NEDO特別講座
画像処理・AI技術活用コース 前編

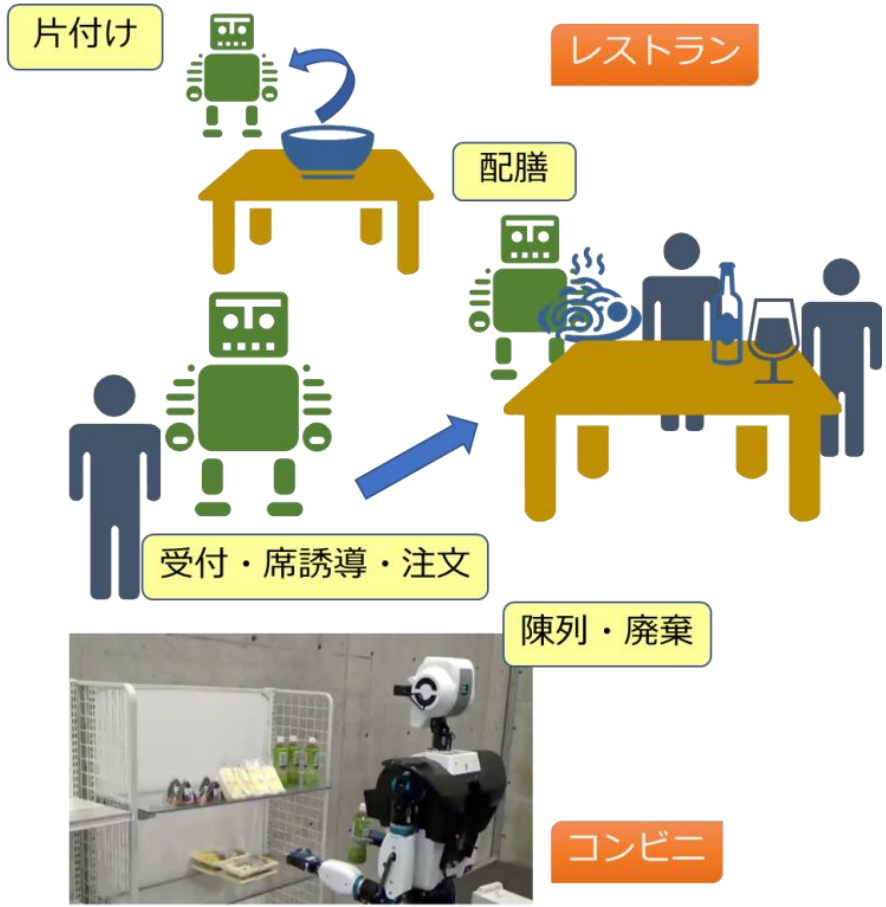
株式会社セック
Systems Engineering Consultants Co., LTD.
加藤 隆典

人と共働して軽作業をするロボットプラットフォームの開発

【我々が目指すロボットプラットフォーム】

- 未活用領域であるサービス分野における軽作業をターゲット
- 人・ロボット・IoT機器が協調・共働してサービス提供
- 信頼性と安全性（機能安全）を備えたロバストなプラットフォーム
- クラウドと連携しロボット運用のためのエコシステムを構築
- ソフトコンソと協力し、共通化機能のI/Fを提案
- プロジェクト終了後に早期事業化

【研究開発・実用化のターゲット領域】



株式会社セック
Systems Engineering Consultants Co., LTD.

講座内容

前半

- ロボットシステムでのAI活用事例
- AIを活用したロボットシステムのチュートリアル

後半

- チュートリアルのおさらい
- AIを活用したロボットシステムの設計のポイント
- AIを活用したロボットシステムの運用のポイント

ロボットシステムでのAI活用事例

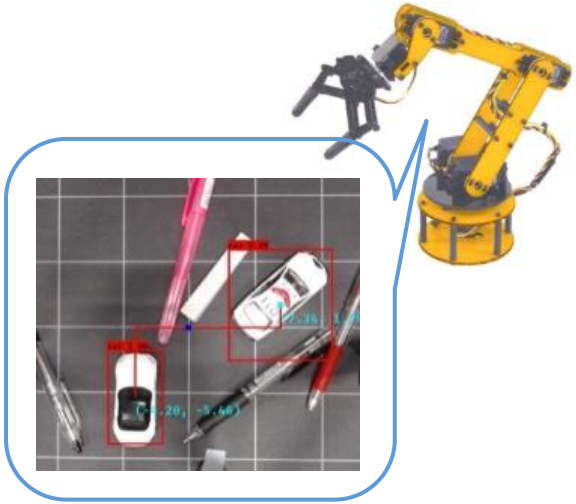
- ロボットに適用されるAI・画像処理技術
- ロボットアーム制御
- 自動運転（Autoware）

ロボットシステムでのAI活用事例：ロボットに適用されるAI・画像処理技術

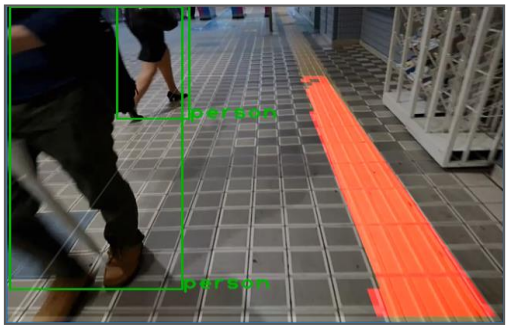
Rtrilo

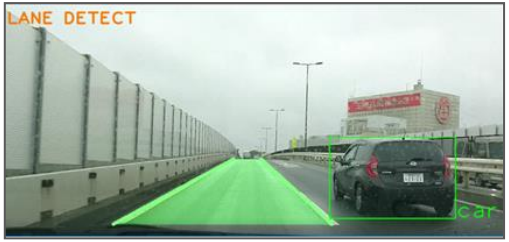
https://www.sec.co.jp/ja/rd/rtrilo.html

ディープラーニング技術や各種画像処理を活用して特定の物体・領域を抽出し、ロボットや機器の高度な制御が可能となります。

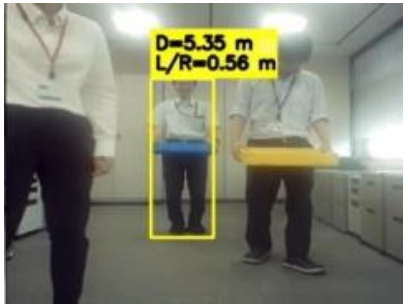


把持対象の特定





走行可能/走行禁止エリアの抽出

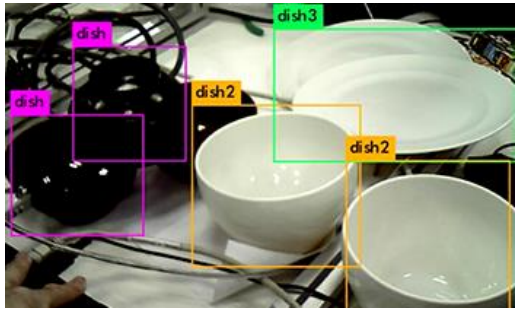


ピンポイントでの物体検出

物体の動き(移動方向)検出

ロボットシステムでのAI活用事例：ロボットに適用されるAI・画像処理技術

- 物体検出
  - YOLO(v3,tiny)
  - SSD
- 分類
  - VGG16
  - Inception(GoogLeNet)
  - ResNet
  - MobileNet
- 物体検出領域（ROI）の絞り込み
- ARマーカ読み取り
- 障害物・走行領域の抽出
- 検出物体の移動と停止識別、物体の向きの検出
- 検出物体までの距離算出



ロボットシステムでのAI活用事例：ロボットアーム制御

強化学習

ロボットアームが試行錯誤を繰り返し、その経験からロボットの制御を学習する

出典：*https://sites.google.com/site/brainrobotdata/home*

Deep Learning for Robots: Learning from Large-Scale Interaction
http://googleresearch.blogspot.jp/2016/03/deep-learning-for-robots-learning-from.html

ロボットシステムでのAI活用事例：ロボットアーム制御

模倣学習

人の動作を模倣してロボットアームの
制御方法を学習する

出典：*https://www.youtube.com/watch?v=YH1TrL1q6Po*

「NEDO次世代人工知能・ロボット中核技術開発」
早稲田大学 尾形研究室のタオルの折り畳みロボット
https://www.youtube.com/watch?v=beZUyZMyHeo

株式会社セック
Systems Engineering Consultants Co., LTD.

ロボットシステムでのAI活用事例：自動運転（Autoware）

Autoware：オープンソースの自動運転ソフトウエア

https://www.autoware.ai/

AIを活用したロボットシステムのチュートリアル

AIを活用したロボットシステムのチュートリアル

本講座のチュートリアルでは、ROSを利用して、物体検出を行うロボットシステムを構築します。ロボットシステムの構築の流れとともに、ロボットシステムにAIを組み込むうえでのポイントを、チュートリアルを通して解説します。

Webカメラの映像を取り込み物体検出の結果を動画として表示する
物体検出ノードは複数のアルゴリズムを実装し切り替え可能とする

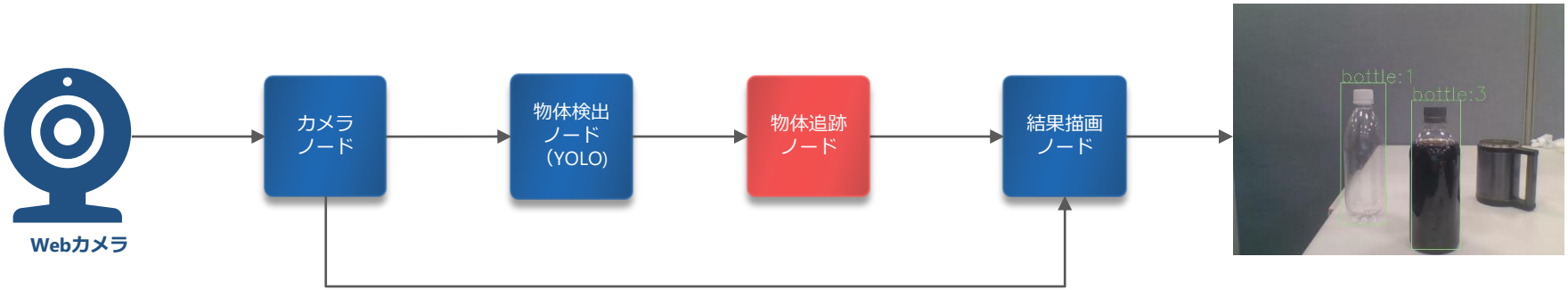
AIを活用したロボットシステムのチュートリアル

本講座のチュートリアルでは、ROSを利用して、物体検出を行うロボットシステムを構築します。ロボットシステムの構築の流れとともに、ロボットシステムにAIを組み込むうえでのポイントを、チュートリアルを通して解説します。

Webカメラの映像を取り込み物体検出の結果を動画として表示する
物体検出ノードは複数のアルゴリズムを実装し切り替え可能とする

AIを活用したロボットシステムのチュートリアル

本講座のチュートリアルでは、ROSを利用して、物体検出を行うロボットシステムを構築します。ロボットシステムの構築の流れとともに、ロボットシステムにAIを組み込むうえでのポイントを、チュートリアルを通して解説します。



Webカメラの映像を取り込み物体検出の結果を動画として表示する
物体検出ノードは複数のアルゴリズムを実装し切り替え可能とする

AIを活用したロボットシステムのチュートリアル

チュートリアル全体の流れ

物体検出システムを作成する

物体検出モデルをSSDに入れ替える

物体追跡機能を追加する

AIを活用したロボットシステムのチュートリアル

チュートリアルの実行環境

- OS: Ubuntu 18.04 LTS
- ROS: Melodic
- カメラ

チュートリアルのソースコード・実装手順については、動画の概要欄に記載のGithubのページを参照してください。

物体検出システムを作成する

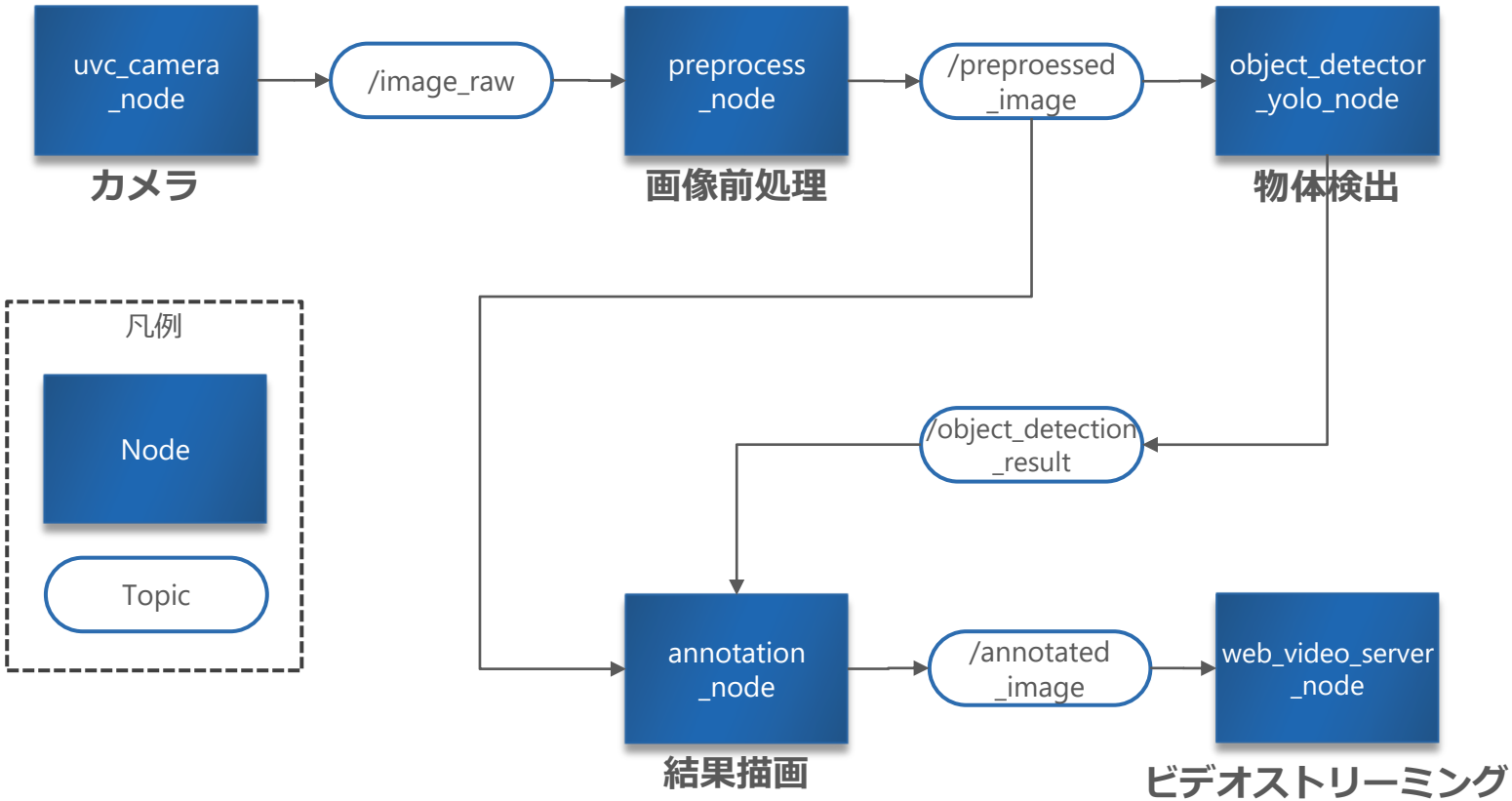
物体検出システムを作成する

物体検出モデルをSSDに入れ替える

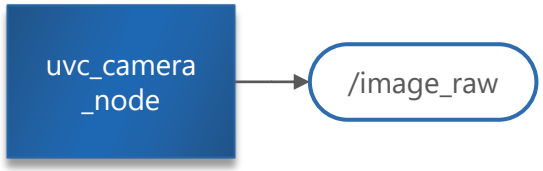
物体追跡機能を追加する

- オープンソースのノードのインストール
  - カメラノード
    ros-melodic-uvc-camera
  - ビデオストリーミング
    ros-melodic-web-video-server
- object_detector_msgパッケージ
  - DetectedObject.msgの作成
  - ObjectDetectionResult.msgの作成
- object_detectorパッケージ
  - 画像前処理ノードの実装
  - 物体検出ノードの実装
  - 結果描画ノードの実装
  - Launchファイルの作成

ノードの構成



カメラノード



uvc_camera_node

uvc_camera (http://wiki.ros.org/uvc_camera)

カメラデバイスから画像を取得するノードには、オープンソースの実装を利用する。

```
sudo apt install ros-melodic-uvc-camera
```

/image_raw

senser_msg/Image (https://docs.ros.org/en/melodic/api/sensor_msgs/html/msg/Image.html)

ROS標準のImageメッセージが送信される。

カメラデバイスやカメラノードの実装によって、サポートする画像のフォーマットが異なる場合があります。
物体検出などのAIモデルは画像をRGBで扱うことが多いため、後のpreprocess_nodeで画像フォーマットを
RGBに変換します。

画像前処理ノード

preprocess_node

src/object_detector/script/preprocess.py

カメラデバイスから取得した画像を、物体検出モデルで利用できるよう変換する。今回の例では画像のフォーマットをRGBに変換する。必要であれば以下の処理も含める。

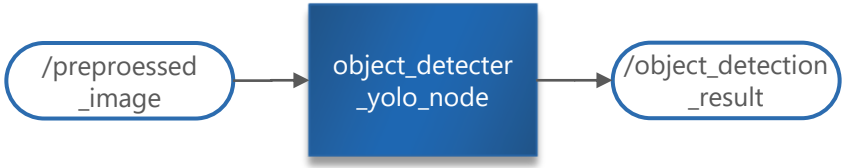
- リサイズ / クロップ / コントラスト調整

/preprocessed_image

senser_msg/Image (https://docs.ros.org/en/melodic/api/sensor_msgs/html/msg/Image.html)

ROS標準のImageメッセージが送信される。

物体検出ノード



object_detector_yolo_node

src/object_detector/script/object_detector_yolo.py

オープンソースで公開されるYOLOの実装を、ROSノードとしてラップする。

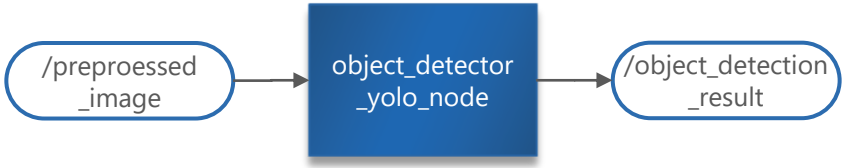
src/object_detector/

scripts/object_detector_yolo.py

lib/pytorch-YOLOv4

```
git clone https://github.com/Tianxiaomo/pytorch-YOLOv4
```

物体検出ノード



object_detecter_yolo_node

src/object_detector/script/object_detector_yolo.py

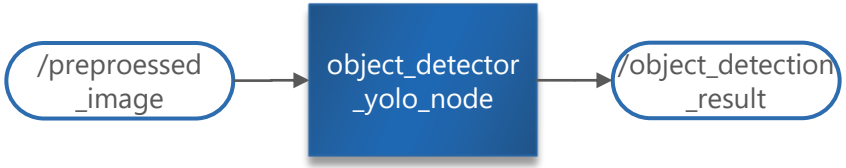
オープンソースで公開されるYOLOの実装を、ROSノードとしてラップする。

ROS1はpython2で動作していますが、今回利用しているYOLOはpython3で実装されているため、このノードは明示的にpython3で実行する必要があります。

object_detector_yolo.py

```
#!/usr/bin/env python3
…
```

物体検出ノード



/object_detecter_result

src/object_detector_msg/msg/ObjectDetectionResult.msg
src/object_detector_msg/msg/DetectedObject.msg

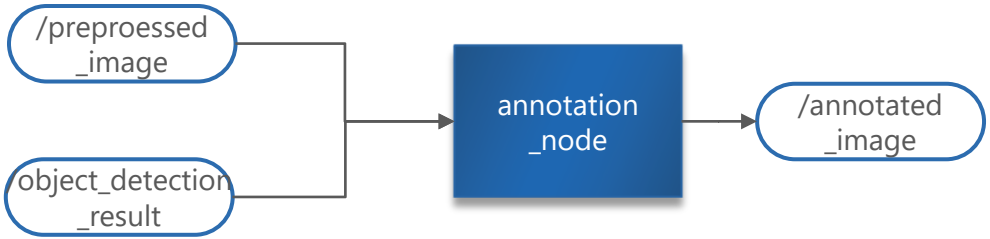
物体検出を結果を、独自のメッセージとして
定義する。

ObjectDetectionResult	
detected_objects	DetectedObject[]

DetectedObject	
xmin	float64
xmax	float64
ymin	float64
ymax	float64
confidence	float64
class_id	int64
name	string

物体の検出位置 → xmin, xmax, ymin, ymax
検出の信頼度 → confidence
物体の分類ID → class_id
物体の名前 → name

結果描画ノード



annotation_node

src/object_detector/script/annotation.py

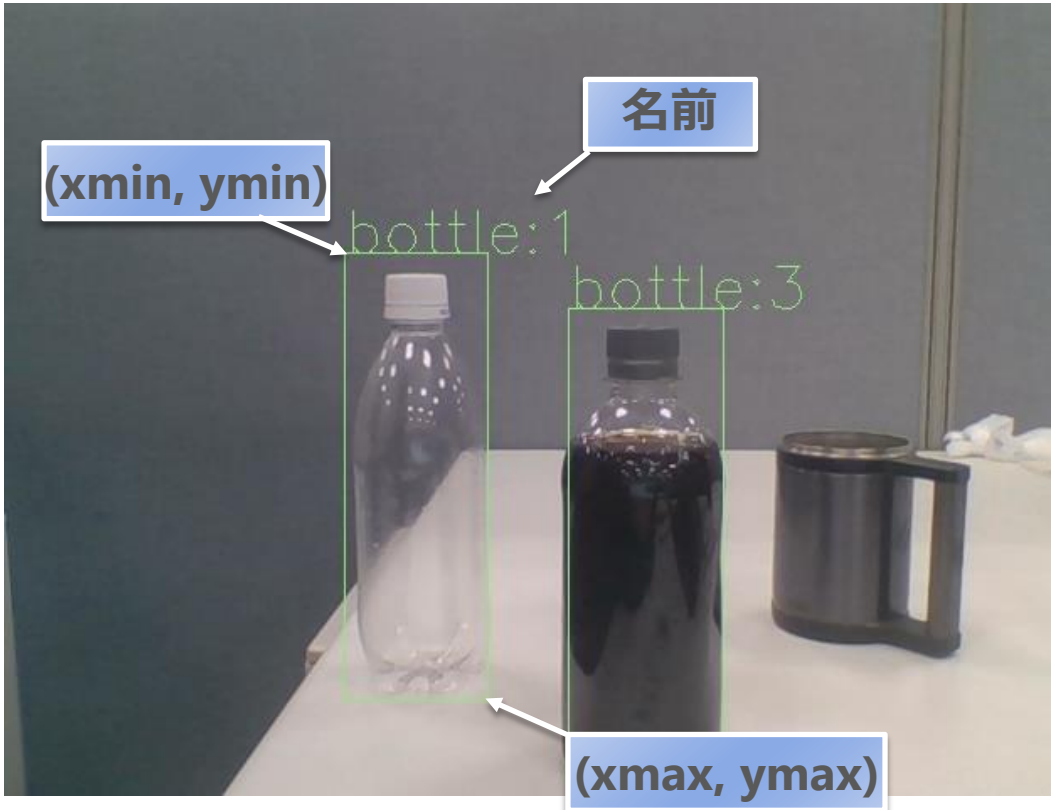
チュートリアルでは、opencvにより右上の画像のような結果を描画する。

/annotated_image

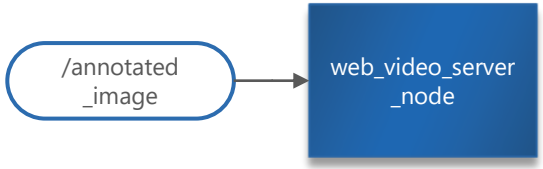
senser_msg/Image (https://docs.ros.org/en/melodic/api/sensor_msgs/html/msg/Image.html)

RGBフォーマットの画像。

物体検出ノード



ビデオストリーミングノード



web_video_server_node

web_video_server (http://wiki.ros.org/web_video_server)

Imageトピックを、Httpでアクセス可能なビデオストリームとして配信するノード。

```
sudo apt install ros-melodic-web-video-server
```

Launchファイル

src/object_detector/launch/object_detector_yolo.launch

```
<launch>
  <node name="uvc_camera_node" pkg="uvc_camera" type="uvc_camera_node" />
  <node name="web_video_server_node" pkg="web_video_server" type="web_video_server" />
  <node name="preprocess_node" pkg="object_detector" type="preprocess.py" />
  <node name="object_detector_yolo_node" pkg="object_detector" type="object_detector_yolo.py">
    <param name="model_path" value="~/catkin_ws/src/object_detector/lib/pytorch-YOLOv4/" />
  </node>
  <node name="annotation_node" pkg="object_detector" type="annotation.py" />
</launch>
```

株式会社セック
Systems Engineering Consultants Co., LTD.

物体検出モデルをSSDに入れ替える

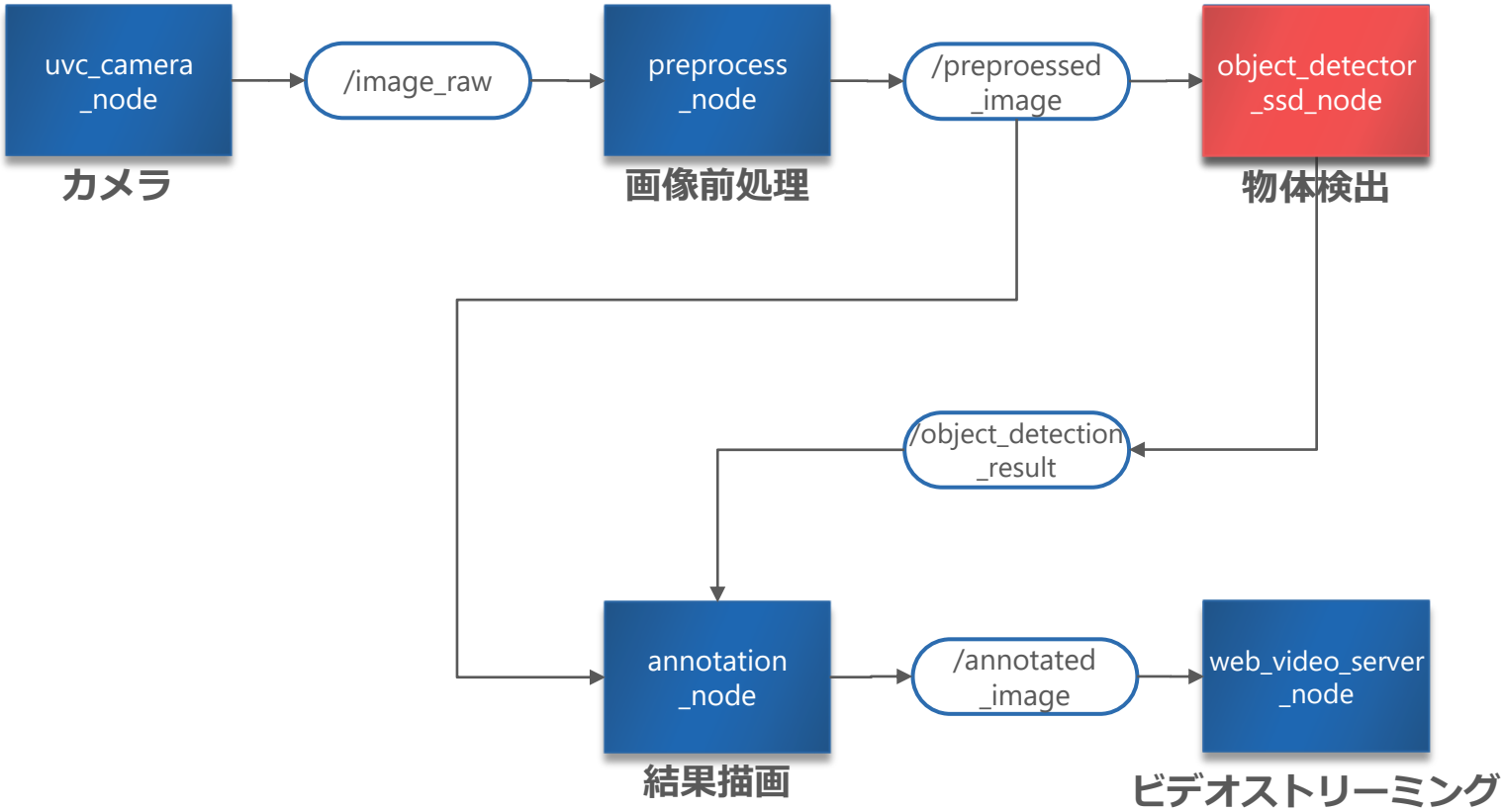
物体検出システムを作成する

物体検出モデルをSSDに入れ替える

物体追跡機能を追加する

- object_detectorパッケージ
  - 物体検出ノード（SSD）の実装
  - Launchファイルの作成

ノードの構成



物体検出ノード

object_detector_ssd_node

src/object_detector/script/object_detector_ssd.py

オープンソースで公開されるSSDの実装を、ROSノードとしてラップする。

src/object_detector/

scripts/object_detector_ssd.py

lib/SSD

git clone https://github.com/lufficc/SSD

物体検出ノード

object_detector_ssd_node

src/object_detector/script/object_detector_ssd.py

入力・出力のトピックが共通であるため、launchファイルの変更のみで物体検出ノードを
入れ替えることができる。

AIを活用したシステムの開発では、精度やパフォーマンス調整のため複数のモデルを
試すことが多いため、モデル部分を入れ替えやすく設計することが、重要なポイントです。

Launchファイル

src/object_detector/launch/object_detector_ssd.launch

```
<launch>
  <node name="uvc_camera_node" pkg="uvc_camera" type="uvc_camera_node" />
  <node name="web_video_server_node" pkg="web_video_server" type="web_video_server" />
  <node name="preprocess_node" pkg="object_detector" type="preprocess.py" />
  <node name="object_detector_ssd_node" pkg="object_detector" type="object_detector_ssd.py">
    <param name="model_path" value="~/catkin_ws/src/object_detector/lib/SSD/" />
  </node>
  <node name="annotation_node" pkg="object_detector" type="annotation.py" />
</launch>
```

株式会社セック
Systems Engineering Consultants Co., LTD.

物体追跡機能を追加する

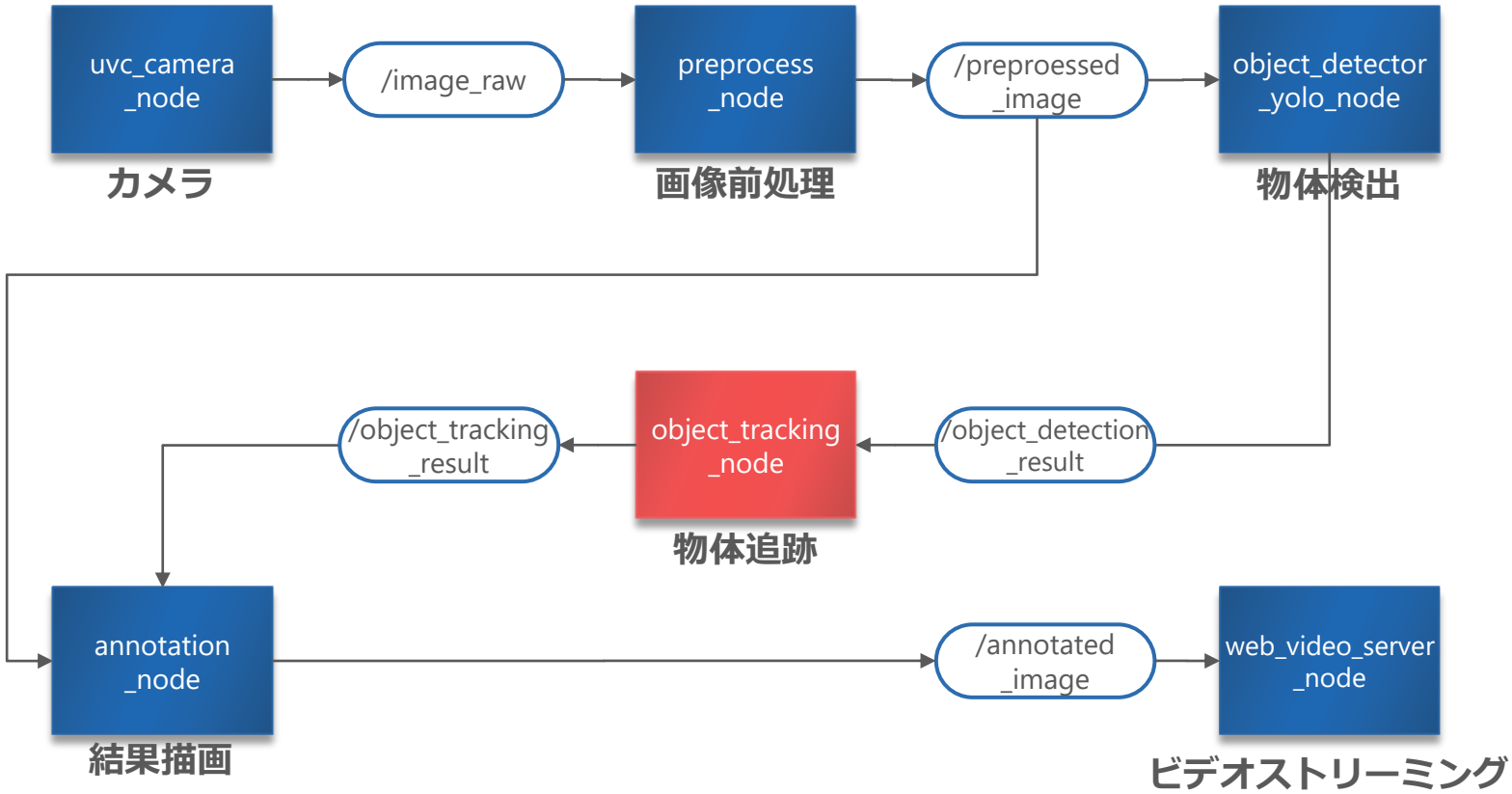
物体検出システムを作成する

物体検出モデルをSSDに入れ替える

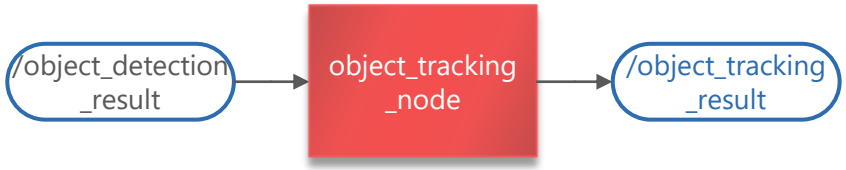
物体追跡機能を追加する

- object_detectorパッケージ
  - 物体追跡ノードの実装
  - Launchファイルの作成

ノードの構成



物体追跡ノード



object_tracking_node

src/object_detector/script/object_tracking.py

オープンソースで公開されるトラッキングの実装を、ROSノードとしてラップする。

```
pip install norfair
```

/object_tracking_result

src/object_detector_msg/msg/ObjectDetectionResult

ObjectDetectionResultを流用。追跡された物体それぞれの名前に追跡番号を付与する。

Launchファイル

src/object_detector/launch/object_tracking.launch

```
<launch>
  <node name="uvc_camera_node" pkg="uvc_camera" type="uvc_camera_node" />
  <node name="web_video_server_node" pkg="web_video_server" type="web_video_server" />
  <node name="preprocess_node" pkg="object_detector" type="preprocess.py" />
  <node name="object_detector_yolo_node" pkg="object_detector" type="object_detector_yolo.py">
      <param name="model_path" value="~/catkin_ws/src/object_detector/lib/pytorch-YOLOv4/" />
  </node>
  <node name="object_tracking_node" pkg="object_detector" type="object_tracking.py" />
  <node name="annotation_node" pkg="object_detector" type="annotation.py">
    <remap from="object_detection_result" to="object_tracking_result" />
  </node>
</launch>
```

講座後半について

前半

- ロボットシステムでのAI活用事例
- AIを活用したロボットシステムのチュートリアル

後半

- チュートリアルのおさらい
- AIを活用したロボットシステムの設計のポイント
- AIを活用したロボットシステムの運用のポイント