いつ終わるか推測可能

アプリケーション製作④　トルク off 時の警告

- 初期姿勢以外でトルクを off にする際、警告を表示



描画の途中でトルクをoffにすると、
アームが落下してしまう

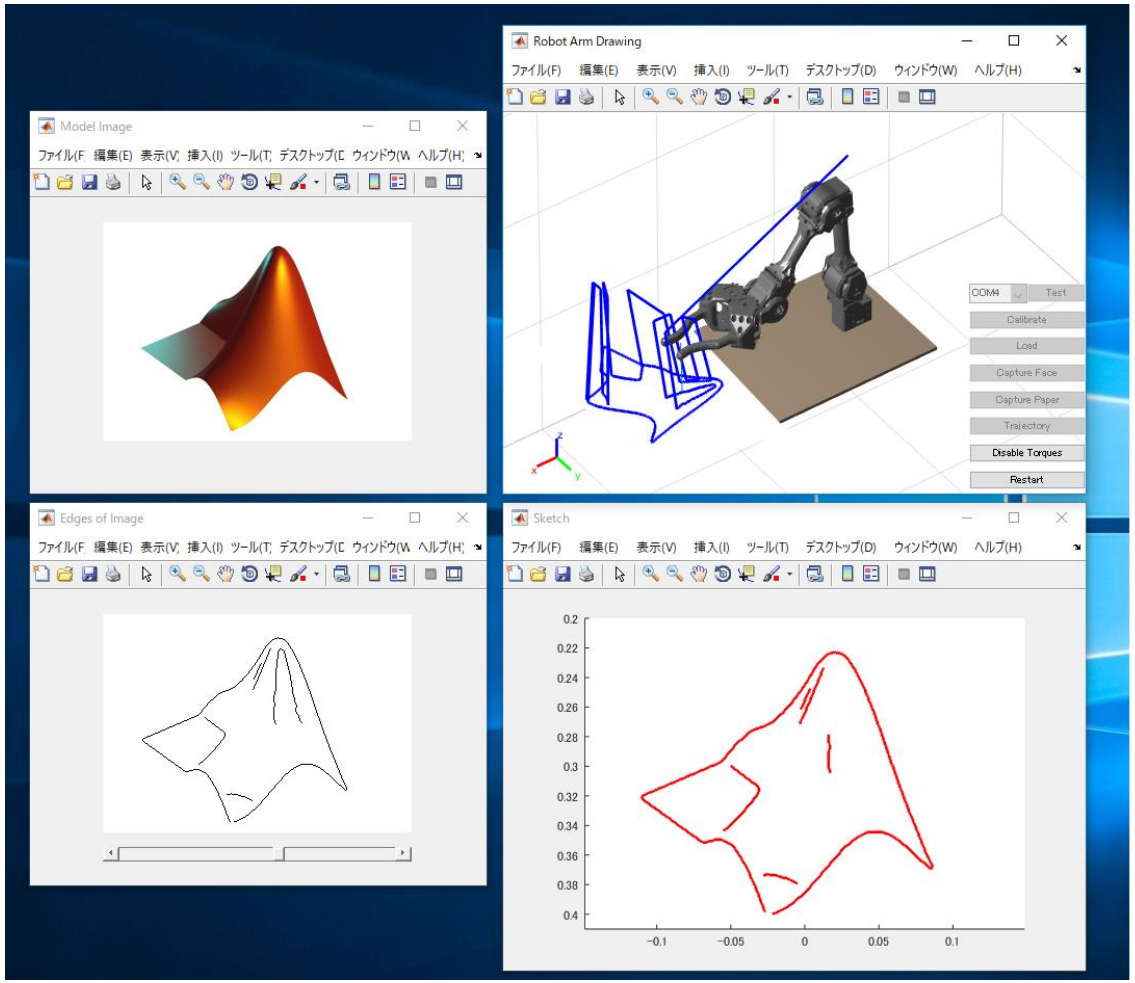


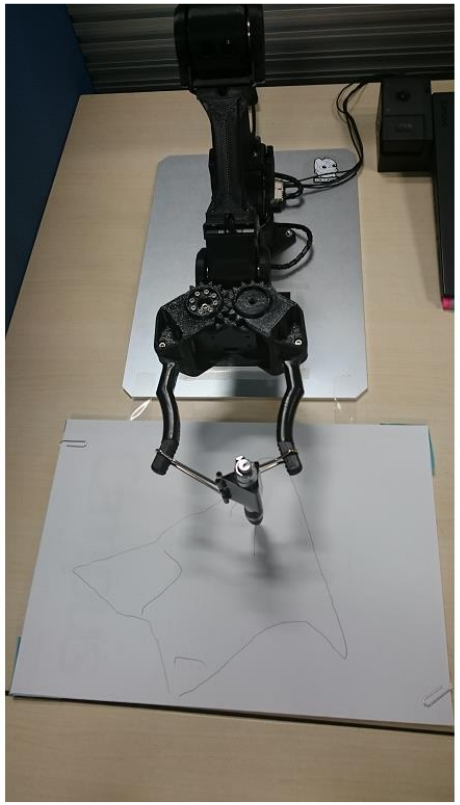
ロボットを手で支えるように警告を表示し、
落下を防ぐ

まとめ

- Image Processing Toolbox の充実した機能を活用
  - 枝分かれの検出から境界トレースまで多種多様な処理の関数
  - 複数を組み合わせることで、経路計画など様々な用途に応用可能
- Robotics System Toolbox によるマニピュレータの解析機能を活用
  - リンク構造の容易な構築と可視化
  - 逆運動学を非常に簡単に解くことが可能
- ロボットアームで似顔絵を描くデモを約1か月間で実現
  - 画像処理や経路計画など複合的な要素を含むロボットタスクを短期間で実装

Appendix：MATLABロゴの自動作画

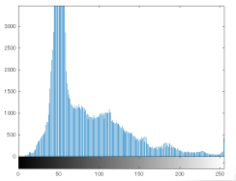




画像処理・コンピュータービジョン・機械学習

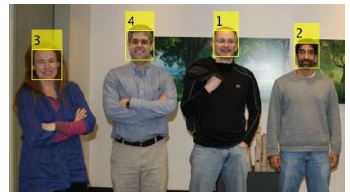
Image Processing Toolbox

- コーナー、円検出
- 幾何学的変換
- 各種画像フィルタ処理
- レジストレーション（位置合せ）
- セグメンテーション（領域分割）
- 画像の領域の定量評価



Computer Vision System Toolbox

- カメラキャリブレーション
- 特徴点・特徴量抽出
- 機械学習による物体認識
- 動画ストリーミング処理
- トラッキング
- ステレオビジョン・３D表示



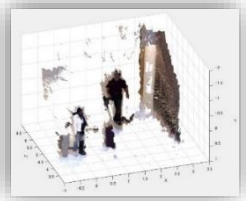
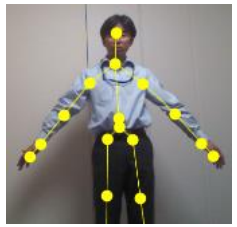


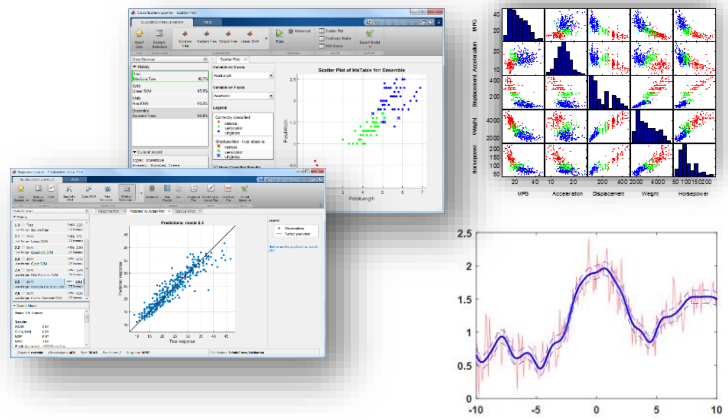
Image Acquisition Toolbox

- デバイスから画像、動画直接取り込み
  - フレームグラバボード
  - DCAM, Camera Link®
  - GigE Vision®, Webカメラ
  - Microsoft® Kinect® for Windows®



Statistics and Machine Learning Toolbox

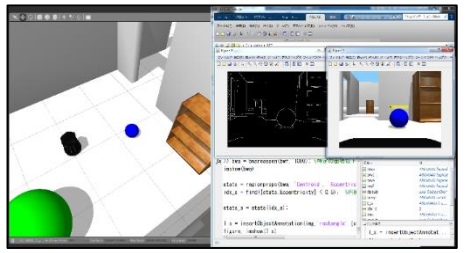
- 機械学習
- 多変量統計
- 確率分布
- 回帰と分散分析
- 実験計画
- 統計的工程管理



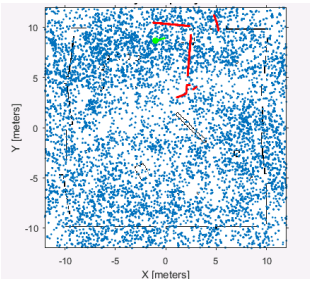
ロボットアルゴリズム開発ソリューション

Robotics System Toolbox

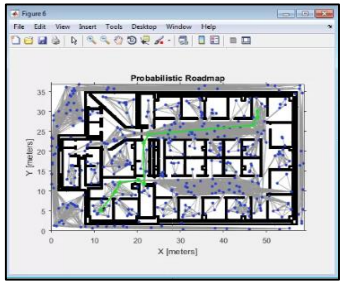
- ROSのインターフェイス提供
  - MATLABをROSマスター、ノードとして起動
  - 直接ROSネットワークに接続して検証
- ROSノード生成
  - SimulinkモデルからC++ ROSノードを生成
- ロボットアルゴリズム開発の支援
  - オイラー角、クォータニオン、座標変換などの便利な関数群
  - パスプランニング、VFH+、モンテカルロローカリゼーション(MCL)などの高度な移動用ロボットアルゴリズム
  - ツリー構造表現、逆運動学解析などのマニピュレーターアルゴリズム



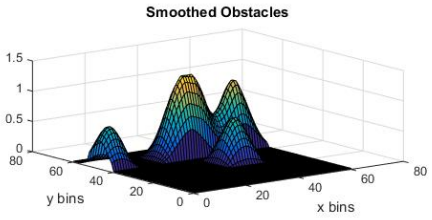
シミュレータや実機とのROS連携



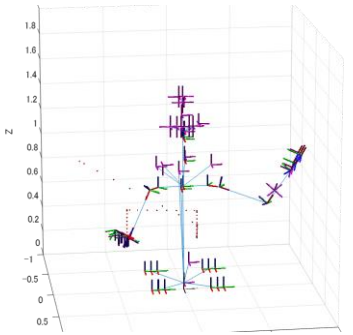
自己位置推定(AMCL)



確率的ロードマップ法(PRM)



衝突回避(VFH+)



ロボットマニピュレーターアルゴリズム開発