カメラ機能共通インタフェース仕様書

（第 2.1 版）

2014 年 6 月 19 日

【改版履歴】

日付	版番号	改版ページ	改版内容
2012.2.24	1.0	全ページ	新規作成
2014.1.24	2.0	8 ページ 13,14 ページ	ColorFormat に CF_PNG, CF_JPEG を追加 ColorFormat に FourCC 関連の形式を追加 カメラデバイス情報を取得するための型および， CameraCaptureService への getDeviceProfile の追加
2014.6.16	2.1	9,10 ページ	ColorFormat に説明を追記

【本書の策定メンバー】

（敬称略、五十音順）

大原賢一　（大阪大学大学院基礎工学研究科）

小笠原哲也　（東京大学大学院 情報理工学系研究科 知能機械情報学専攻）

河井良浩　（独立行政法人産業技術総合研究所 知能システム研究部門 タスクビジョン研究グループ）

川端聡　（独立行政法人産業技術総合研究所 知能システム研究部門 タスクビジョン研究グループ）

中本啓之　（株式会社セック 開発本部 第四開発部）

二宮恒樹　（富士ソフト株式会社 ロボット事業グループ 商品開発ユニット）

（所属は 2012 年 2 月 24 日現在）

目次

表目次

図目次

1　はじめに

近年、ロボットの開発を効率化するためにコンポーネントベースのミドルウェア開発が盛んになっている。コンポーネントベースのミドルウェア開発において、インタフェースの共通化は、コンポーネントの相互接続性や相互運用性を確保するうえで非常に重要である。このような背景に基づき、本書では、ロボットのカメラ機能に関わるインタフェースの共通仕様を定義する。

1.1　対象機能の概要

本仕様書では、ロボットシステムがカメラを使用して周囲の環境などを計測する際に使用するキャプチャモジュールの「カメラ機能共通インタフェース仕様」を規定している。

本仕様書で規定する共通インタフェースは、マルチカメラを含む画像に関連した幅広い用途への利用を想定しており、階層的なデータ構造の定義を行っている。また、OpenCV に準じた形式で内部パラメータ及びレンズの歪みパラメータを利用できる形となっている。

カメラ機能共通インタフェースを実装した RT コンポーネントの使用シーンの一例を以下に示す。

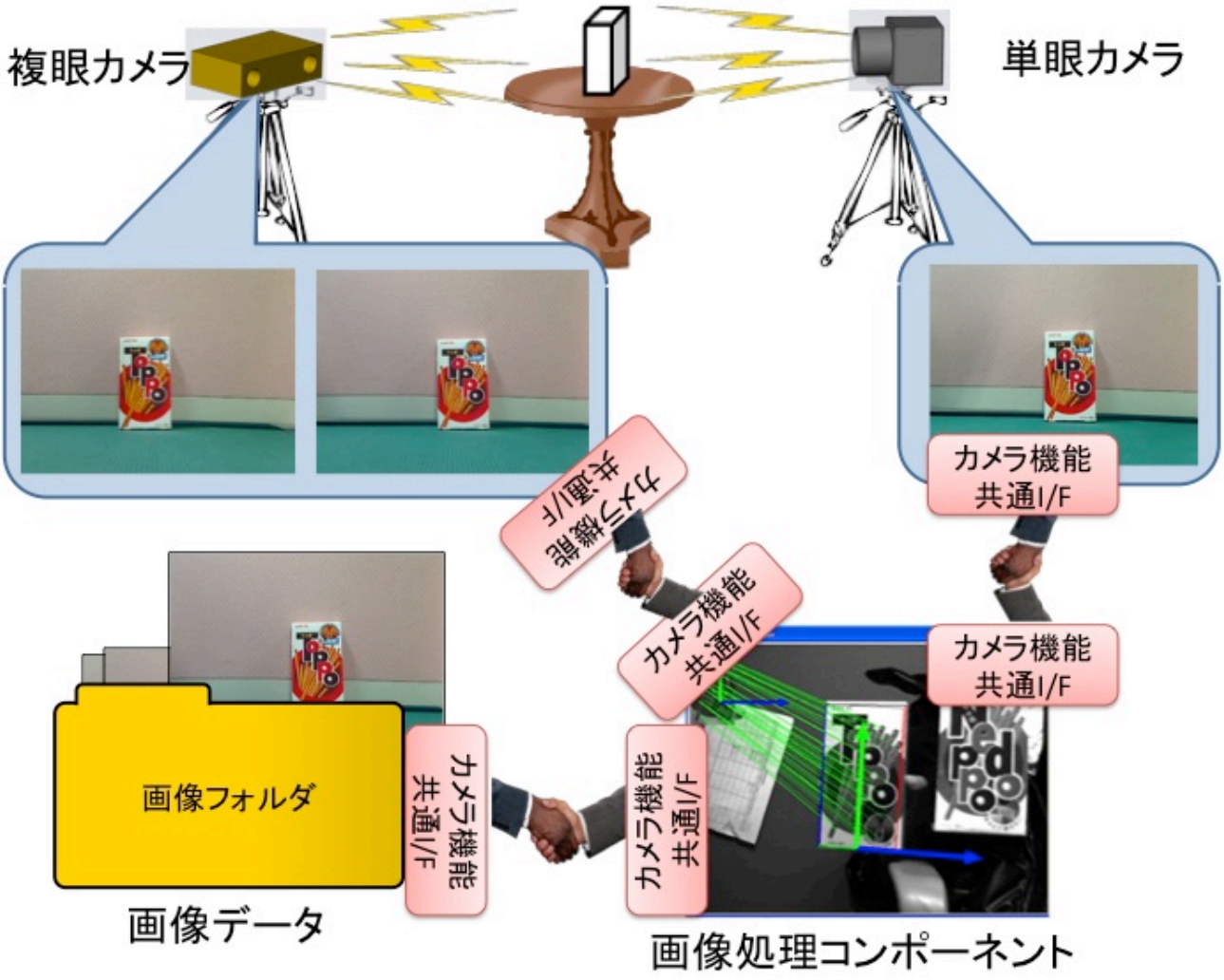

図 1.1 カメラ機能共通インタフェースの使用シーン例

1.2　標準システム構成

カメラ機能共通インタフェースを利用した標準的なシステム構成例を以下に示す。

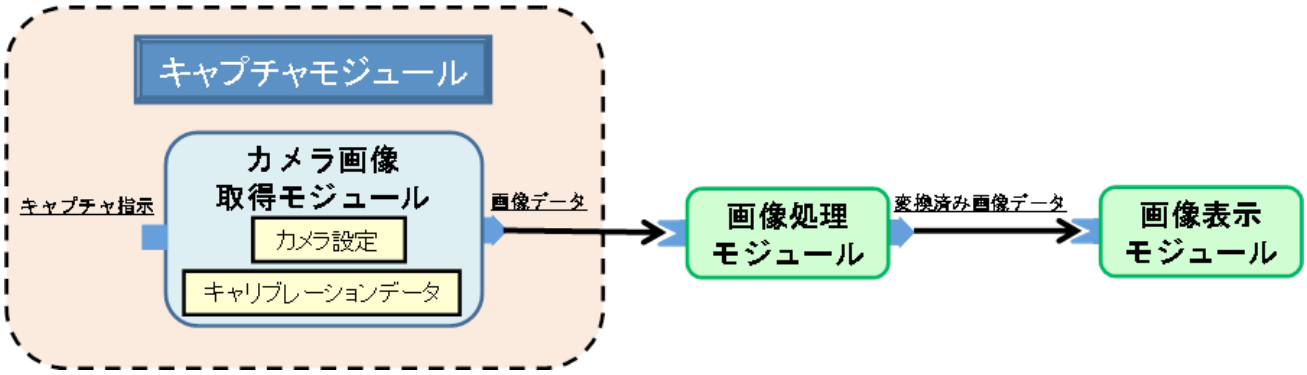

図　1.2　カメラ機能共通インタフェースを使用したシステム例（その１）

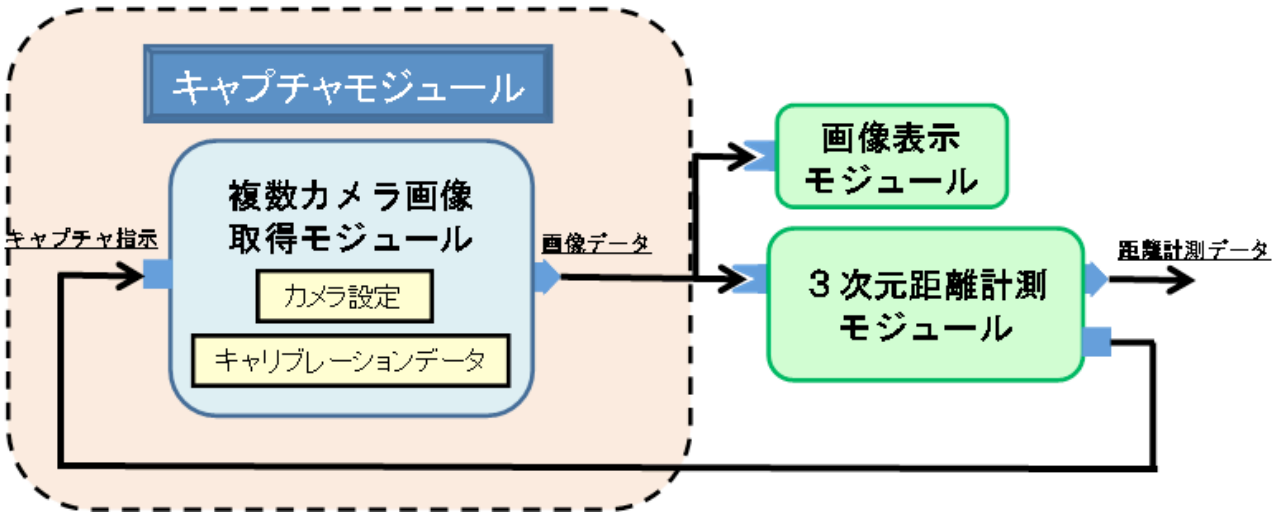

図　1.3　カメラ機能共通インタフェースを使用したシステム例（その２）

本仕様書中では、上図中の「キャプチャモジュール」を対象としており、このモジュールが出力するデータ形式および、外部コンポーネントから「キャプチャモジュール」に画像情報の取得を指示するためのインタフェースを規定している。

また、「キャプチャモジュール」が画像情報を取得するための手段としては、外部ファイルからの読み込み、単一のカメラを用いた撮影、複数のカメラを用いた撮影を想定しており、それぞれに対応したデータ型を階層的に定義している。

2　本書を読む上での注意

2.1　基本方針

インタフェース仕様の共通化は、仕様に合致しないコンポーネントを排除するため、時に開発内容を制限してしまうこともある。本仕様では、そのような制限を低減するために、以下のような方針で共通インタフェース仕様を定義する。

- 最低限のインタフェース仕様の定義:コンポーネントを相互接続・相互運用するために必要な最低限のインタフェース仕様のみを定義する。開発の制約となる仕様は最低限にとどめ、その他の部分は開発者が自由に拡張することができるようにする。
- 任意の機能の定義:いくつかの機能については実装を任意とする。実装された場合は、本書に書かれた仕様に準拠することを要求するが、実装をするかどうかは任意であり、それを実装していなかったからといって共通インタフェース仕様から外れるものとはしない。

2.2　フォーマットと表現方法

2.2.1 列挙型定義

本仕様書では、列挙型定義を次の表形式を用いて記述する。

表 XX 〈列挙型名〉

〈定数名〉	〈内容〉

2.2.2 型定義

本仕様書では、型定義を次の表形式を用いて記述する。

表 XX 〈型名〉

属性		
〈要素名〉	〈要素型〉	〈内容〉
・・・	・・・	・・・

2.2.3 インタフェース定義

本仕様書では、インタフェース定義を次の表形式を用いて記述する。

表 XX 〈インタフェース名〉

メソッド				
〈メソッド名〉	〈戻り値型〉	〈内容〉		
	〈方向〉	〈パラメータ名〉	〈パラメータ型〉	〈内容〉
	・・・	・・・	・・・	・・・

2.3　本仕様書における前提条件

2.3.1 カメラパラメータについて

本仕様書で規定しているカメラ機能共通インタフェースでは、ピンホールカメラモデルを採用している。ピンホールカメラモデルは、以下に示すように小穴(ピンホール)を焦点としたカメラであり、被写体は上下反転した姿で画像平面に投影される。

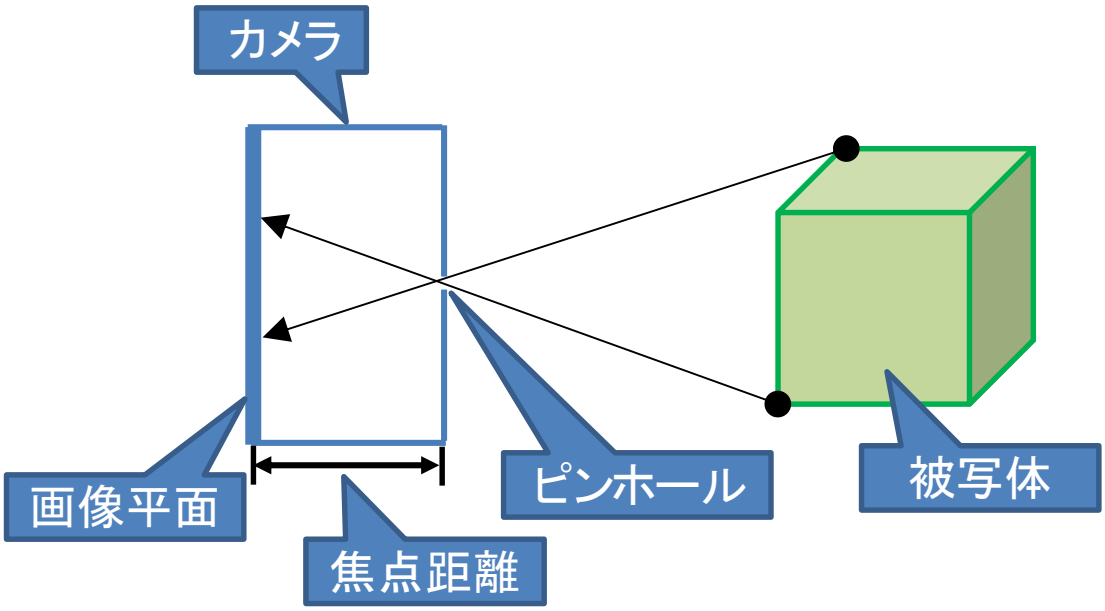

図　2.1 ピンホールカメラモデル

ここで上記のモデルは、画像平面をピンホールの前に配置し、以下のように考えても等価となる。このようなカメラモデルを透視射影モデルと呼ぶ。

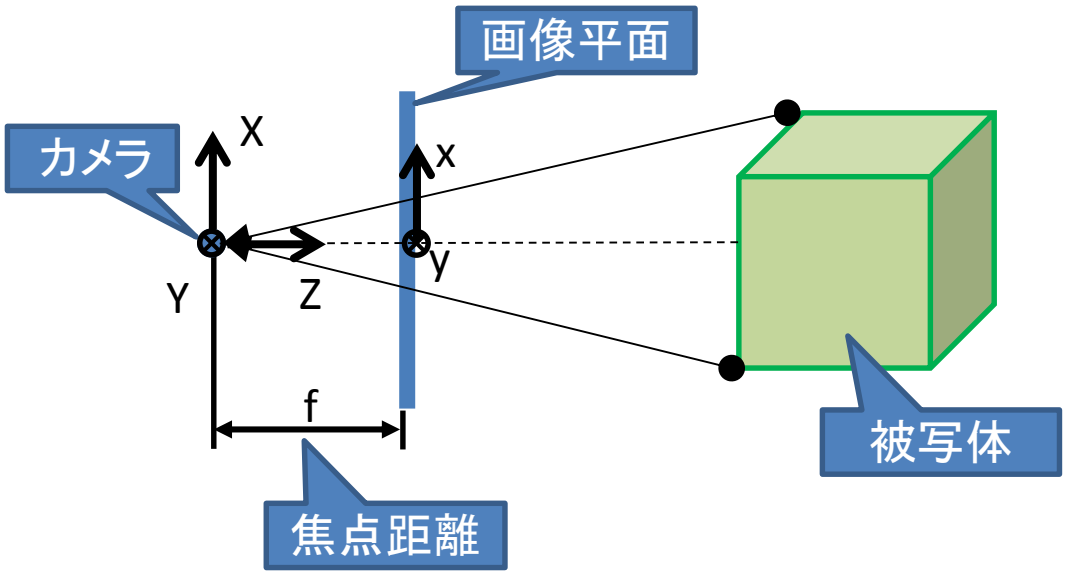

図　2.2 透視射影モデル

ここで、カメラレンズ中央に座標系 o-XYZ、画像平面中の画像中心に o-xy をそれぞれ定義すると、これらの座標系間の関係は以下のように示すことができる。

$$\lambda\begin{bmatrix}x\\y\\1\end{bmatrix}=\begin{bmatrix}f&0&0&0\\0&f&0&0\\0&0&1&0\end{bmatrix}\begin{bmatrix}X\\Y\\Z\\1\end{bmatrix}$$

f:焦点距離

λ:任意の定数

しかし、実際のカメラモデルでは、画像中心のズレ、アスペクト比、スキューなどを考慮する必要がある。そこで以下のような座標系を考える。座標系 c-xy は画像中心 c を原点とし、x 軸と y 軸は同じスケールをもつ。座標系 o-uv はカメラ内のデジタル画像座標系であり、x 軸と y 軸が異なるスケールを持つ場合も、x 軸と y 軸が直交しない場合もありえる。

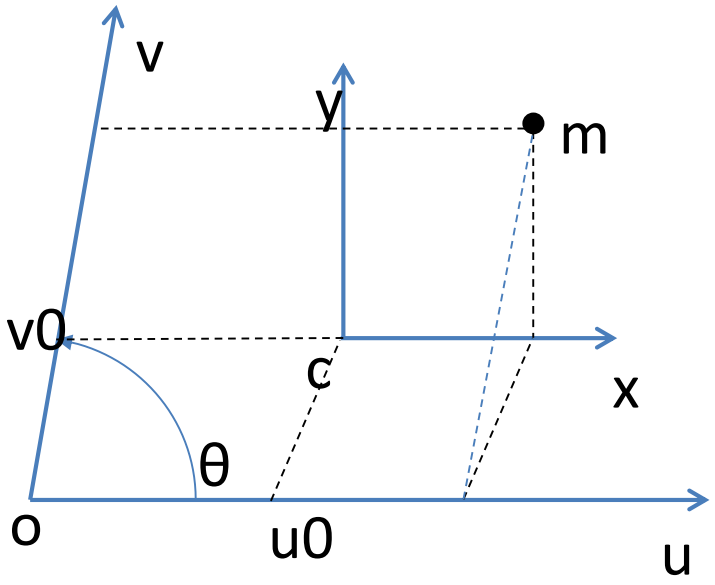

図 2.3 カメラ内部変数

ここで、まず x 軸を u 軸と並行に設定する。そして、u 軸と v 軸の単位長を座標系 c-xy を基準にそれぞれ ku, kv とする。さらに u 軸と v 軸のなす角度を θ と(必ずしも直角ではないと)し、座標系 o-uv における画像中心の座標を $[u0, v0]^T$ とする。

座標系 o-uv の座標を $m=[u, v]^T$、座標系 c-xy の座標を $ms=[x, y]^T$ とすると、以下の関係が成り立つ。

$$\tilde{m}=H\tilde{m}s$$

ここで

$$H=\begin{bmatrix}ku&-ku\cot\theta&u0\\0&kv/\sin\theta&v0\\0&0&1\end{bmatrix}$$

ku:座標系 c-xy を基準とした u 軸の単位長

kv:座標系 c-xy を基準とした v 軸の単位長

そして、元々の幾何関係を考えると、以下のような関係となる。

$$\lambda \tilde{m} = P \tilde{M}$$

または、

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} fku & -fku\cot\theta & u0 & 0 \\ 0 & fkv/\sin\theta & v0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

ここで焦点距離 f が単位長である正規化カメラを考えると上記の式は以下のように変形することができる。

$$\lambda \tilde{m} = A P_N \tilde{M}$$

または、

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha u & -\alpha u \cot\theta & u0 \\ 0 & \alpha v/\sin\theta & v0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

ここで、各記号は以下を示す。

(X,Y,Z)： ワールド座標系での3次元座標

(u,v)： 画像平面に投影された点の座標

(u0,v0)： 主点(通常は画像中心)

αu, αv： 縦軸と横軸のスケール

ここで再度、焦点距離fを導入するとともに、画像中心のズレまでも考慮すると最終的に以下の関係が導出される。

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f \cdot \alpha u & -f \cdot \alpha u \cdot \cot\theta & u0 \\ 0 & f \cdot \alpha v/\sin\theta & v0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r11 & r12 & r13 & t1 \\ r21 & r22 & r23 & t2 \\ r31 & r32 & r33 & t3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

または

$$\lambda m' = A\left[R \mid t\right] M'$$

行列 A は、カメラ行列またはカメラの内部パラメータ行列と呼ばれ、ビューに依存せずカメラ内部の変数のみによって構成される行列となっている。このため、一度推定すれば、焦点距離が変更されない限り、繰り返し使用することができる。

$$A = \begin{bmatrix} f \cdot \alpha u & -f \cdot \alpha u \cdot \cot\theta & u0 \\ 0 & f \cdot \alpha v/\sin\theta & v0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a11 & a12 & a13 \\ 0 & a22 & a23 \\ 0 & 0 & 1 \end{bmatrix}$$

さらに、実際のカメラレンズでは、主に半径方向の歪みや、わずかに円周方向の歪みを持っているため、厳密にはこれらの歪みを考慮する必要がある。これらは歪み係数と呼ばれ以下の形式で表現される。

(k1, k2, p1, p2 [,k3])

ここで、k1,k2,k3 は半径方向の歪み係数, p1,p2 は円周方向の歪み係数をそれぞれ示す。

また、[R|t]は外部パラメータ行列と呼ばれ、並進-回転の同次変換を行う行列となっている。この行列は、静的環境に対するカメラの動き、または逆に固定カメラの前の物体の剛体運動を表現する。

3　名前空間定義

カメラ機能共通インタフェースでは、固有の名前空間として「Img」を定義している。

本仕様書において規定するデータ型およびインタフェース定義において、名前空間の記載がない場合は、名前空間 Img に属しているものとする（ただし、Primitive 型は除く）。

4　データ型定義

カメラ機能共通インタフェースで使用するデータ型を以下に示す。

4.1　標準型

4.1.1 RTC::Time

時刻情報を格納するための型。OpenRTM-aist の標準型として BasicDataType.idl 内で定義されている。

表 4.1 RTC::Time

属性		
sec	unsigned long	秒単位の時刻情報
nsec	unsigned long	ナノ秒単位の時刻情報

4.2　型宣言

4.2.1 Vec3

実数の3次元ベクトルを表現するための型。double の配列として定義する。

```
typedef double Vec3[3];
```

4.2.2 Mat44

実数の4×4行列を表現するための型。double の4×4配列として定義する。

```
typedef double Mat44[4][4];
```

4.3 画像

4.3.1 ColorForamt

対象画像の色フォーマットを指定するため型。列挙型(enum 型)として定義する。なお，CF_XXX で定義される画像形式は，本インタフェースで定義した形式であり，その他の画像形式は，FourCC において定義されているピクセルフォーマットの形式に基づいた定義となっている．詳細は下記 HP を参照の上、利用すること。

FourCC ウェブサイト: http://www.fourcc.org

表 4.2 ColorFormat

CF_GRAY	グレースケール画像(*1)
CF_RGB	RGB 形式カラー画像(*2)
CF_JPEG	JPEG 形式画像(*3)
CF_PNG	PNG 形式画像(*4)
RGB	Windows ビットマップ形式画像(Reserved) (*5)
RLE8	ランレングス圧縮された 8 ビット RGB 画像
RLE	ランレングス圧縮された 4 ビット RGB 画像
RAW	非圧縮 RGB ビットマップ(Reserved) (*5)
RGBA	アルファ成分を含む非圧縮 RGB 形式カラー画像(Reserved) (*5)
RGBT	透明度を含む非圧縮 RGB 形式カラー画像(Reserved) (*5)
AYUV	アルファ成分を含んだ YUB 画像
CLJR	シーラス・ロジック社のフォーマットに基づく画像
CYUV	UYUV 形式と同じ形式の画像
GREY	Y800 や Y8 形式と同じ形式の画像
IRAW	インテル社の非圧縮 YUV 形式画像
IUYV	インタレース対応の UYVV 形式画像
IY41	インタレース対応の Y41P 形式画像
IYU1	IEEE1394 カメラのモード2で使われる 12 ビットの画像形式
IYU2	IEEE1394 カメラのモード 0 で使われる 24 ビットの画像形式
HDYC	UYVY と同じ形式
UYNV	Nvidia 社によって登録されている UYVY 画像形式
UYVP	YCbCr 4:2:2 の画像形式
UYVY	YUV 4:2:2 の画像形式
V210	10bit 4:2:2YCrCb の画像形式
V422	UYVY 形式のデータ列を逆さにした画像形式
V655	16bit YUV 4:2:2 形式の画像
VYUY	ATI 社により策定された YUV 形式の画像
Y422	UYVY と同じ形式

YUY2	UYVY に対して, YUV 4:2:2 とした画像形式
YUYV	YUY2 と同じ形式
YUNV	NVidia 者によって策定された YUY2 画像形式
YVYU	UYVY に対して YUV 4:2:2 とした画像形式
Y41P	YUV4:1:1 の画像形式
Y411	YUV4:1:1 の画像形式
Y211	YUV 画像形式の一種
Y41T	Y41P 画像形式の一種
Y42T	UYVY 画像形式の一種
YUVP	YCbCr4:2:2 の画像形式の一種
Y800	グレースケール画像
Y8	Y800 と同形式
Y16	16 ビット非圧縮グレースケール画像

(*1) CF_GRAY は OpenCV の CV_8UC1 に対応するグレー画像を想定したフォーマット
(*2) CF_RGB は OpenCV の CV_8UC3 に対応する 24bit カラー画像を想定したフォーマット
(*3) CF_JPEG は OpenCV の imencode において, JPEG 圧縮した画像を想定したフォーマット
(*4) CF_PNG は OpenCV の imencode において, PNG 圧縮を想定したフォーマット
(*5) FourCC で規定されていながら, ビット数, 画素値の並びに自由度があるため, ColorFormat に記載は行うが, 使用の推奨はしない.

4.3.2 ImageData

対象画像の生データを保持するための型。一般に, 画像の生データはサイズが大きくなる傾向がある。このため、転送速度を向上させるために octet 型として定義する。

表 4.3 ImageData

属性		
width	long	対象画像の幅
height	long	対象画像の高さ
format	ColorFormat	対象画像データの色フォーマット
raw_data	sequence<octet>	対象画像の各ピクセルの生データ

4.3.3 TimedImage

RT コンポーネント間で画像データをやりとりするための型。ファイルに保存された画像データを読み込んで RT コンポーネント間で送受信を行う際などに利用する。

表 4.4 TimedImage

属性		
tm	RTC::Time	対象画像を送信した時刻

data	ImageData	対象画像の生データ
error_code	long	対象画像を取得する際に発生したエラー情報

※RTC::Time 型は OpenRTM-aist にて事前定義されている標準型。

※error_code に格納するエラー情報のコード体系などについては未定義。

4.4 カメラ画像

4.4.1 CameraIntrinsicParameter

対象画像を取得する際に使用したカメラの内部パラメータを格納するための型。内部パラメータ行列の要素と歪みパラメータを保持する。

表 4.5 CameraIntrinsicParameter

属性		
matrix_element	double[5]	カメラパラメータを格納するための配列
distortion_coefficient	sequence<double>	使用したカメラの歪みパラメータ

matrix_element は、使用したカメラの内部パラメータ行列 A の要素を以下の形式で格納する。

$$A = \begin{bmatrix} a11 & a12 & a13 \\ 0 & a22 & a23 \\ 0 & 0 & 1 \end{bmatrix}$$

$$a11 = matrix\_element[0]$$
$$a12 = matrix\_element[1]$$
$$a22 = matrix\_element[2]$$
$$a13 = matrix\_element[3]$$
$$a14 = matrix\_element[4]$$

distortion_coefficient は、カメラの歪みパラメータを (k1, k2, p1, p2 [,k3])の形式で格納する。

4.4.2 CameraImage

単一のカメラを使用して取得した画像情報を格納するための型。

表 4.6 CameraImage

属性		
capture_time	RTC::Time	対象画像を撮影した時刻
image	ImageData	対象画像の生データ
intrinsic	CameraIntrinsicParameter	使用したカメラの内部パラメータ
extrinsic	Mat44	使用したカメラの外部パラメータ

Extrinsic には、対象画像を撮影する際に使用したカメラの外部パラメータを、Mat44 型を用いて4×4の同次変換行列 T の形式で格納する。

$$T = \left[\begin{array}{c|c} R & t \\ \hline 0 & 1 \end{array}\right] = \begin{bmatrix} r11 & r12 & r13 & t0 \\ r21 & r22 & r23 & t1 \\ r31 & r32 & r33 & t2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

R = 姿勢変換行列
t = 位置ベクトル

4.4.3 TimedCameraImage

単一のカメラを使用して取得した画像情報を RT コンポーネント間でやりとりするための型。

表 4.7 TimedCameraImage

属性		
tm	RTC::Time	対象画像を送信した時刻
data	CamraImage	カメラで取得した画像情報
error_code	long	対象画像を取得する際に発生したエラー情報

※RTC::Time 型は OpenRTM-aist にて事前定義されている標準型。

※error_code に格納するエラー情報のコード体系などについては未定義。

4.5 複数カメラ画像

4.5.1 MultiCameraImage

複数のカメラを使用して取得した画像情報を格納するための型。

表 4.8 MultiCameraImage

属性		
image_seq	Sequence<CameraImage>	各カメラで撮影した画像情報。キャリブレーションデータも含む。
camera_set_id	long	使用したカメラセットの ID

※camera_set_id のコード体系については未定義。

4.5.2 TimedMultiCameraImage

複数のカメラを使用して取得した画像情報を RT コンポーネント間でやりとりするための型。

表 4.9 TimedMultiCameraImage

属性		
tm	RTC::Time	対象画像を送信した時刻
data	MultiCamraImage	複数カメラで取得した画像情報
error_code	long	対象画像を取得する際に発生したエラー情報

※RTC::Time 型は OpenRTM-aist にて事前定義されている標準型。

※error_code に格納するエラー情報のコード体系などについては未定義。

4.6　カメラデバイス情報

4.6.1 NamedValue

カメラデバイスの追加プロファイル情報を格納するための型。

表 4.10 NamedValue

属性		
Name	string	プロファイルの名称
Value	String	値

4.6.2 CameraDeviceProfile

カメラデバイスのプロファイルを格納するための型。

表 4.11 TimedMultiCameraImage

属性		
devtypeid	string	カメラデバイスの ID
guid	string	
unit	short	
vendor_name	string	デバイスの製造者名
model_name	string	デバイスの型番など
intrinsic	CameraIntrinsicParameter	使用したカメラの内部パラメータ
properties	NVList	デバイスへの付加情報
error_code	long	対象画像を取得する際に発生したエラー情報

※NVList は NamedValue の sequence

※error_code に格納するエラー情報のコード体系などについては未定義。

共通インタフェース定義

以下にカメラ機能共通インタフェースで使用する共通インタフェースの定義を示す。

4.7　データポート

4.7.1 画像データインタフェース

カメラキャプチャモジュールが、各種画像処理モジュールや画像表示モジュールに画像データを受け渡すためのインタフェースである。

画像データを表現するための型は、用途に応じて階層化された形で定義されているが、通常は TimedMultiCameraImage 型を使用することを想定している。

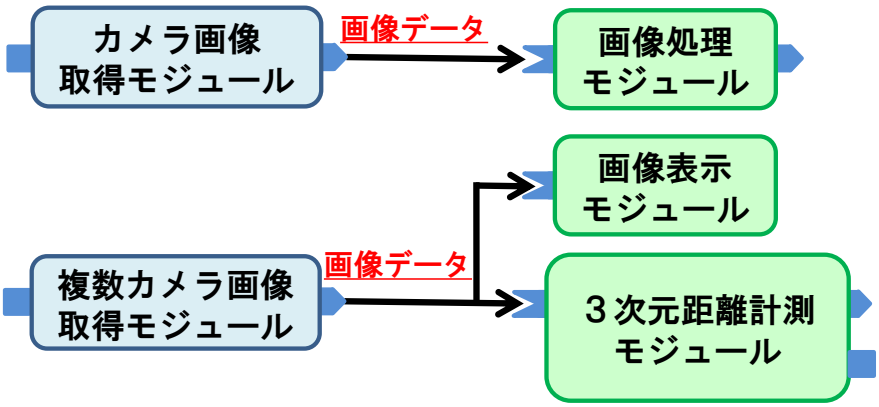

図　4.1　画像データインタフェース

4.8　サービスポート

4.8.1 CameraCaptureService

カメラキャプチャモジュールに画像情報の取得を指示するためのインタフェース。

表　4.12 CameraCaptureService

メソッド				
take_one_frame		oneway void		1枚の画像取得を実行する
take_multi_frames		oneway void		複数枚の画像取得を実行する
	in	num	long	撮影枚数
start_continuous		oneway void		連続での画像取得を開始する
stop_continuous		oneway void		連続での画像取得を停止する
getProfile		void		カメラのプロファイルを取得する
	out	profile	CameraDeviceProfile	カメラデバイスのプロファイル情報

各操作は非同期なオペレーション(oneway)として定義されている。

複数枚撮影、連続撮影を実行する際の撮影時間間隔については、各カメラおよび RT コンポーネントの性能に依存する。

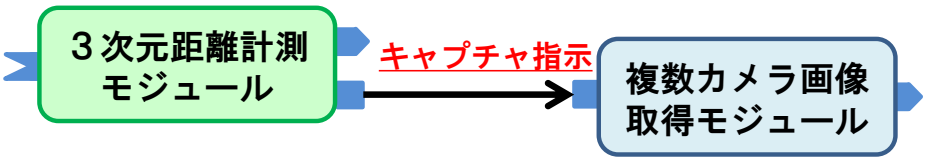

図　4.2 CameraCaptureService インタフェース

5　共通インタフェースを利用したシステム構築例

5.1　アピアランスベース物体位置・姿勢推定コンポーネント

○開発者:国立大学法人 大阪大学 大学院基礎工学研究科

○詳細 URL:http://www.openrtm.org/openrtm/ja/project/NEDO_Intelligent_PRJ_ID257

○概要

予め取得したモデル画像を基に、アピアランスベースで任意物体の検出および位置・姿勢の推定を行うコンポーネントである。物体検出および位置・姿勢推定には SIFT 特徴を用いており、日常環境における照明条件の変化や物体の見え方におけるスケール変化、回転変化に強固な物体検出を実現している。

本コンポーネントを利用したシステム構成例を以下に示す。図中の赤字の部分が、本仕様書で規定している共通インタフェースを使用している部分である。

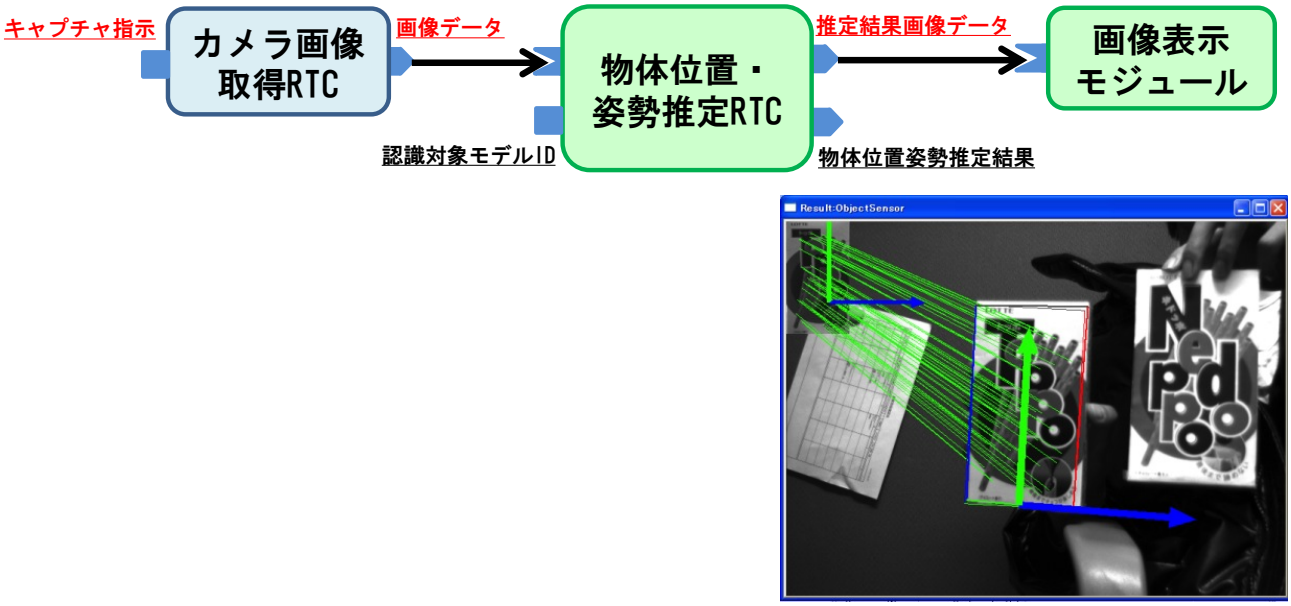

図 5.1 アピアランスベース物体位置・姿勢推定コンポーネント

5.2　作業対象認識モジュール群

○開発者:独立行政法人 産業技術総合研究所 知能システム研究部門 タスクビジョン研究グループ

○詳細 URL:http://www.openrtm.org/openrtm/ja/project/NEDO_Intelligent_PRJ_ID367

○概要

複数台のカメラによって撮影したデジタル画像データから、指定された作業対象物の3次元位置・姿勢を出力するモジュール群である。位置・姿勢の推定は、三角測量の原理を応用してシーンの距離を計測し、3次元のモデルと照合することで実現している。

本モジュール群のシステム構成例を以下に示す。図中の赤字の部分が本仕様書で規定している共通インタフェースを使用している部分である。

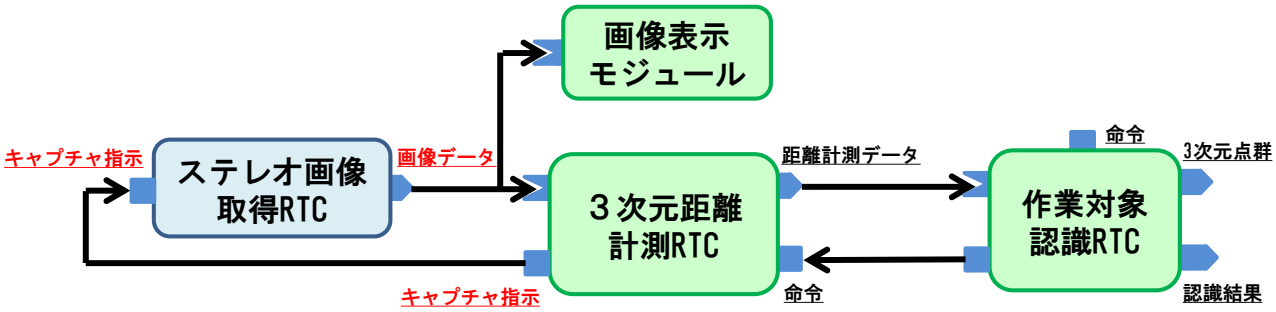

図 5.2 作業対象認識モジュール群

図 5.3 ステレオ画像取得 RTC の画像データ出力例

6 CORBA IDL

カメラ機能共通インタフェースの IDL 定義を以下に示す。

6.1 Img.idl

```
/* version 2.0 */

#ifndef IMG_IDL
#define IMG_IDL

#include <BasicDataType.idl>

module Img {
	//=============================================================
	// Definition of basic matrix for image.
	//=============================================================
		typedef double Vec3[3];
		typedef double Mat44[4][4];

		//=============================================================
		//Image Data Structure
		//=============================================================

		//-------------------------------------------------------------
		// Color Formats Definition
		// The parameters of CF_XXX are defined by this idl.
		// The other parameters are cited from FourCC definition.
		// To obtain the detail information about FourCC formats, please visit
		// following website:
		// http://www.fourcc.org/
		//-------------------------------------------------------------
		enum ColorFormat
		{
			CF_UNKNOWN,		//Unknown Color Format
			CF_RGB,			//RGB raw data format
			CF_GRAY,		//8bit gray image data format
			CF_JPEG,			//JPEG image format
			CF_PNG,			//PNG image format
```

```
//Paramter definitions based on FourCC
RGB,      //Basic Windows bitmap format
RLE8,     //Run length encoded 8bpp RGB image
RLE,      //Run length encoded 4bpp RGB image
RAW,      //Uncompressed RGB bitmaps
RGBA,     //Raw RGB with alpha
RGBT,     //Raw RGB with a transparency field
AYUV,     //Combined YUV and alpha
CLJR,     //Cirrus Logic format with 4 pixels packed into a u_int32
CYUV,     //Essentially a copy of UYUV except that the sense of the height is reserved
GREY,     //Apparently a duplicate of Y800 and Y8
IRAW,     //Intel uncompressed YUV
IUYV,     //Interlaced version of UYUV
IY41,     //Interlaced version of Y41P
IYU1,     //12bit format used in mode 2 of the IEEE 1394 Digital Camera 1.04 spec.
IYU2,     //24 bit format used in mode 0 of the IEEE 1394 Digital Camera 1.04 spec.
HDYC,     //YUV 4:2:2 (Y sample at every pixel, U and V sampled at every second
          pixel horizontally on each line)
UYNV,     //A direct copy of UYVY registered by NVidia to work around problems in
          some old codecs which did not like hardware which offered more than 2
          UYVY surfaces.
UYVP,     //YCbCr 4:2:2 extended precision 10-bits per component in U0Y0V0Y1
          order
V210,     //10-bit 4:2:2 YCrCb equivalent to the Quicktime format of the same name
V422,     //This is an upside down version of UYVY
V655,     //16 bit YUV 4:2:2 format registered by Vitec Multimedia
VYUV,     //Duplicate of YUV2
YUNV,     //A direct copy of YUY2 registered by NVidia to work around problems in
          some old codecs which did not like hardware which offered more than 2
          YUY2 surfaces
YVYU,     //YUV 4:2:2 as for UYVY but with different component ordering within the
          u_int32 macropixel
Y41P,     //YUV 4:1:1 with a packed, 6 byte/4 pixel macroblock structure
Y211,     //Packed YUV format with Y sampled at every second pixel across each line
          and U and V sampled at every fourth pixel
Y41T,     //Format as for Y41P but the lsb of each Y component is used to signal pixel
          transparency
Y42T,     //Format as for UYVY but the lsb of each Y component is used to signal
          pixel transparency
YUVP,     //YCbCr 4:2:2 extended precision 10-bits per component in Y0U0Y1V0
          order
```

```
        Y800,     //Simple, single Y plane for monochrome images
        Y8,       //Duplicate of Y800
        Y16       //16-bit uncompressed grayscale image
    };

    //============================================================
// Camera Image Data Structure
    //============================================================

    //------------------------------------------------------------
    // Image Data Structure for still image
    //------------------------------------------------------------
    struct ImageData
    {
      long width;
      long height;

      ColorFormat format;
      sequence<octet> raw_data;
    };

    //------------------------------------------------------------
    // Camera Intrinsic Parameter Structure
    //------------------------------------------------------------
    struct CameraIntrinsicParameter
    {
      double matrix_element[5];
      sequence<double> distortion_coefficient;
    };

    //------------------------------------------------------------
    // Camera Image Structure
    //------------------------------------------------------------
    struct CameraImage
    {
      RTC::Time captured_time;
      ImageData image;
      CameraIntrinsicParameter intrinsic;
      Mat44 extrinsic;
    };
```

```
//------------------------------------------------------------
// Timed Camera Image Structure
// This structure includes time stump.
//------------------------------------------------------------
struct TimedCameraImage
{
   RTC::Time tm;
   CameraImage data;
   long error_code;
};

//============================================================
// Multi Camera Image Data Structure
//============================================================
//------------------------------------------------------------
// Multi Camera Image Structure
//------------------------------------------------------------
struct MultiCameraImage
{
   sequence<CameraImage> image_seq;
   long camera_set_id;
};

//------------------------------------------------------------
// Time Multi Camera Image Structure
// This structure includes time stump.
//------------------------------------------------------------
struct TimedMultiCameraImage
{
   RTC::Time tm;
   MultiCameraImage data;
   long error_code;
};
```

```
        //=============================================================
        // Camera Device Profile Structure
        //=============================================================
        struct NamedValue
        {
                string name;
                string value;
        };

        typedef sequence<NamedValue> NVList;

        struct CameraDeviceProfile
        {
                string devtypeid;
                string guid;
                short unit;
                string vendor_name;
                string model_name;
                CameraIntrinsicParameter intrinsic;
                NVList properties;
        };

        //=============================================================
        // Camera Control Service Interface
        //=============================================================
        //-------------------------------------------------------------
        // Camera Control Service Interface for image capture
        //-------------------------------------------------------------
        interface CameraCaptureService
        {
                oneway void take_one_frame();
                oneway void take_multi_frames(in long num);
                oneway void start_continuous();
                oneway void stop_continuous();
                void getProfile(out CameraDeviceProfile profile);
        };

}; /* module */

#endif /* IMG_IDL */
```

7　参考文献

[1] 共通カメラインタフェースの提案:大原賢一, 川端聡, 河井良浩:第 29 回日本ロボット学会学術講演会予稿集(2011)

[2] 3次元ビジョン:徐剛, 辻三郎:共立出版(1998)

8　謝辞

本仕様は NEDO 次世代ロボット知能化技術開発プロジェクトにおいて策定されました.