学部 3 年生プロジェクト

固定カメラベースの AR：カメラの位置姿勢推定による現実空間と仮想空間の重ね合わせと，カメラによるモーキャプアバターのユーザへの重畳

2022 年度

岐阜大学　工学部

電気電子・情報工学科　木島研究室

藤嶋 駿輔　今川 勝喜　吉川 柊太　渡辺 隼平

目次

第1章　はじめに

1.1　本取り組みの目標

本取り組みの目標は以下の図 1.1 のように現実世界の人間と同じ動きをする仮想世界内のアバターが現実世界に存在するかのような映像を提示することである.

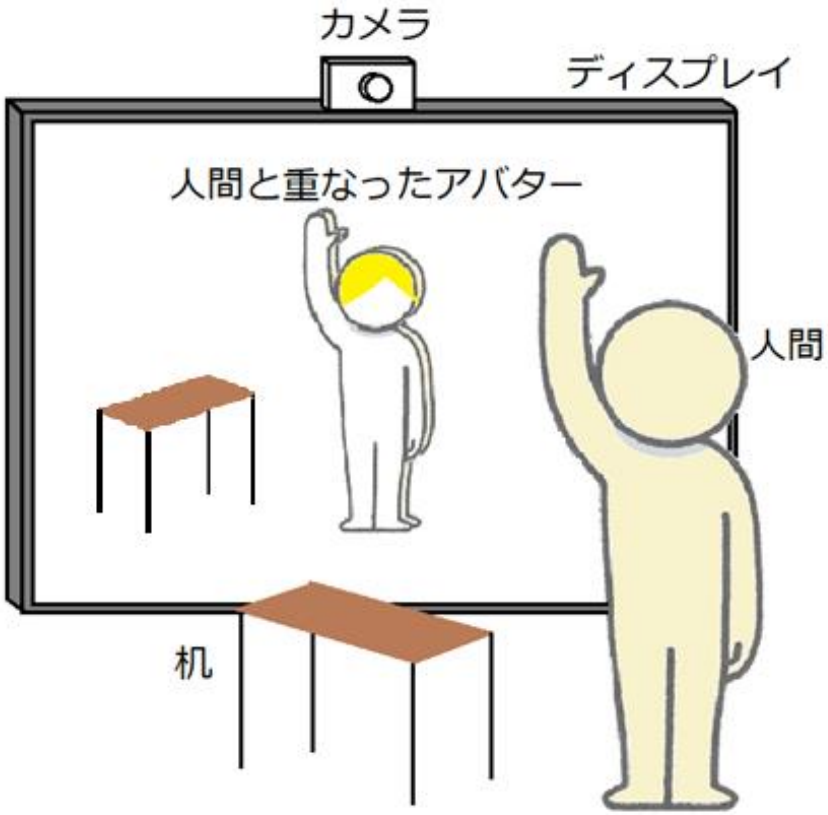


図 1.1　完成想定図

これは，現実世界に存在するカメラと仮想世界のカメラの位置と姿勢を一致させること，及びプレイヤーに対してモーションキャプチャを行うことによって達成される.

1.2　システム構成図

本取り組みでは，PC，HTC VIVE コントローラー，HTC VIVE 用ベースステーション，Web カメラ，ディスプレイ，Unity を用いた．

PC にゲームエンジンソフトウェアである Unity をインストールし，Steam VR plugin を用いて VR 空間をプレイするための HMD(Head Mounted Display)である HTC VIVE を使用できるようにした．そこに USB 接続で HTC VIVE が接続されている．また，PC と HDMI ケーブルで接続されているディスプレイの左右に，HTC VIVE のヘッドセットやコントローラーをトラッキングするための HTC VIVE 用ベースステーションを設置した．ベースステーションは VR 空間で遊ぶためにプレイエリアと呼ばれる領域を設定し，HTC VIVE のトラッキングはプレイエリア内でのみ行われる．

図 1.2 がその様子である．カメラと PC は USB，PC とディスプレイは HDMI，PC と HTC VIVE 用ベースステーション・HTC VIVE コントローラーと PC は Bluetooth で接続している．

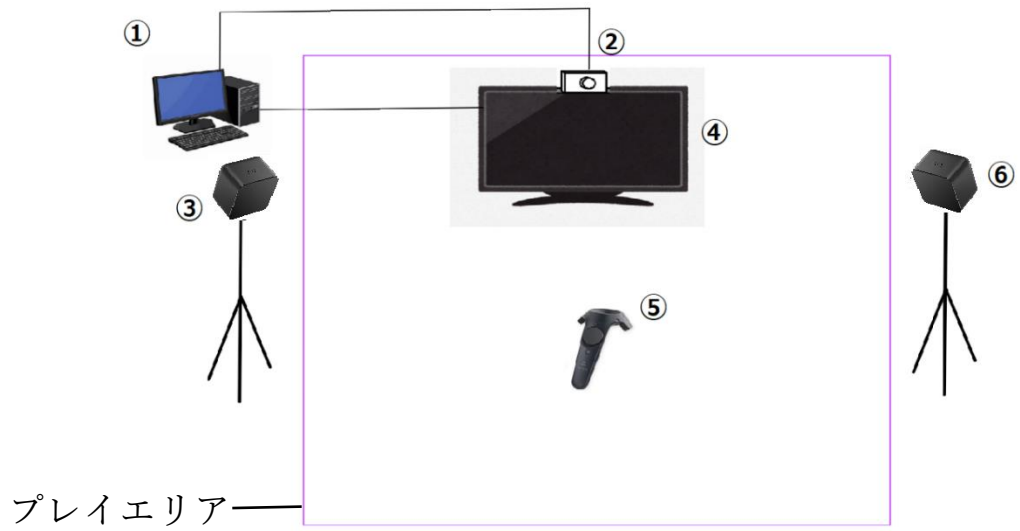


図 1.2　デバイス配置図

①PC
②Web カメラ
③HTC VIVE 用ベースステーション
④ディスプレイ
⑤HTC VIVE コントローラー
⑥HTC VIVE 用ベースステーション

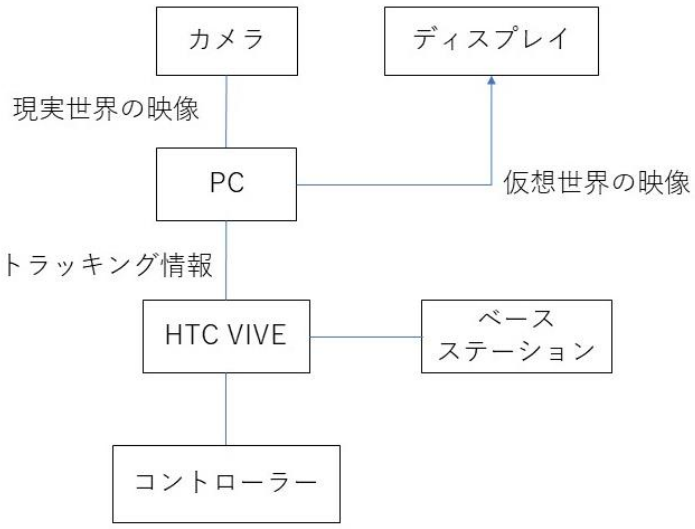


図 1.3　システム構成図

1.3 節で使用した機械,　システムについて詳しく述べていく.

1.3　各要素の諸元

表 1.1：ゲームエンジン

Unity （バージョン 2021.3.12）	
開発者	Unity Technologies 社

表 1.2：トラッカー（1）

HTC VIVE 用コントローラー	
通信速度	250[Hz] ~ 1000[Hz]

表 1.3：トラッカー（2）

HTC VIVE 用ベースステーション	
計測範囲	水平面 2[m]×2[m]以上・高さ 2[m]
センサ	レーザーポジションセンサ・加速度センサ ジャイロセンサ
視野	水平視野 150[deg]・垂直視野 110[deg]
精度	位置誤差 1[mm]以下

表 1.4：Steam 上で HTC VIVE を使用するための機能

Steam VR Plugin

表 1.5：PC

CPU	Intel®Core i5-6600
GPU	NVIDIA GeForce GTX 1060 3GB
メモリ	8.0 GB

表 1.6：Web カメラ

製造社	BUFFALO
製品型番	BSW20KM11BK
外形寸法	幅 48[mm]×高さ 105[mm]×奥行き 50[mm]
質量	70 [g]
視野角度	120 [deg]
最大解像度	1920×1080 [ピクセル]
有効画素数	200 万画素
最大フレームレート	30 [fps]

図 1.4　Web カメラ

表 1.7：ディスプレイ

製造社	Panasonic
製品型番	THP50VT2
外形寸法	幅 1224[mm]×高さ 810[mm]×奥行き 335[mm]
質量	30.5 [kg]
画素数	約 200 万画素（1920×1080）

図 1.5　ディスプレイ

1.4　透視投影変換

透視投影変換については文献[1]，[2]，[3]，[4]を参考にした．

透視投影変換の前段階として透視投影の話をする．　透視投影とは 3 次元空間内にある物体を投影面に表示する際に，人間の視覚と同様に近くの物体は大きく，遠くの物体は小さく映すことである．また，物体が移動する際には近くの物体は投影平面に占める移動幅が大きく，遠くの物体は投影面に占める移動幅が小さくなる。これは 3 次元物体をある観測点に集まるように投影線を引き，投影を行うことで実現できる．

このようなことを実現するために物体の座標を変換することを透視投影変換という．

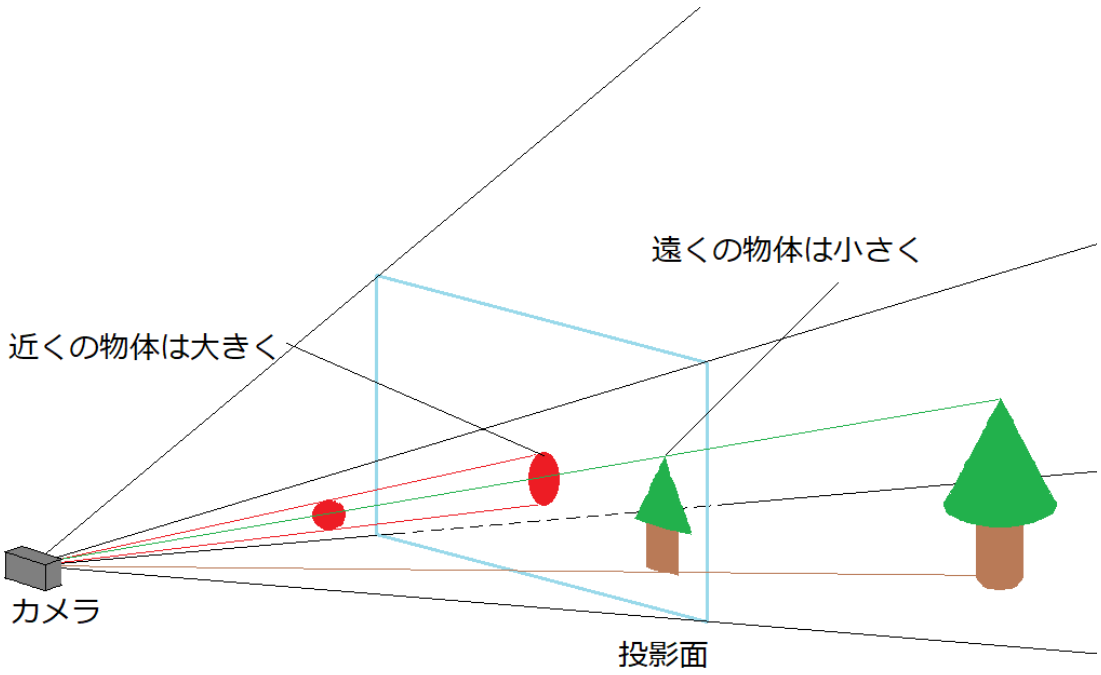


図 1.6　透視投影

透視投影変換の式は一般的に以下の式で表される．

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$は画像座標点と呼ばれ，変換後の画面上の座標点を表す．

$\begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$は世界座標点と呼ばれ，変換前の 3 次元空間上の絶対的な座標点を表している．

つまり，上記の透視投影変換の一般式は世界座標から画像座標への変換であるということを表しているが，実際には世界座標からカメラ座標への剛体変換とカメラ座標から画像座標への透視投影変換で構成されている．

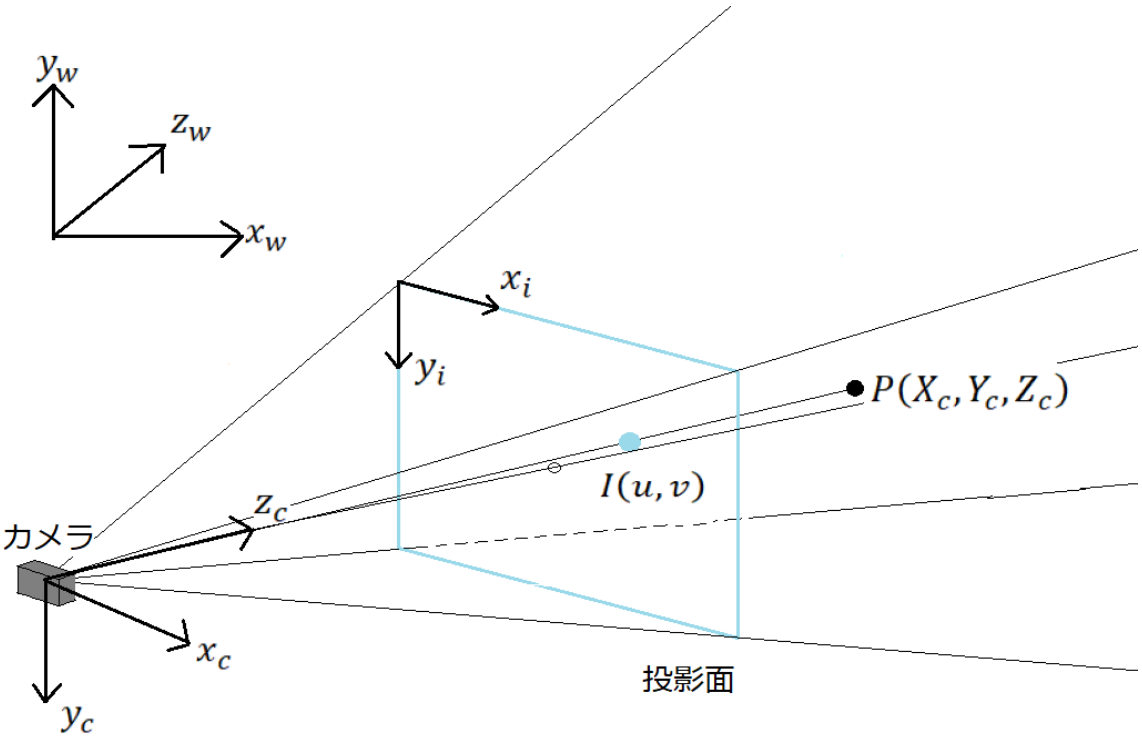


図 1.7　物点の投影の様子

図 1.7 のような 3 次元空間の絶対的な座標系である世界座標系$\left(x_w,\ y_w,\ z_w\right)$，カメラのレンズの中心を原点とし，光軸方向を$z$軸とした座標系であるカメラ座標系$\left(x_c,\ y_c,\ z_c\right)$，投影後の 2 次元の画面上の座標系である画像座標系$\left(x_i,\ y_i\right)$において，カメラ座標系における物点$P\left(X_c,\ Y_c,\ Z_c\right)$を投影面に$I(u,\ v)$となるよう透視投影変換することを考える．

図 1.8 のように物点$P\left(X_c,\ Y_c,\ Z_c\right)$を投影面上にあるように奥行き$Z_c$で割り，焦点距離$f$（ f_xとf_yはほぼ近似できるものとする$(f_x \fallingdotseq f_y = f)$）を掛ける．さらに原点が投影面の左上に来るように$c_x,\ c_y$だけ足している．これが透視投影変換である．この式は，

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c/Z_c \\ Y_c/Z_c \\ 1 \end{bmatrix}$$

と表すことができる．さらに両辺にZ_cをかけて，

$$Z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \quad \cdots (1)$$

となる．

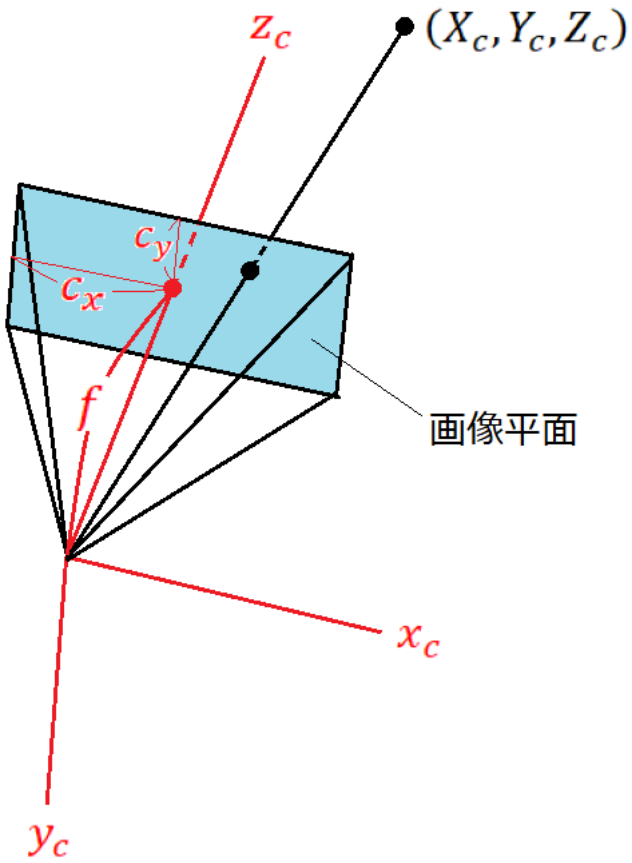
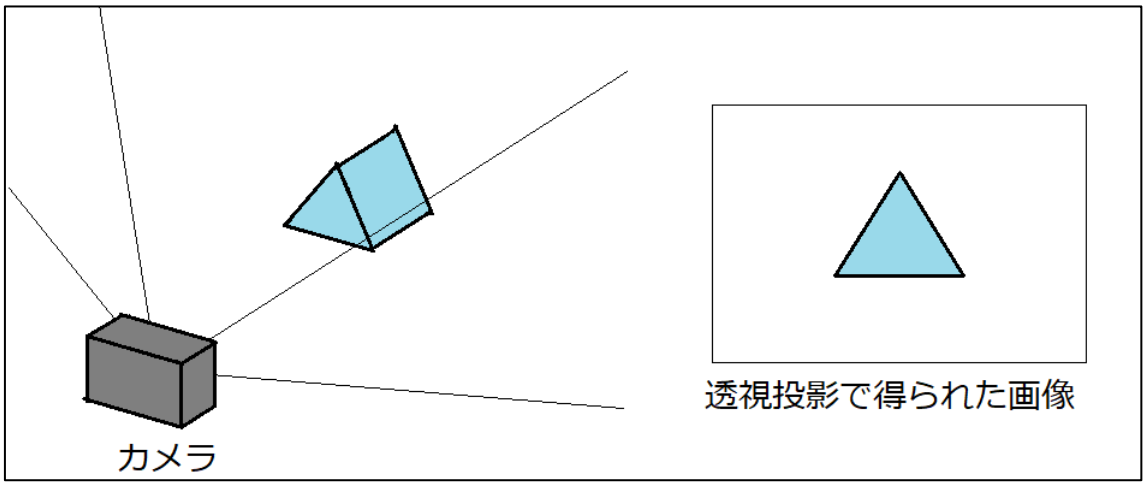
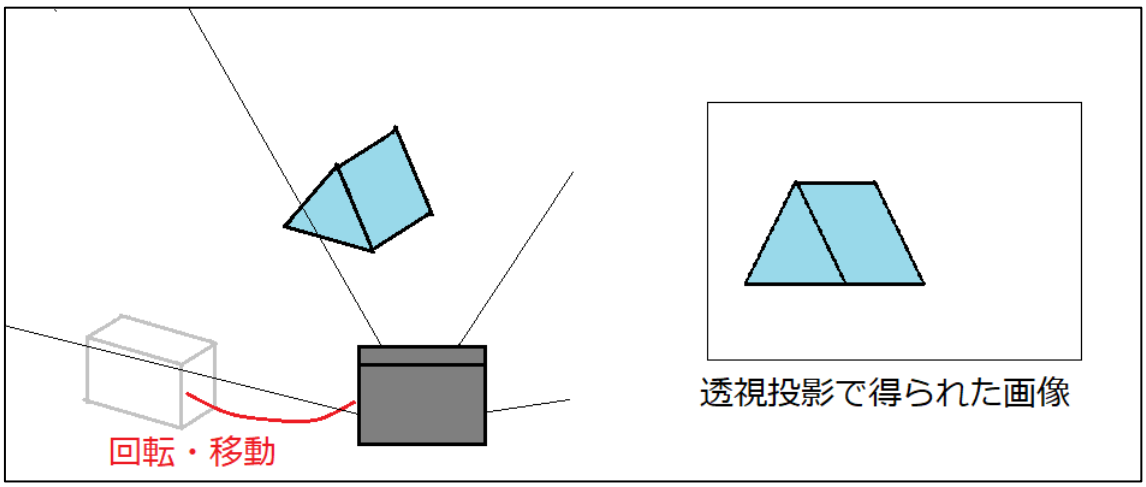


図 1.8　カメラ座標系から画像座標系への変換

ここで，物点$P\left(X_c,\ Y_c,\ Z_c\right)$はカメラ座標系で表されているが，実際の 3 次元空間の座標系である世界座標系上では$\left(X_w,\ Y_w,\ Z_w\right)$と表される．物点 P を透視投影する際に世界座標系の原点から透視投影するのとカメラ座標系の原点から透視投影するのでは結果が違ってくる．透視投影変換はカメラ座標系で定義されるので，透視投影変換をする前に，物点 P の座標系を世界座標系からカメラ座標系へと変換しなければならない．世界座標系からカメラ座標系への変換には，世界座標系におけるカメラ座標系の基底ベクトルと世界座標系の基底ベクトルが等しくなるように世界座標系を回転させ，原点が同じになるように世界座標系の原点を移動させれば良いが，物点 P に逆回転・逆移動をさせたとしても，物点 P の見え方は同じとなる．すなわち，透視投影変換の結果が等しくなる．





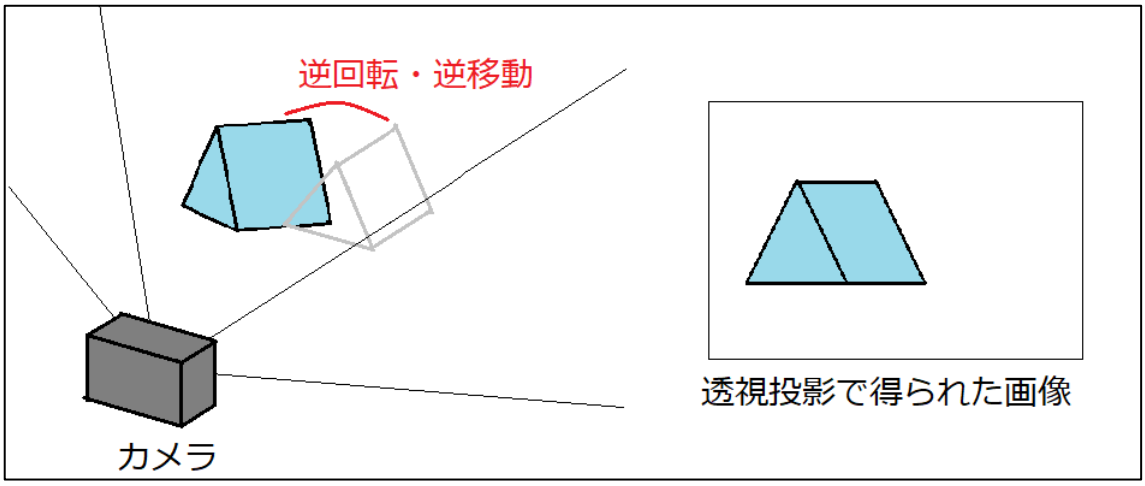


図 1.9　座標変換と見え方

図 1.10 のように世界座標系$\left(x_w,\ y_w,\ z_w\right)$をカメラの姿勢を表すベクトル$\left(x_c,\ y_c,\ z_c\right)$と同じ向きとなる$\left(x'_w,\ y'_w,\ z'_w\right)$に回転させると，物点 P はカメラから見た向きとなる．さらに，座標系の原点O_wをカメラの位置O_cへと移動させると，物点 P はカメラから見た位置となる．
すなわち，カメラ座標系への変換が行われる．これが世界座標系からカメラ座標系への変換を表す回転と並進の剛体変換である．

世界座標系の回転・移動は物点の逆回転・逆移動と同義のため，上の変換を式に表すと

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \quad \cdots(2)$$

となる．これは世界座標$\begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix}$に回転を表す行列$\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$をか

け，移動させる分$\begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}$を足す計算を同次座標系で表した式である．

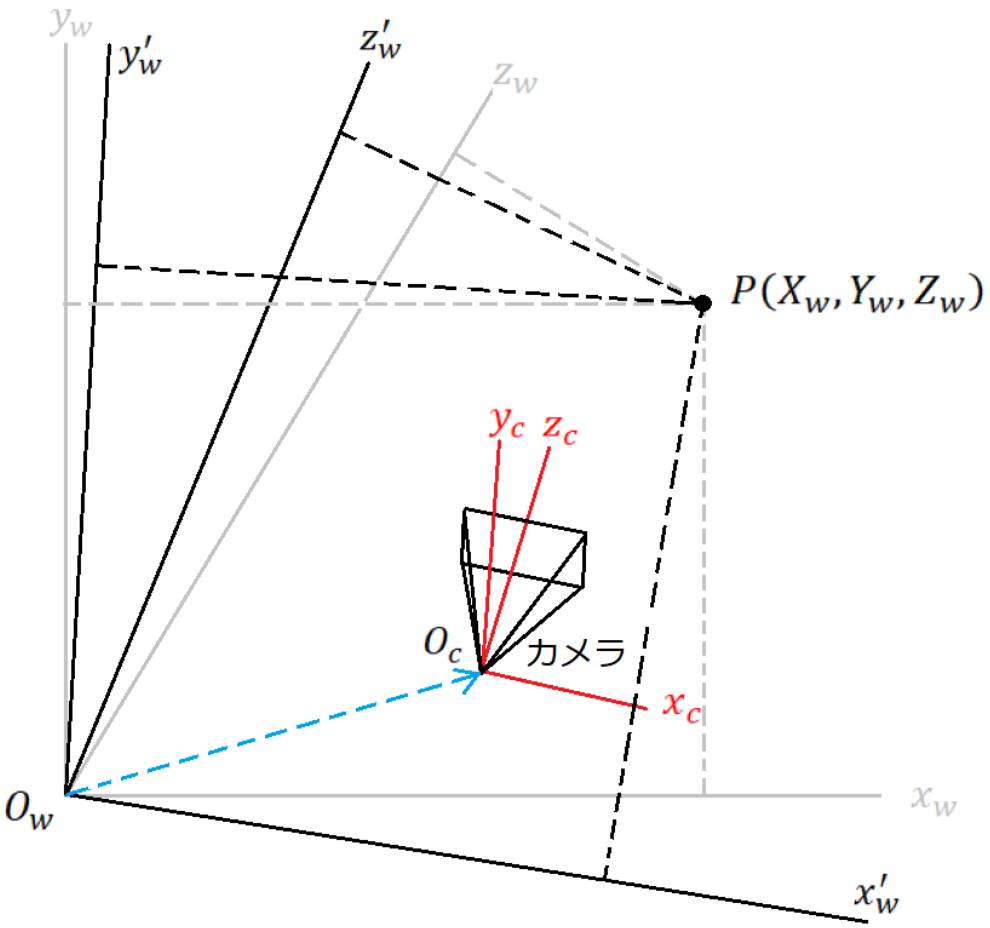


図 1.10 世界座標系からカメラ座標系への変換

(1)に(2)を代入すると，

$$Z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

となる．$s = Z_c$とすると，上記の一般式

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

となる.

第2章　作成準備

2.1　作成における準備

ここでは現実世界のカメラと仮想世界のカメラとを一致させる必要性について説明していく.

まず現実空間と仮想空間上にそれぞれ立方体があることを考える．これを現実空間上のディスプレイでこの 2 つの立方体を全く同じ位置・姿勢・大きさで重ねるために現実空間上と仮想空間上で立方体が全く同じ位置・姿勢・大きさであることを前提とする.

この前提条件の下，現実空間上のディスプレイで 2 つの立方体を完全に重ねて表示するための方法を考える．現実空間と仮想空間において 2 つの立方体は位置・姿勢・大きさを一致しているため，現実空間と仮想空間それぞれのカメラの画角とアスペクト比が一致しているならば，この状況は一方の空間内にある立方体を 2 つのカメラが撮影しているのと同義である.

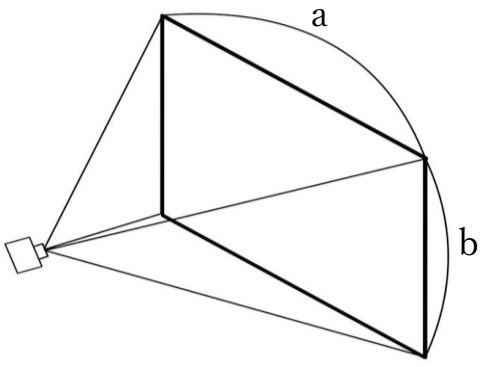


図 2.1　アスペクト比(a : b)

そのような状況で二つのカメラが撮影する画像を一致させるためには，立方体を同じ位置・姿勢から見ることが必要である．

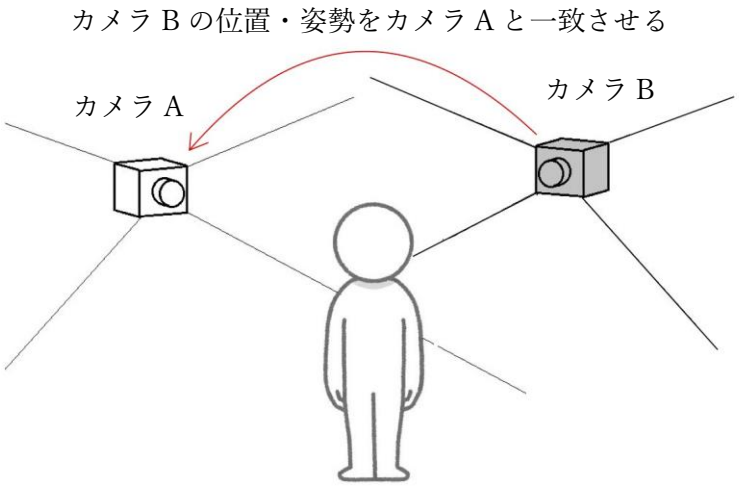


図 2.2　同一空間における 2 つのカメラの一致

つまり，現実空間と仮想空間においてカメラの位置・姿勢を一致させることで，現実空間上のディスプレイでこの 2 つの立方体を全く同じ位置・姿勢・大きさで重ねることが可能となる．

これは立方体だけでなく人間においても同様である．　前提条件として先に掲げた，仮想世界の物体と現実空間の物体が同じ位置・姿勢・大きさであることについても，人間の場合は仮想空間上でモーションキャプチャを用いた 3D アバターを人間に適用し，姿勢・大きさを一致させる．　位置については HTC VIVE のコントローラーを人間に取り付け，トラッキング機能を用いることによって一致させることが可能である．

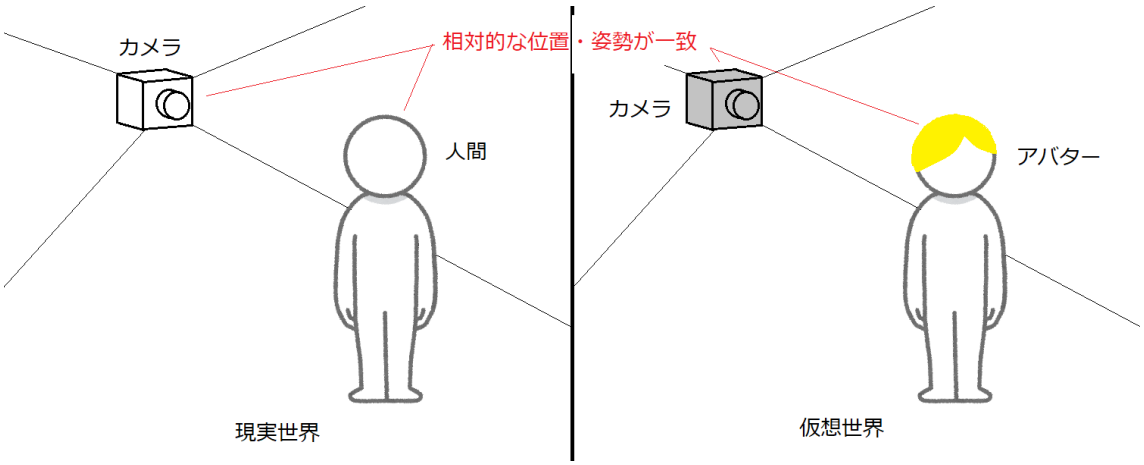


図 2.3　現実世界と仮想世界が一致しているイメージ

このように前提条件は達成出来ることから，以降の章では実際に現実世界のカメラと仮想世界のカメラの位置，姿勢を一致させる手順について記載していく．

2.2　現実のカメラの位置と姿勢を推定するための準備

2.2.1　現実世界におけるコントローラーの位置の計測

現実世界と仮想世界を一致させるためには，現実世界のカメラの位置と姿勢を仮想世界における座標系で表現する必要がある．そこで，現実世界の位置や姿勢を仮想世界上の座標系で表すことができる HTC VIVE コントローラーのトラッキング機能を使用することとした.

2.2.2　仮想世界の環境

筆者らは Unity（2021.3.12f1）を用いて仮想世界を構築した．仮想世界にはモーションキャプチャ用のアバター（図 2.4),

図 2.4　モーションキャプチャ用のアバター

© Unity Technologies Japan/UCL

カメラ，　現実世界の映像を映す仮想空間のディスプレイの役割を果たす Canvas（図 2.5），プレイエリアがある．

図 2.5　Canvas に表示される現実世界のカメラで撮影されている映像

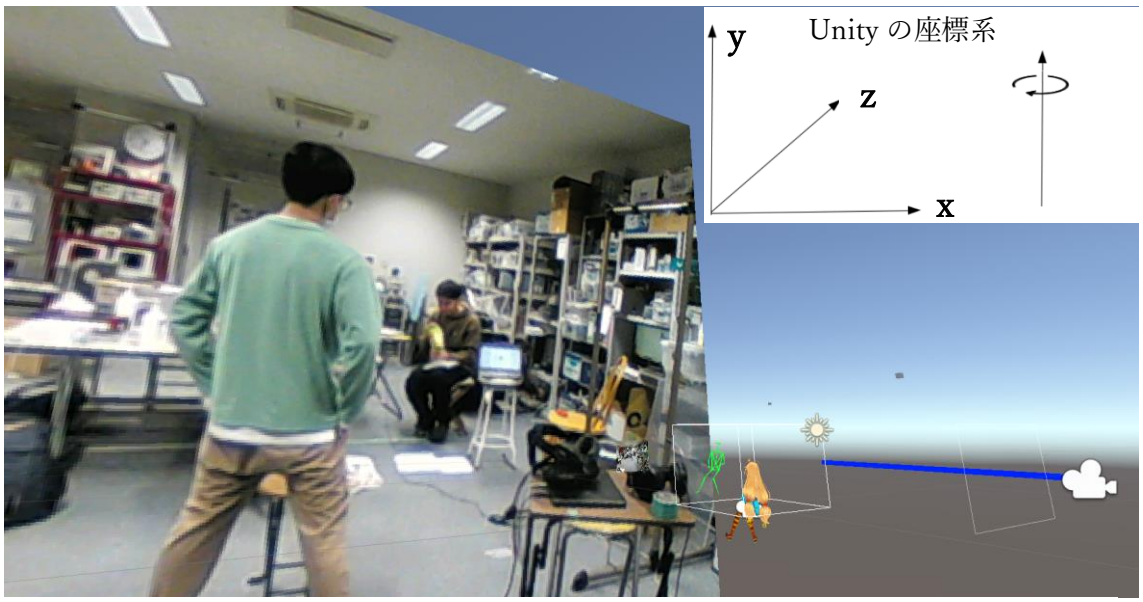


図 2.6　仮想空間の環境

左側に Canvas，中央にプレイエリア，右側に仮想空間のカメラ

上記のカメラの画角とアスペクト比は現実空間に設置したカメラの画角とアスペクト比と同じ値とした．また，Unity で用いられる座標系は左手系である．今後記載する仮想空間上のカメラの座標系は全て左手系(x，y，z)で表現する．また，仮想空間上の物体の回転については Unity の Quaternion を用いる。Quaternion は回転軸を指定し，その軸に対して左ねじの法則で回転を行う方式である。以降現実空間のカメラの座標系，回転を表現する際にもそれぞれ Unity と同様の左手系，Quaternion を用いて表現する．

2.2.3　コントローラーの計測点

HTC VIVE コントローラーのモデルは Unity に導入してあり，コントローラーを用いる場合はそれを表示する事が出来る．これは HTC VIVE の機能の一つである．加えてトラッキングされたコントローラーの座標を小さな物体を配置することで表示したところ，図 2.7 の赤点の位置に配置された。この位置をコントローラーの位置として見なすことにする．

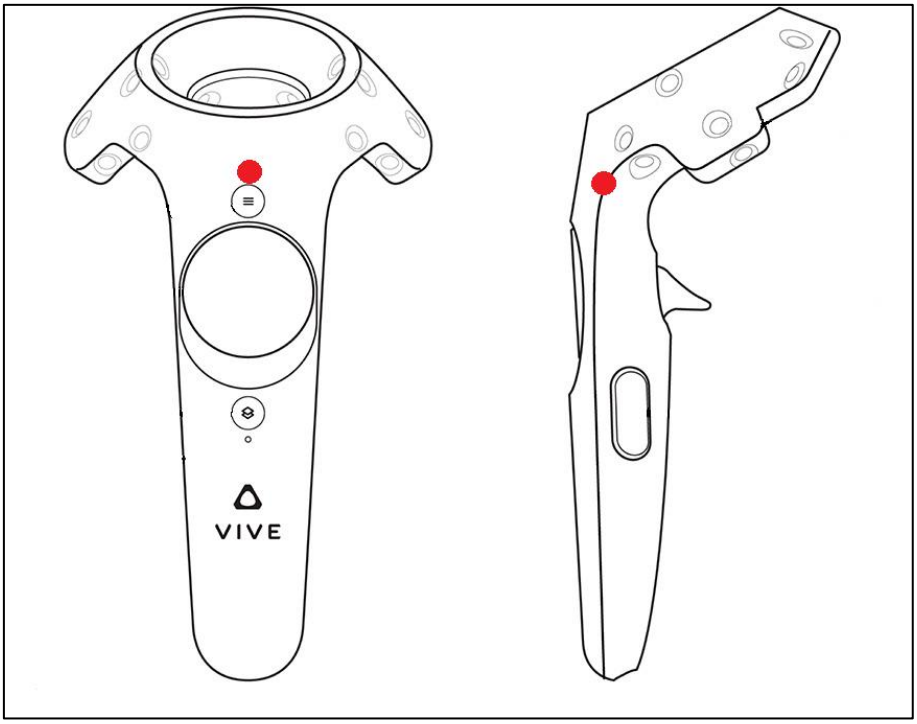


図 2.7　コントローラーの計測点

第 3 章　現実世界のカメラの位置と姿勢の推定

現実世界のカメラの位置と姿勢の推定には，最小二乗法を用いて 3 次元空間上に直線と平面を推定するが，後に紹介する 3.2.1 項の推定方法（最小二乗法で推定した直線を用いた推定方法）以外では直線および平面の位置は推定に使用せず，直線や平面のベクトルを使用できれば良いためコントローラーのどの部分をトラッキングした位置座標としているかはあまり重要ではない．また，最小二乗法の簡単な説明は付録に記載する．

3.1　現実世界のカメラの姿勢の推定

仮想空間のカメラと現実のカメラで姿勢を一致させるために，現実世界のカメラの姿勢を推定し，それに合わせて仮想空間のカメラの姿勢を変更することとした．現実のカメラの姿勢を表すベクトルを下(図 3.1)のように定義する．

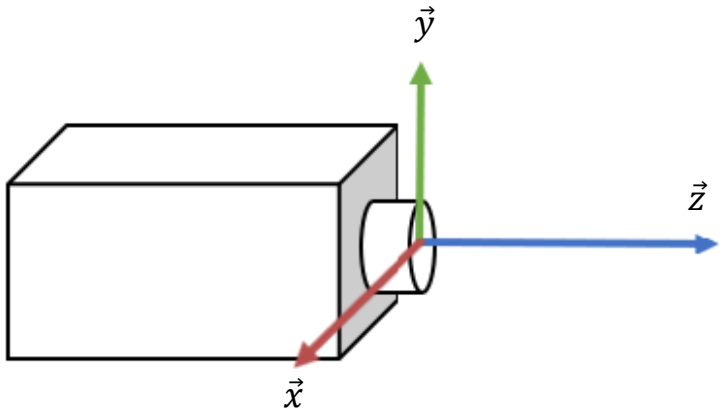


図 3.1　現実世界のカメラのベクトル

図 3.2 における中心の直線のベクトルは$\vec{z}$と一致する．図 3.2 のような 4 点×5 組を理想的な計測をした際には，すべての点が同一平面上に存在しているはずであり，この平面は現実のカメラにおける$\vec{z}$と$\vec{x}$が作る平面と一致する．また，その法線ベクトルは$\vec{y}$と一致する．つまりこれらの平面，ベクトルは現実のカメラの姿勢を表すものであり，これを用いて仮想世界のカメラの姿勢を変更すれば良い．

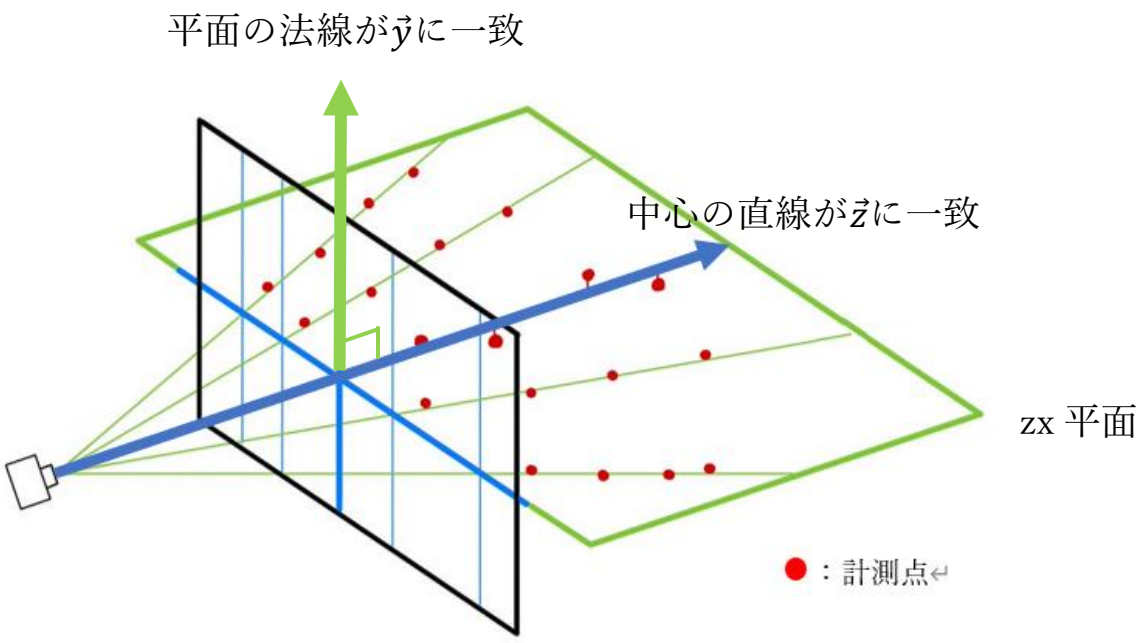


図 3.2　求まる平面，ベクトル

3.1.1　カメラの姿勢の推定方法

カメラの姿勢は 3 つの直交するベクトルで表現できる．3 つのベクトルの内 2 つを求めることができれば残り 1 つは 2 つのベクトルの外積で表すことができる．よって現実世界のカメラの 3 つのベクトルの内 2 つを推定することができればカメラの姿勢も推定できるはずである．上記 3 つのベクトルについて，現実世界のカメラのベクトル(図 3.1)と同様に定義する．

カメラ姿勢の推定は以下のような手順で行った．

1. 図 3.3 の青い縦線と横線の交点でコントローラーの位置を 4 点ずつ計測する
2. 中心の縦線でのみ最小二乗法で直線を推定し，その方向ベクトルをカメラの$\vec{z}$とする
3. 4 点×5 組の計 20 点で適切な平面を推定する
4. 手順 3 で推定した平面の法線ベクトルをカメラの$\vec{y}$とする
5. 手順 2，4 で推定したベクトルの外積をカメラの$\vec{x}$とする

手順の 1 についてはカメラ位置の推定の時と同じ手順をとっているが，図 3.3 の横線にコントローラーの高さを合わせれば，中心の縦線を除き正確に縦線メモリにコントローラーを合わせて計測する必要はない．

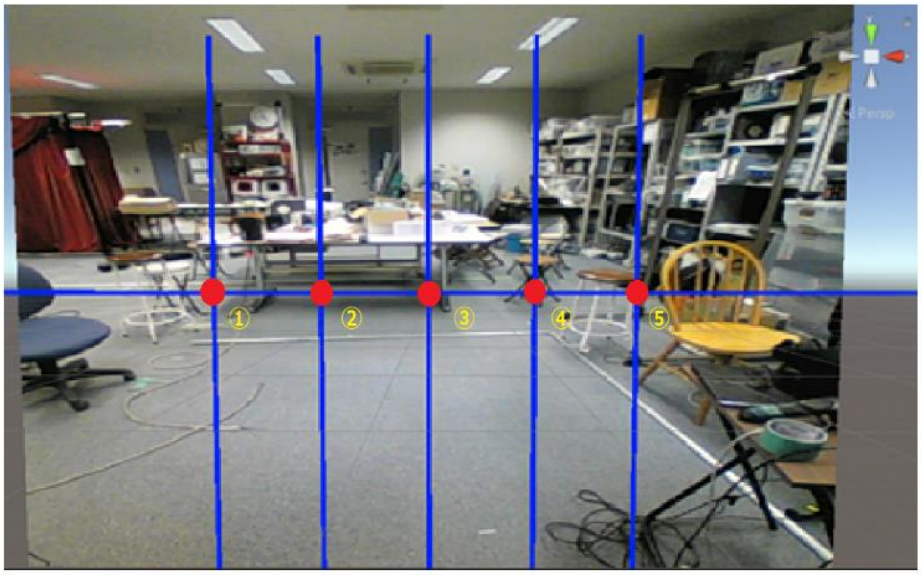


図 3.3　映像に描いた縦線と横線

2. 中心の縦線でのみ最小二乗法で直線を推定する
   中心の縦線は画面全体の中心なので，中心の 4 点で推定される直線はカメラの 3 つのベクトルの内の$\vec{z}$となる．

3. 計 20 点で適切な平面を推定する
   推定には最小二乗平面を用いており，その計算式は付録に記す．

4. 手順 2 で推定した平面の法線ベクトルをカメラの$\vec{y}$とする

5. 手順 2，4 で推定したベクトルの外積$(\vec{z} \times \vec{y})$をカメラの$\vec{x}$とする

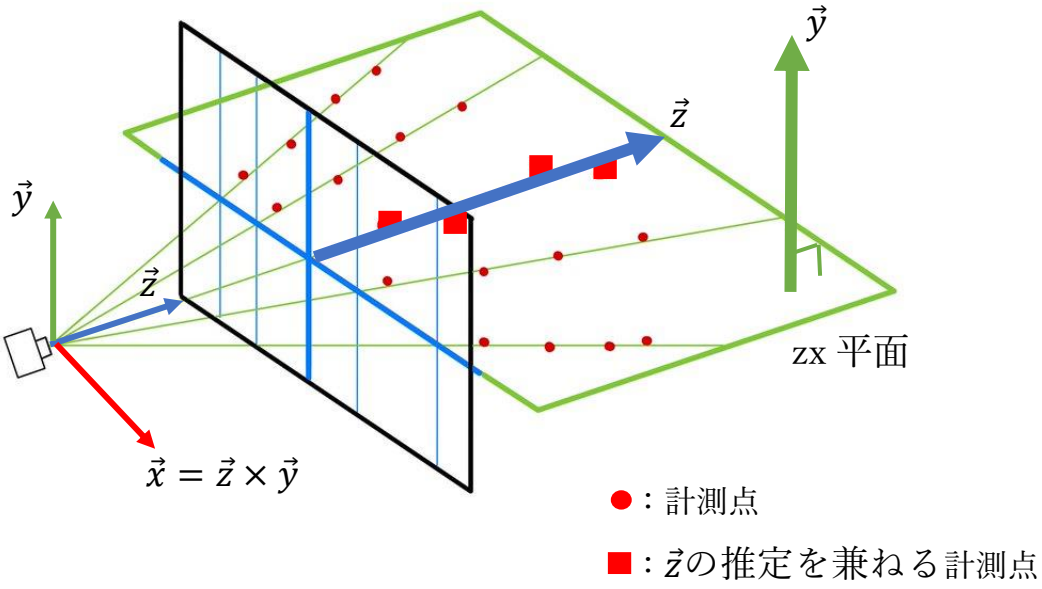


図 3.4　カメラの姿勢推定

この推定方法で仮想世界のCanvas にレンダリングされた現実世界のコントローラーと仮想世界のコントローラーのモデルとの誤差が少ないため，カメラの姿勢は正しく推定できたといえる(図 3.7).

図 3.5　現実コントローラー(青)と仮想コントローラー(黄)の誤差

3.1.2　最小二乗平面

上記の通り，本取り組みではカメラの姿勢の推定の過程で平面を求める．推定方法として最小二乗法を用いることから求める平面を最小二乗平面と呼称する．　この平面は実際の観測データとの高さの差の二乗の和が最小になるような平面である．最小二乗平面 $z = a + bx + cy$（xyを水平面，zを高さとする）の係数a，b，cは以下のようになるが，それを求める解法については付録に記載しておく．

$$a = \frac{1}{\sum 1}\left\{\sum z_i - \left(\sum x_i\right) b - \left(\sum y_i\right) c\right\}$$

$$b = \frac{\sum x_i z_i \sum z_i - \sum 1 \sum x_i^2 - (\sum x_i)^2 \dfrac{\sum 1 \left\{\left(1 - \dfrac{1}{\sum 1}\right) \sum x_i \sum y_i\right\}^2}{\sum 1 \sum x_i^2 - (\sum x_i)^2} c}{\sum 1 \sum x_i^2 - (\sum x_i)^2}$$

$$c = \frac{\sum y_i z_i - 2 \sum x_i z_i + \sum z_i}{\left[\left\{1 - \dfrac{1}{\sum 1} \sum x_i \sum y_i - \sum x_i^2 + \dfrac{(\sum x_i)^2}{\sum 1}\right\} (\sum x_i - \sum y_i)\right]} + \left\langle \left[\sum x_i \sum y_i^2 - \frac{\sum x_i (\sum y_i)^2}{\sum 1} - \frac{\sum x_i \sum 1 \left\{\left(1 - \dfrac{1}{\sum 1}\right) \sum x_i \sum y_i\right\}^2}{\sum 1 \sum x_i^2 - (\sum x_i)^2}\right] \right\rangle \{\sum 1 \sum x_i^2 - (\sum x_i)^2\}$$

3.2　現実世界のカメラの位置の推定

まず，カメラとある物体を直線で結んだ際に，その直線上で物体を移動させることを考える．　その時，図 3.8 のようにカメラで撮影される映像では物体の位置は変化せず，大きさのみが変化するはずである．　このことから，カメラで撮影される映像上で位置が変化しないように物体を動かすことができれば，その移動の延長線上にカメラが存在するはずである．　また，異なる位置で同様の移動を行うと，図 3.9 のように二つの延長線の交点にカメラは存在するはずである．このことを用いて現実のカメラの位置を推定する．

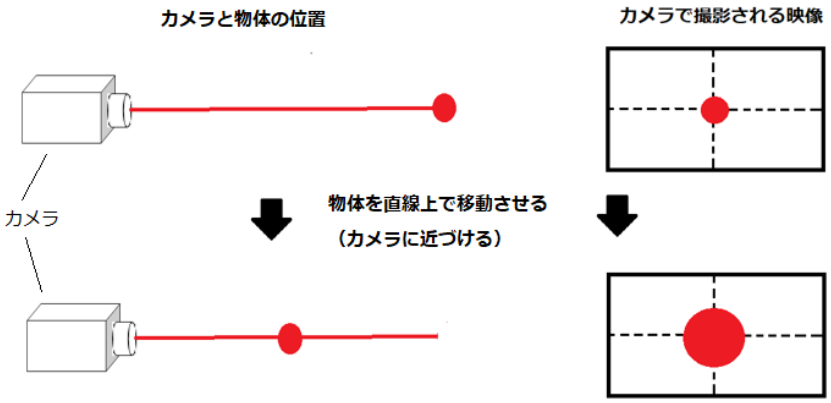


図 3.6　カメラと物体を横から見た図とカメラの映像

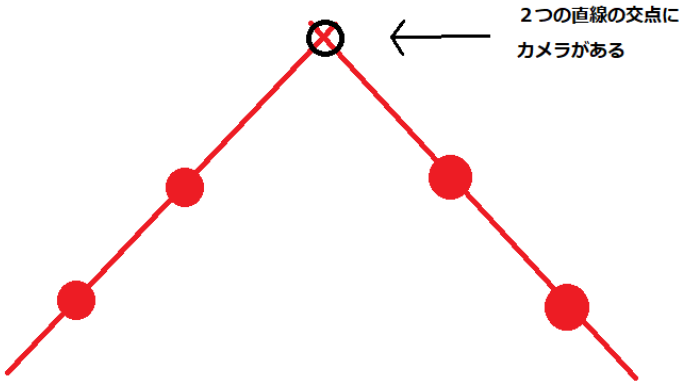


図 3.7　2 直線の交点を上から見た図

3.2.1　最適化された直線によって位置を推定する方法

筆者らは移動させる物体として HTC VIVE コントローラーを用いた．移動経路上の座標を HTC VIVE のトラッキング機能によって計測する事で，仮想世界内で直線を定義できる．この直線から先述の通りカメラの位置を推定することが可能となる．

ただし計測には誤差が生じることから求めたい直線上と異なった位置に計測点が取られることを考慮し，各直線に対して 4 点を計測，それらから最小二乗法を用いて最適な直線を推定することとする．　また，2 直線が交わらずねじれの位置関係になる可能性を考え，その 2 直線が最も近づく位置がカメラの位置として適当な位置とする．

最後にこれらの試行は誤差の影響を小さくするため，5 つの直線で行うことする．

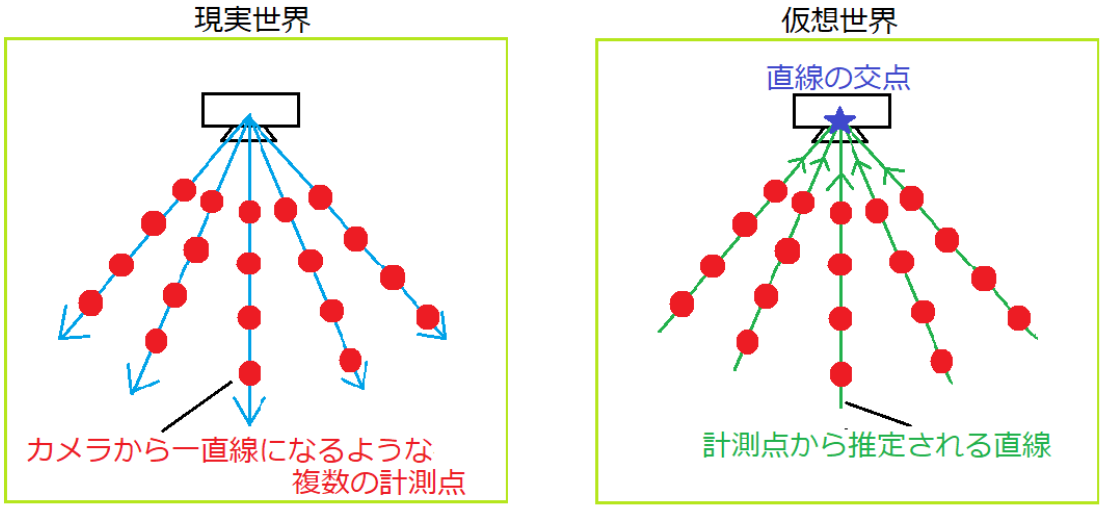


図 3.8　最小二乗法で推定した直線を用いた推定方法の図

3.2.1.1　推定の手順

以下のような手順で推定を行う.

1. 図 3.3 の青色の縦線と横線の交点でコントローラーの位置を計測する
2. 図 3.3 の①～⑤それぞれで計測した 4 点を用いて最小二乗法で直線を推定する
3. 隣り合う 2 本の直線の組み合わせで 2 本の線が最も近づく点を求める.
4. 求まった 4 点の平均の座標値をカメラの位置とする

以下にそれぞれの手順についての詳細を説明する.

1. 図 3.3 の青色の縦線と横線の交点でコントローラーの位置を計測する

図 3.3 の赤い丸の位置にコントローラーを合わせてボタンを押すことで，仮想世界内で点を設置する.

図 3.3 の③の位置で 4 点計測した後，そのほかの場所でも 4 点ずつ計測する(計 20 点).

目盛りの横の線は現実の映像を映す Canvas の縦方向の大きさの中心，③の計測点の直線は Canvas の横方向の大きさの中心になるように設置されている．

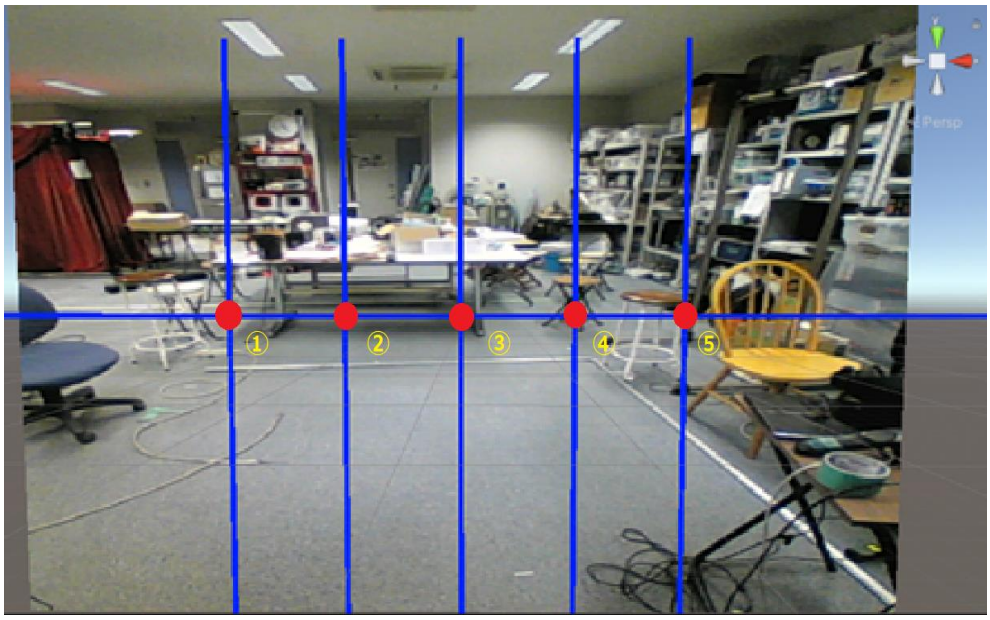


図 3.3　映像の目盛り

図 3.11 が 20 点取り終えたあとの状態である．カメラから一直線上になるような 4 点が 5 組存在している．

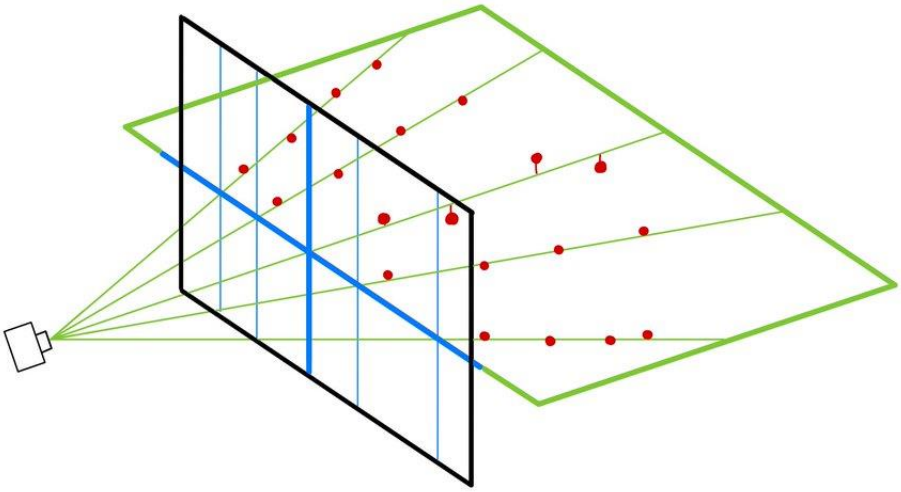


図 3.9　20 点計測後の様子

2.　図 3.3 の①～⑤それぞれのメモリ上の 4 点を用いて最小二乗法で直線を推定する．最小二乗法を用いた直線の推定方法についてはレポートの末尾に記載する．
図 3.10 のような 5 つの直線が推定される．

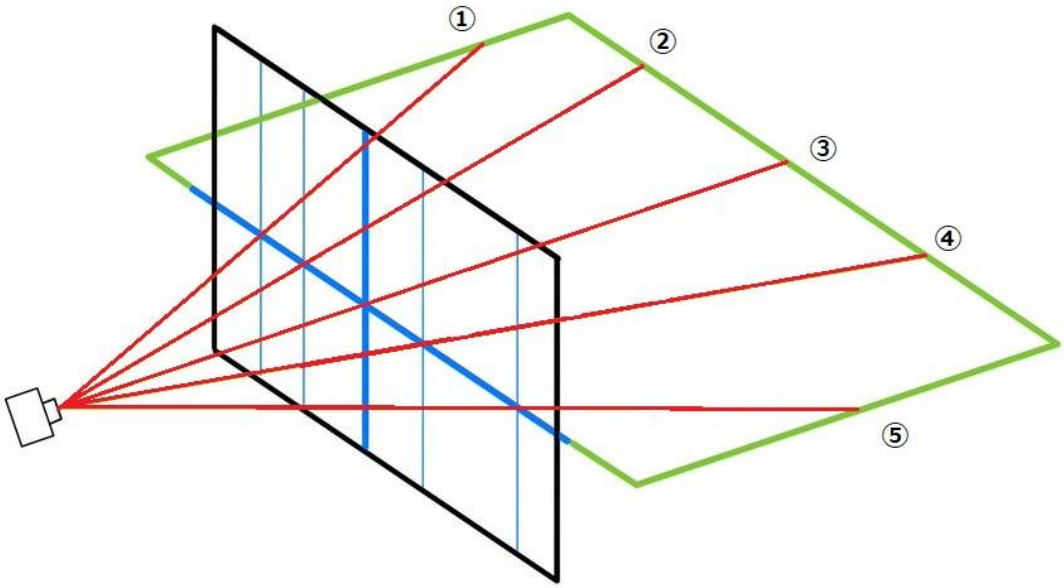


図 3.10　最小二乗法で推定した 5 つの直線

3. 図 3.10 の隣り合う 2 本の直線の組み合わせ(①, ②)(②, ③), (③, ④), (④, ⑤)で 2 本の線が最も近づく座標を求める.

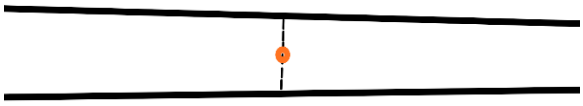


図 3.11A　ねじれの位置の 2 直線が最も近づく点を横から見た図

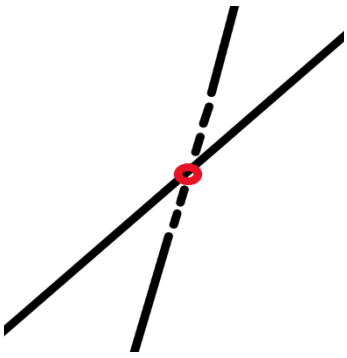


図 3.11B　ねじれの位置の 2 直線が最も近づく点を上から見た図

4.　求まった 4 点の平均の座標をカメラの位置とする

3.2.1.2　直線の推定

3.2.1 で述べた通り，直線の推定には最小二乗法を用いる．　この最適化では三次元空間上の直線を，(1)xy 平面，(2)yz 平面の 2 つの平面に射影し，これらの直線の方程式の係数部分を最小二乗法によって求めることで 3 次元空間上の直線の方程式を求めることができる．

求める直線を

$$\vec{P} = \overrightarrow{P_0} + t\vec{l}$$

と表す．

ただし，$\overrightarrow{P_0} = \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix}, \vec{l} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$ $(a^2 + b^2 + c^2 = 1$かつ$a \neq 0,\quad b \neq 0,$

$c \neq 0)$

とする．

(1)，(2)の平面に射影された直線の方程式を以下のように表す．

$$y = Ax + B, \quad (1)$$

$$y = Cz + D, \quad (2)$$

これらの式の係数部分を求めることによって求めたい直線上の任意

の点$P_0\left(x_0,\ y_0,\ z_0\right)$と直線のベクトル$\vec{l}=\begin{bmatrix}a\\b\\c\end{bmatrix}$を以下のように求めることができるがその解法は付録に記載する.

$$\overrightarrow{P_0}=\begin{pmatrix}B\\0\\D\end{pmatrix},\ \vec{l}=\begin{pmatrix}A\\1\\C\end{pmatrix}$$

$$A=\frac{-B\sum x_i+\sum y_i x_i}{\sum x_i^2},\qquad B=\frac{\frac{\sum y_i x_i\sum x_i}{\sum x_i^2}-\sum y_i}{n-\frac{(\sum x_i)^2}{\sum x_i^2}}$$

$$C=\frac{-D\sum z_i+\sum y_i z_i}{\sum z_i^2},\qquad D=\frac{\frac{\sum y_i z_i\sum z_i}{\sum z_i^2}-\sum y_i}{n-\frac{(\sum z_i)^2}{\sum z_i^2}}$$

3.2.1.3　推定した位置のずれ

この推定方法では，計測点を計測する人によってカメラの位置にずれが生じることとなった．そこで筆者らはこのカメラのずれを画面上のコントローラーのずれを用いて計測することとした．現実世界のカメラで撮影でき，カメラから前方 1.5[m]ほどの位置に 3 点の計測地点を設定して，その点にコントローラーを置き，現実世界のコントローラーと仮想世界のコントローラーの位置のずれを用いて，カメラの 3 次元座標（x，y，z）のずれの大きさを計測した．x軸方向にカメラの位置がずれていた場合，最終的に作成される目標となる映像上ではコントローラーは横方向にずれているはずである．y軸方向にずれている場合は同様に考えると縦方向にずれている．　また，画面端の計測点ほどずれが大きくなるが，z方向にずれていた場合，計測点の位置によるx軸及びy軸方向のずれの大きさがカメラの位置のずれによって変化している．この変化は一章の透視投影変換で述べた，カメラから近い物体ほど移動幅が大きく，カメラから遠い物体ほど移動幅が小さく表示される特性によって引き起こされると考える．例えば，現実世界上のカメラが実在のコントローラーを撮影する距離より，仮想世界上のカメラが仮想世界のコントローラーを撮影する距離が近く設置された場合，実在のコントローラーよりも仮想世界のコントローラーは映像の中心から遠ざかるように表示される．

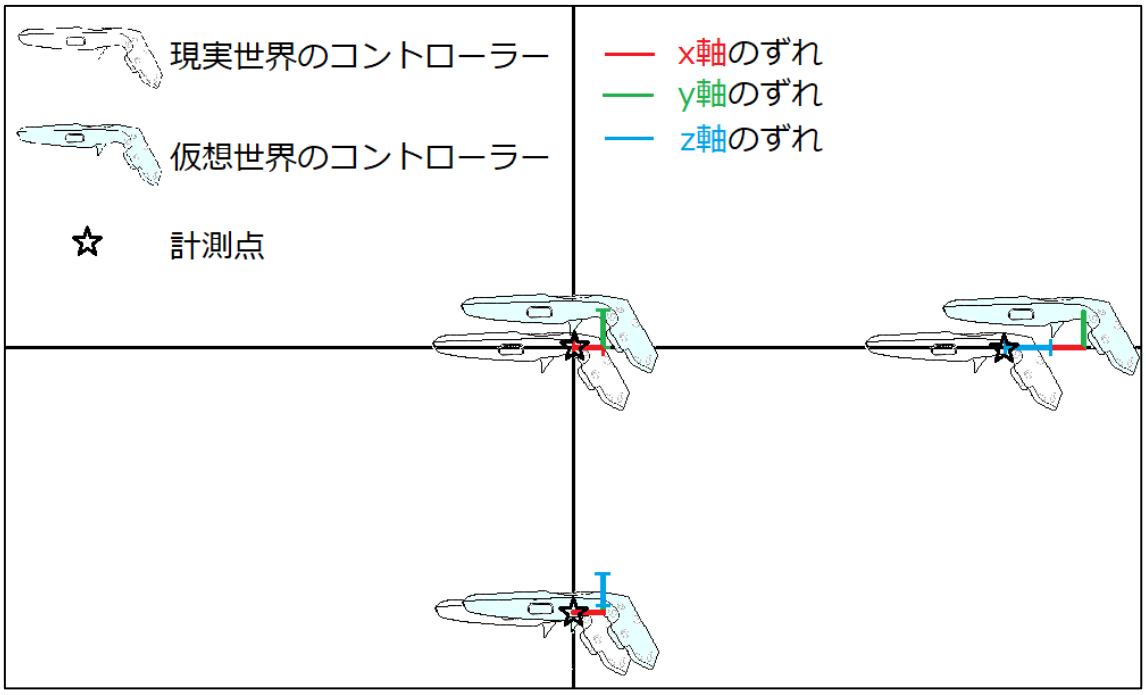


図 3.12　生じるずれ

この推定方法で生じた位置のずれをパソコンの画面上(13.3[inch]，比率 16 : 9，　横幅 29.4[cm]，　縦幅 16.6[cm])でスクリーンショットを表示して計測した．z軸方向のずれについては画面中央の計測点と画面端の計測点とのずれの差を計測することとした．y軸方向には大きく差が現れないことから，x軸方向のずれの差を計測している．8 回の施行を行ったところ，以下のような結果となった．またこの位置のずれには 3.1 で述べた姿勢のずれも含まれている．

表 3.1：3.2.1 の方法で生じたカメラ位置のずれ

ずれ	X 軸方向	Y 軸方向	Z 軸方向
平均[mm]	2	2.333333	4
分散	8.857143	6.839286	9.410714

この原因として，計測点がメモリから外れている方向に偏りがある，計測点が多いためにミスが起きやすいなどの人的要因が大部分を占めていると考えられる．計測する人によって生じるずれのむらを極力減らすために筆者らは 3.2.2 の手法を用いてカメラの位置を推定することとした．

3.2.2　直接カメラの位置を測定する方法

3.2.2.1　測定の手順

この推定方法は以下のような手順で行った.

1. カメラの正面にコントローラーをおき，カメラに接触するようにしてコントローラーの位置を取得する

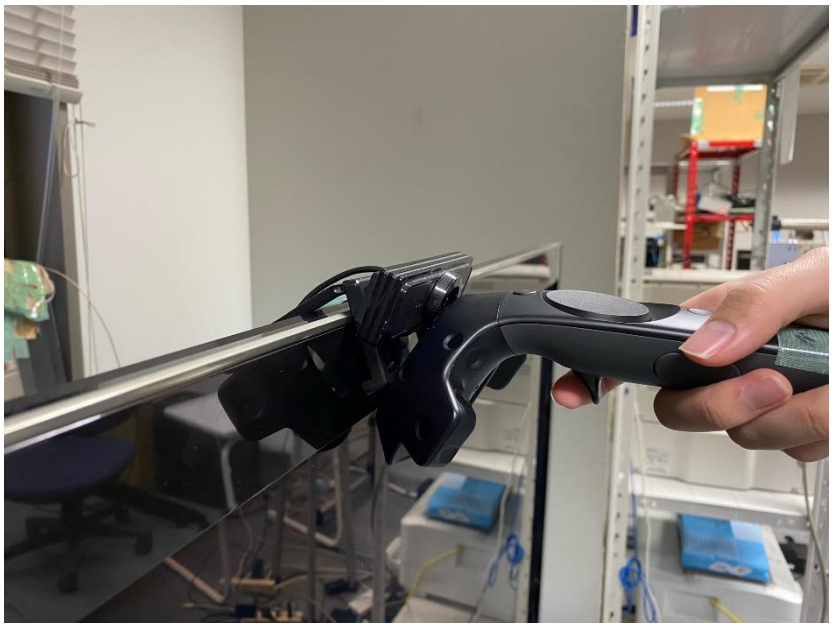


図 3.13　推定方法 3.2.2 におけるカメラの位置の測定の仕方

2.　取得した位置座標を仮想世界カメラの座標にする

3.2.2.2　推定した位置のずれ

3.2.2 の推定方法では，3.2.1 の推定方法よりも容易でありながら計測者に由来するカメラの位置のずれは小さくなった．

この推定方法で生じた位置のずれを 3.2.2.3 と同じように 8 回計測した．結果は以下のようになった．またこの位置のずれには 3.1 で述べた姿勢のずれも含まれている．

表 3.2：3.2.2 の方法で生じた位置のずれ

ずれ	X 軸方向	Y 軸方向	Z 軸方向
平均[mm]	1.625	1.5	5.875
分散	1.982143	1.714286	0.410714

3.2.1 の推定方法で生じた位置のずれと 3.2.2 の推定方法で生じた位置のずれについて，ずれの大きさの平均は z 方向以外のずれは 3.2.1 の方が大きく，分散についても 3.2.1 の方が大きくなっている．ここから，3.2.2 の推定方法の方がカメラの位置をより正しく推定できていたといえる．また，分散が減少したことから計測者による推定されるカメラの位置の違いも小さくなっていたといえるだろう．

第 4 章　モーションキャプチャ

本取り組みでは，モーションキャプチャをするにあたって Web カメラを用いた，3 次元姿勢推定を行う．　またこの方法については，　はなちるのマイノート[5]を参考にした．　そこでは以下のように説明されていた.

4.1　手段

モーションキャプチャには Unity Technologies 社によって製作された推論ライブラリである Barracuda[6]及び，それにロードさせるための学習モデルとして ResNet[7]を用いたセマンティックセグメンテーションモデルを利用している.

4.2　特徴点

モーションキャプチャによって 3D モデルを動かすためには，関節点や耳，鼻，腰などの特徴点が必要となる．　この特徴点によって，人間がどのような姿勢を取っているかを認識することが可能となる.

4.3 で述べるセマンティックセグメンテーションを用いて人間を認識した後，そこからさらに左右の目，耳，肩，肘，手首，親指の付け根，中指の付け根，股関節，膝，足首，つま先，鼻，おなかの合計 24 点を判別し，それを特徴点として人間の姿勢を推定している．　図 4.1 に特徴点の場所を示す.

この特徴点の位置関係から 3D モデルの体の向きを推定していることが考えられるが，それは参考文献の中では明らかにされていない.

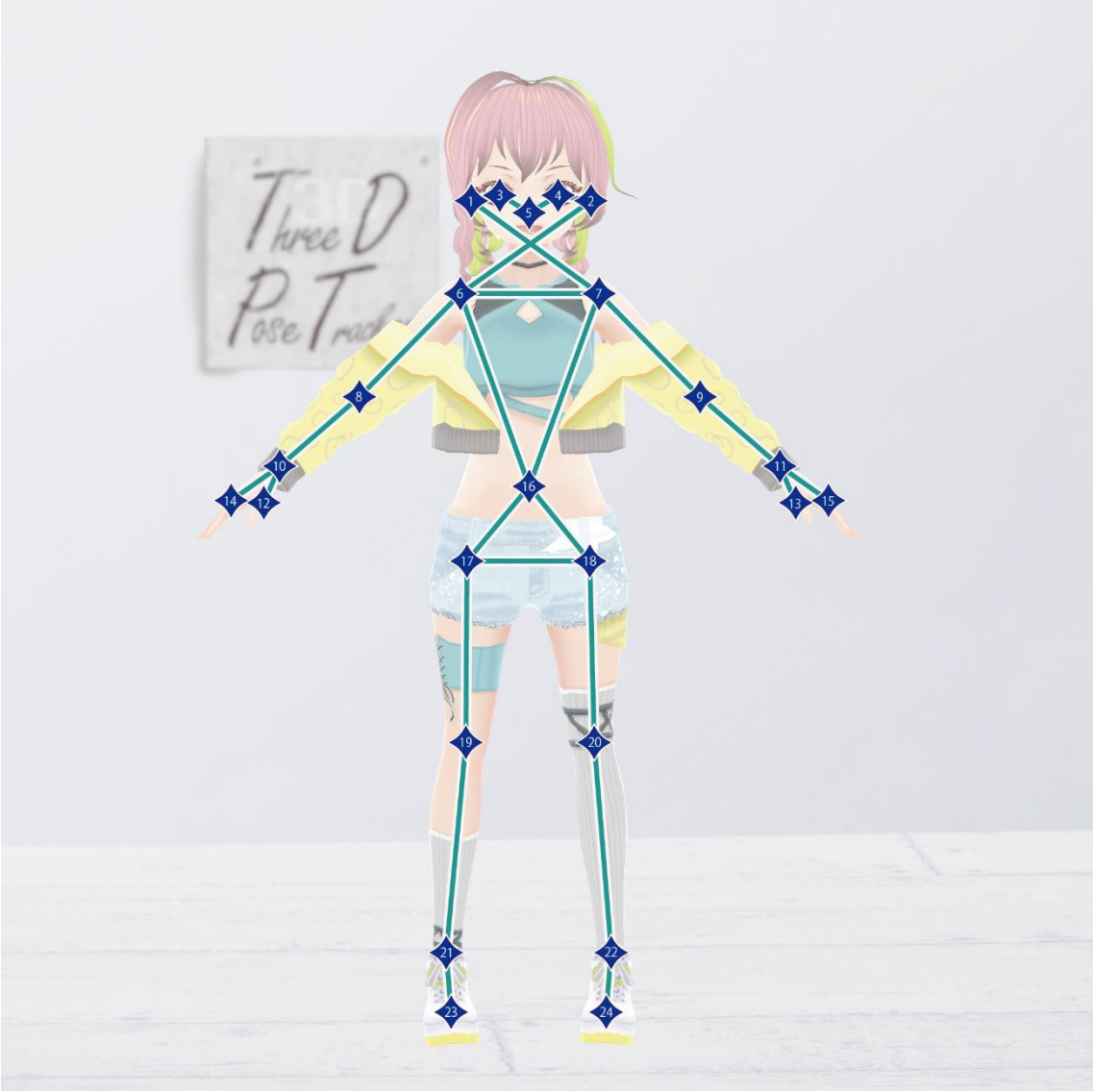


図 4.1　モーションキャプチャの特徴点

4.3　セマンティックセグメンテーション

セマンティックセグメンテーションとは，画像のピクセル 1 つ 1 つに対して何が写っているかを，ラベルやカテゴリで関連付けることである． 例を挙げると，以下の図 4.2 及び，図 4.3 のように人や乗り物，植物，空などでカテゴリ分けをすることで，物の判別を行っている．

図 4.2　元の画像

図 4.3 カテゴリ分けした画像

第 5 章　製作物の纏め

5.1　完成様子

制作物は以下の図のようになった.

図 5.1　正面

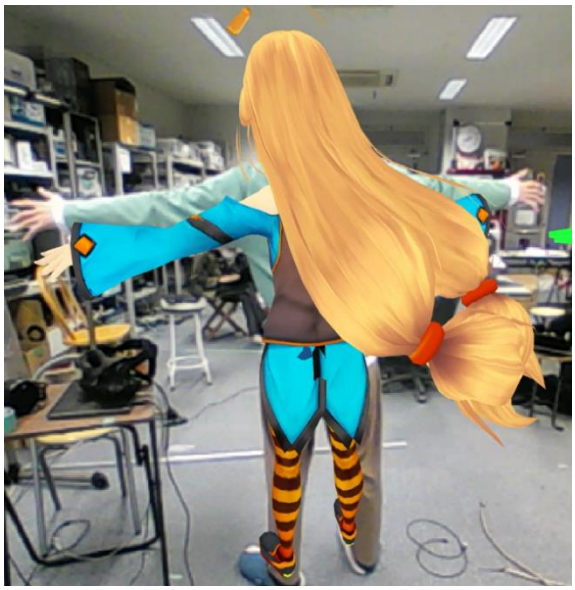


図 5.2　背面

図 5.3　側面(左向き)

図 5.4　側面(右向き)

図 5.5　プレイの様子

図 5.6　コントローラーの装着様子

5.2　仕様

1. プレイヤーは下腹部に HTC VIVE の左コントローラーを装着する.
2. プレイヤーは現実のディスプレイに表示される仮想空間の映像から，手順通りに現実空間のカメラの位置・姿勢の推定を行う.
3. 仮想空間のカメラを推定した現実空間のカメラと一致するように動かす.
4. プレイヤーの体に重なるように表示されたアバターがプレイヤーと同じ動作をする.

6章　纏め・今後の展望

本取り組みでは，以下の 2 つの機能の実装を目標としていた．

1.モーションキャプチャ

2.現実世界と仮想世界のカメラの一致

この 2 つの機能の実装に概ね成功した．ユーザがプレイエリア内にいるという条件の下であるが，現実世界のカメラと仮想世界のカメラの位置，姿勢を合わせ，モーションキャプチャと組み合わせることで，アバターにユーザと同様の動きをさせることに成功した．

しかし，本取り組みで実装した機能では，

1. 同時に一人までしかモーションキャプチャを行うことができない
2. 点を計測する人によってカメラの位置・姿勢が変化する

ことが問題として挙げられる．この問題を解決することによってプレイヤーはより本取り組みの内容について楽しむことができると考えられる． また，本取り組みでは仮想世界の空間の映像を提示していないことからこれらと，機能の問題の解決の必要がある．

その方法として

1. 現実世界でのカメラを増やすことによって複数人のモーションキャプチャを行う
2. 仮想世界上に現実世界と同じ机やいすなどのオブジェクトを用意し，位置合わせを行うことでプレイヤーに仮想世界上で現実のような世界を体験させる

ことが考えられる．

今後の展望として，アバターではなく仮想空間上に表示した服のみを人間に被せるなど，バーチャルな試着を行うことも考えられる．

謝辞

本取り組みを進めるに当たり，御指導，御助言を賜りました岐阜大学工学部電気電子・情報工学科の木島竜吾准教授に対し，心から感謝の意を表する．最後に，日頃お世話になっている木島研究室の先輩の方々に感謝の意を表する．

参考文献

[1]水上孝一．　コンピュータグラフィックス　－情報化社会と映像－．　朝倉書店，　1989，　227p

[2]白井良明．　コンピュータビジョン．　昭晃堂，　1980，　192p

[3] “透視投影変換とは”，　NO MORE! 車輪の再発明，　2018，　透視投影変換とは ｜NO MORE! 車輪の再発明（mem-archive．com）

[4] 林 昌希，　“カメラモデル（Camera Model）と透視投影（Perspective Projection）”，CVML エキスパートガイド，　2019，https://cvml-expertguide．net/terms/cv/camera-geometry/camera-model/

[5] はなちる，　“【Unity】パソコンのカメラからモーションキャプチャをして AnimationClip を作成する”，　はなちるのマイノート，2020，　https://www．hanachiru-blog．com/entry/2020/02/19/120000

[6] 高橋啓治郎，　“Unity Barracuda を使用した ONNX ニューラルネットワークモデルのマルチプラットフォーム運用”，　Unity Learning Materials，　2021，https://learning．unity3d．jp/7557/

[7] Deep Age，　“ResNet の理解とチューニングのベストプラクティス”，　Deep Age，　2016，https://deepage．net/deep_learning/2016/11/30/resnet．html

[8] edo_m18, “[数学] 最小二乗平面をプログラムで求める”, Qiita, 2017, https://qiita.com/edo_m18/items/82659d6b1b122e80f645#:~:text=%E6%9C%80%E5%B0%8F%E4%BA%8C%E4%B9%97%E5%B9%B3%E9%9D%A2%E3%81%AF%E3%80%81%E6%9C%80%E5%B0%8F%E4%BA%8C%E4%B9%97%E6%B3%95%E3%81%8B%E3%82%89%E6%B1%82%E3%82%81%E3%81%9F%24a%2C%20b%2C,c%24%E3%82%92%E4%B8%8A%E8%A8%98%E3%81%AE%E6%96%B9%E7%A8%8B%E5%BC%8F%E3%81%AB%E5%BD%93%E3%81%A6%E3%81%AF%E3%82%81%E3%81%A6%E6%B1%82%E3%82%81%E3%82%89%E3%82%8C%E3%82%8B%E5%B9%B3%E9%9D%A2%E3%81%A8%E3%81%AA%E3%82%8A%E3%81%BE%E3%81%99%E3%80%82%20%E3%81%A4%E3%81%BE%E3%82%8A%E3%80%81%E3%81%A9%E3%81%AE%E7%82%B9%E3%81%8B%E3%82%89%E3%82%82%E6%9C%80%E5%B0%8F%E8%B7%9D%E9%9B%A2%E3%81%AE%E4%BA%8C%E4%B9%97%E3%81%A8%E3%81%AA%E3%82%8B%E5%B9%B3%E9%9D%A2%E3%80%81%E3%81%A8%E3%81%84%E3%81%86%E3%81%93%E3%81%A8%E3%81%A7%E3%81%99%E3%80%82

[9] ” 【プログラミング演習 2 資料】LU 分解による連立方程式の解法”, 宇都宮大学情報工学教育研究用計算機システム, 2023, http://www.ced.is.utsunomiya-u.ac.jp/lecture/2011/prog/p2/kadai3/no3/lu.pdf

[10] toughman, ”3 次元空間の回帰直線の求め方”, 未来数理合同会社, 2016, https://math2future.

com/main/2016/12/22/regression_line_in_3d_space/

図一覧

表一覧

付録

1　1次元方向に対する最小二乗法

本取り組み内において最適な直線を推定するために一次元方向に対する最小二乗法を用いたので紹介する．

最小二乗法は誤差の生じる測定で得られた数値の組（ここでは$\left(x_i,\ y_i\right)$とする）の誤差の二乗の和を最小にすることで最も確からしい関係式を求める方法のことである．

最小二乗法の最も簡単な例として平面での直線の推定をあげると，測定の際に含まれる誤差は以下の式で表される

$\varepsilon = y - f(x)$　　（誤差ε，測定データy，モデル関数$f(x)$）

よって，データの組$\left(x_i,\ y_i\right)$の誤差の二乗の和は

$$\sum_{i=1}^{n}\{y_i - f(x)\}^2$$

と表すことができる．最小二乗法では，この誤差の二乗の和が最小になるような$f(x)$を最も確からしい関係式とする．

2　平面の推定

本取り組み内で最適な平面の推定に用いた計算方法を紹介する[8].

本章 3.1.1 で述べた推定方法ではコントローラーの計測点を元に最適な平面を推定し，その平面をカメラのzx平面としてその法線がy軸となる事を利用することでカメラの姿勢を求めた．ここでは，カメラの姿勢を求めるために必要な最適な平面の方程式を求める．

図 3.3 の①～⑤にコントローラーが重なるように深さを変えコントローラーの位置を計測した．計測点は図 3.4 のようになる．計測点を$p_i = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix}$ $(0 \leq i \leq N-1, \quad N = 20)$とする．

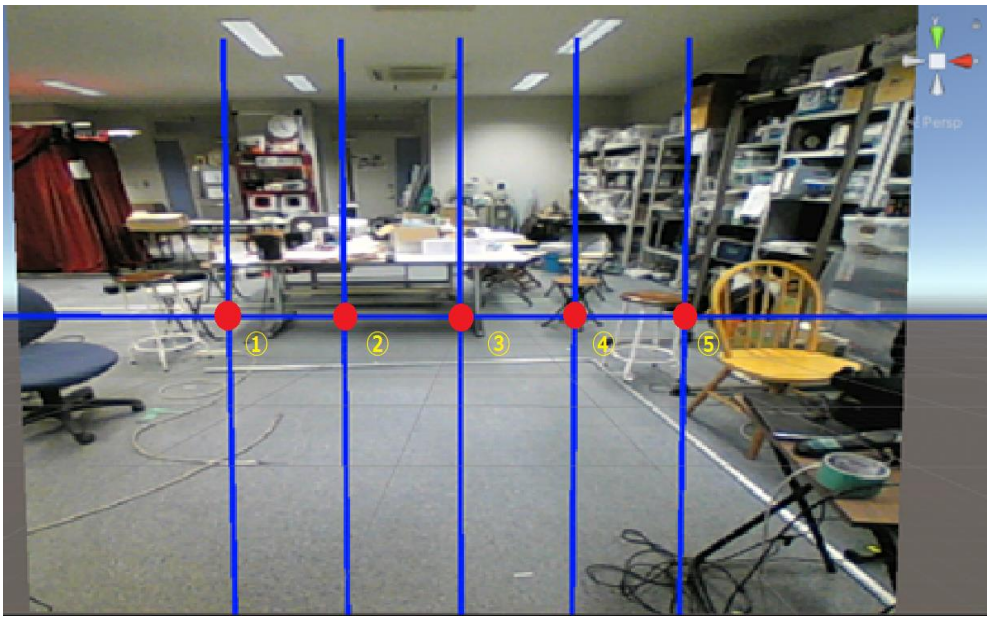


図 3.3　映像に描いた縦線と横線

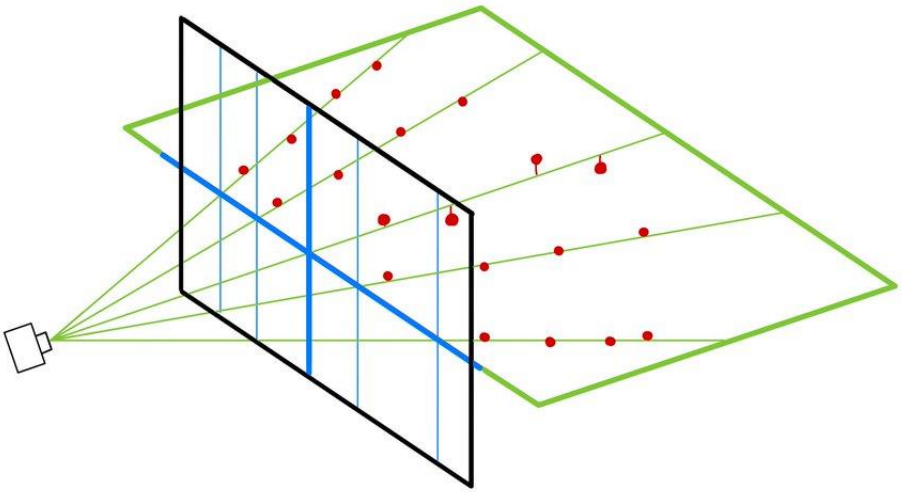


図 3.4　カメラの姿勢推定

求めたい最小二乗平面(xz平面)の方程式を

$$z = a + bx + cy$$（xyを水平面，zを高さとする）

と表すこととする．測定データは（x_i，y_i，z_i）なので，求めたい平面と測定データとの測定誤差εは

$$\varepsilon_i = (a + bx_i + cy_i) - z_i$$

となる．よって，εの二乗の和をφとすると

$$\varphi = \sum {\varepsilon_i}^2 = \sum (a + bx_i + cy_i - z_i)^2$$

となる．このφを評価関数としてφが最小になるようなa，b，cを求める．φをa，b，cそれぞれについて偏微分した値が 0 となるとき，φは極小値をとるが，　φは 2 次方程式なためφは最小値をとる．

a，b，cそれぞれの偏微分は以下のようになる．

$$\frac{\partial F}{\partial a} = 2\sum(a + bx_i + cy_i - z_i) = 0$$
$$\frac{\partial F}{\partial b} = 2\sum(a + bx_i + cy_i - z_i)x_i = 0$$
$$\frac{\partial F}{\partial c} = 2\sum(a + bx_i + cy_i - z_i)y_i = 0$$

これらの式を連立方程式として解きa，b，cを求める．

この連立方程式を解く際に，筆者らは LU 分解という手法を用いた．LU 分解とは正方行列を上三角行列と下三角行列の積に分解することである．上記の連立方程式を

$$A = \begin{bmatrix} \sum 1 & \sum x_i & \sum y_i \\ \sum x_i & \sum x_i^2 & \sum x_i y_i \\ \sum y_i & \sum x_i y_i & \sum y_i^2 \end{bmatrix},$$

$$\vec{x} = \begin{bmatrix} a \\ b \\ c \end{bmatrix},$$

$$\vec{b} = \begin{bmatrix} \sum z_i \\ \sum x_i z_i \\ \sum y_i z_i \end{bmatrix}$$

とすると，連立方程式は以下のように表すことができる．

$$A\vec{x} = \vec{b}$$

ここで，行列 A を LU 分解すると

$$LU\vec{x} = \vec{b}$$

と変形できる．$U\vec{x} = \vec{y}$とすると，

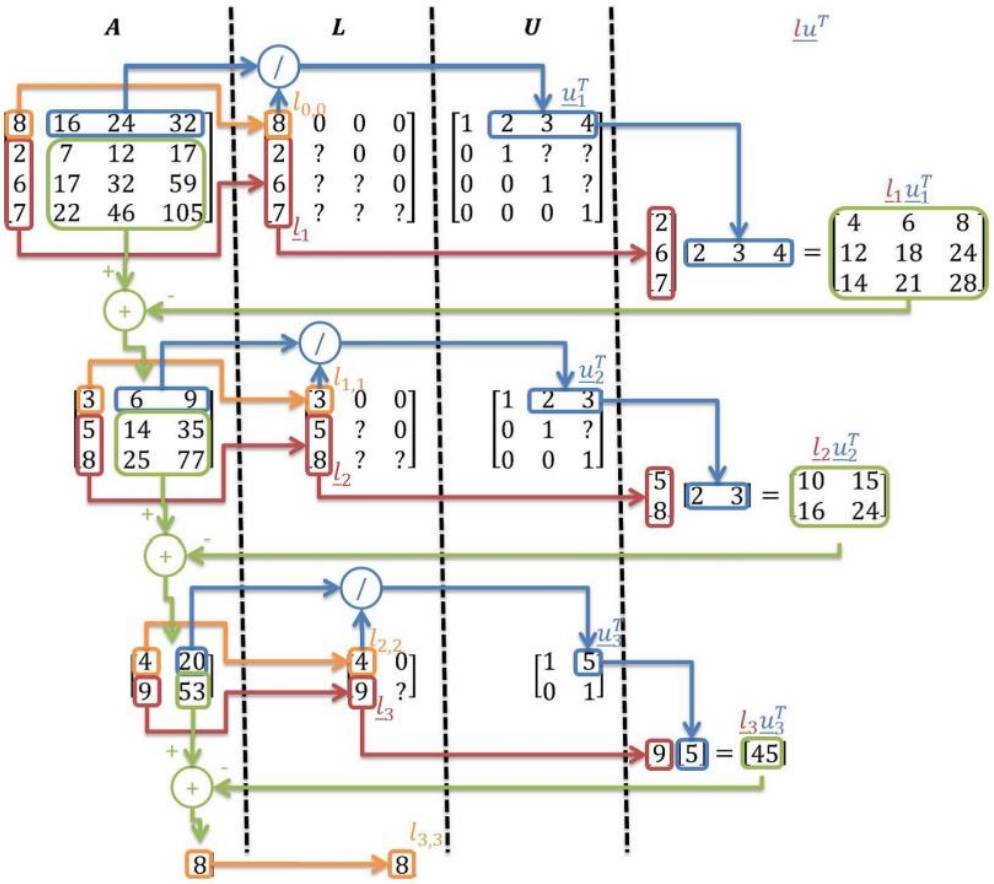
$$L\vec{y} = \vec{b}$$

と表すことができる．後は 2 元連立方程式

$$U\vec{x} = \vec{y}$$
$$L\vec{y} = \vec{b}$$

を解くことで最小二乗平面を求めることができる．

LU 分解は，詳細は省くが以下の画像のように再帰的に行う．



（参考文献[9]より引用）

2 元連立方程式を解き，$\vec{x} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$を求めると，

$$a = \frac{1}{\sum 1}\left\{\sum z_i - \left(\sum x_i\right) b - \left(\sum y_i\right) c\right\}$$

$$b = \frac{\sum x_i z_i \sum z_i - \sum 1 \sum x_i^2 - (\sum x_i)^2 \dfrac{\sum 1 \left\{\left(1 - \dfrac{1}{\sum 1}\right) \sum x_i \sum y_i\right\}^2}{\sum 1 \sum x_i^2 - (\sum x_i)^2} c}{\sum 1 \sum x_i^2 - (\sum x_i)^2}$$

$$c$$

$$= \frac{\sum y_i z_i - 2\sum x_i z_i + \sum z_i}{\left[\left\{1 - \frac{1}{\sum 1}\sum x_i \sum y_i - \sum x_i^2 + \frac{(\sum x_i)^2}{\sum 1}\right\}(\sum x_i - \sum y_i)\right] + \left\langle\left[\sum x_i \sum y_i^2 - \frac{\sum x_i (\sum y_i)^2}{\sum 1} - \frac{\sum x_i \sum 1\left\{\left(1 - \frac{1}{\sum 1}\right)\sum x_i \sum y_i\right\}^2}{\sum 1 \sum x_i^2 - (\sum x_i)^2}\right]\right\rangle \{\sum 1 \sum x_i^2 - (\sum x_i)^2\}}$$

となる.

3　最適化された直線の推定

本章 3.2.1 で述べた推定方法では図 3.10 のように 5 つの直線を推定して，それらの交点がカメラの位置であるとしてカメラの位置を推定した．

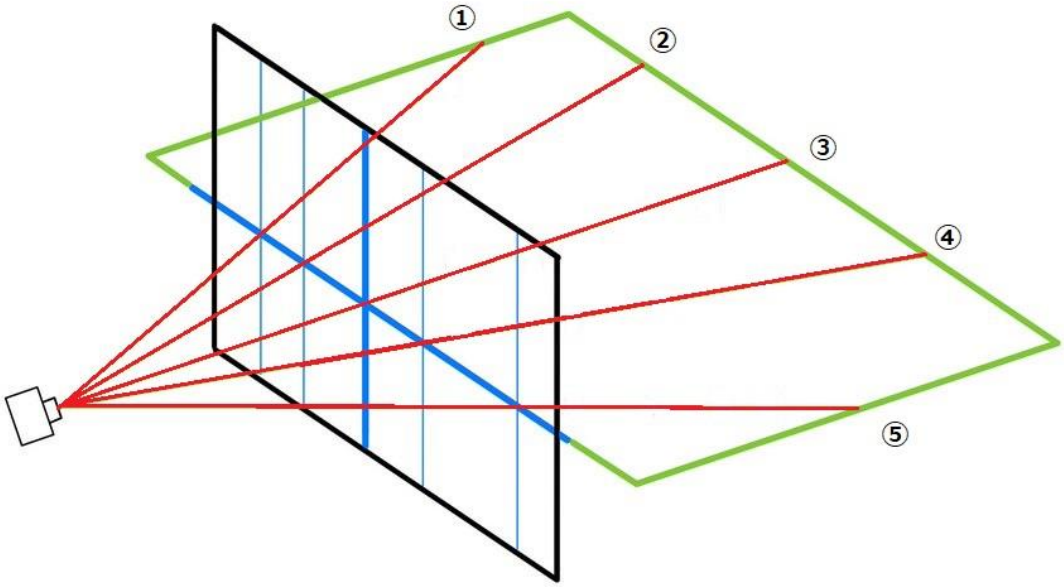


図 3.4　最小二乗法で推定した 5 つの直線

ここでは最小二乗法を用いて 3 次元空間上に 1 本の直線を推定する．この推定は[10]を参考にした．

求める直線を

$$\vec{P} = \overrightarrow{P_0} + t\vec{l}$$

と表す．

ただし，$\overrightarrow{P_0} = \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix}, \vec{l} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$ $(a^2 + b^2 + c^2 = 1$かつ$a \neq 0, \quad b \neq 0,$

$c \neq 0)$

とする．

3 次元空間上の直線を(1)はyx平面，(2)はzy平面上にそれぞれ正射影した直線の方程式を

$$y = Ax + B, \quad (1)$$

$$y = Cz + D, \quad (2)$$

と定義する．これらの直線の方程式の係数部分を最小二乗法によって求めることで 3 次元空間上の直線の方程式を求めることができるはずである．これらの直線が最小二乗法によって最適化された結果として求まる 3 次元空間上の直線が最適化されているかは断言できないが，それらしい 1 本の直線を求めることができると筆者らは考える．

まず(1)について解く．(1)の式を付録 1 の最小二乗法の式に代入したものを M とすると，

$$M = \sum_{i=1}^{n} (y_i - (Ax_i + B))^2$$

右辺を展開して，

$$M = \sum y_i^2 + A^2 \sum x_i^2 + nB^2 - 2A \sum y_i x_i + 2AB \sum x_i - 2B \sum y_i$$

M が最小の値をとるのは，A，B それぞれについて偏微分がともに 0 をとる時である．よって求めるのは，

$$\frac{\partial M}{\partial A} = 2A \sum x_i^2 - 2 \sum y_i x_i + 2B \sum x_i = 0$$

$$\frac{\partial M}{\partial B} = 2nB + 2A \sum x_i - 2 \sum y_i = 0$$

この連立方程式の解である．これを A，B について整理すると，

$$B = \frac{\frac{\sum y_i x_i \sum x_i}{\sum x_i^2} - \sum y_i}{n - \frac{(\sum x_i)^2}{\sum x_i^2}}$$

$$A = \frac{-B\sum x_i + \sum y_i x_i}{\sum x_i^2}$$

となる.

(2)についても同様に最小二乗法によって係数を求めると,

$$D = \frac{\frac{\sum y_i z_i \sum z_i}{\sum z_i^2} - \sum y_i}{n - \frac{(\sum z_i)^2}{\sum z_i^2}}$$

$$C = \frac{-D\sum z_i + \sum y_i z_i}{\sum z_i^2}$$

ここで，(1)，(2)の式が表す直線の傾きは射影する前の直線の傾きの y 成分を 1 としたときの x 成分と z 成分にあたる．よって求める直線の傾きは(A, 1, C)となる.

3 次元空間上で，求めたい直線と xy 平面の交点をP_0とすると，P_0は$(x_0,\ 0,\ z_0)$になるので，$P_0 = (B,\ 0,\ D)$とすることができる.

従って,求める直線の傾き$\vec{l}$と任意の直線上の点$\overrightarrow{P_0}$は

$$\overrightarrow{P_0} = \begin{pmatrix} B \\ 0 \\ D \end{pmatrix},\ \vec{l} = \begin{pmatrix} A \\ 1 \\ C \end{pmatrix}$$

$$A = \frac{-B\sum x_i + \sum y_i x_i}{\sum x_i^2}, \qquad B = \frac{\frac{\sum y_i x_i \sum x_i}{\sum x_i^2} - \sum y_i}{n - \frac{(\sum x_i)^2}{\sum x_i^2}}$$

$$C = \frac{-D\sum z_i + \sum y_i z_i}{\sum z_i^2}, \qquad D = \frac{\frac{\sum y_i z_i \sum z_i}{\sum z_i^2} - \sum y_i}{n - \frac{(\sum z_i)^2}{\sum z_i^2}}$$