Sketching Reality論文の解説。

要するに、各エッジをX, Y, Z軸のいずれかに並行な線であると見なし、3D座標を計算する。しかし、軸に並行であるという制約を直接使うわけではないので、得られた3D座標は軸に並行でなくなっていることが多い（スケッチのエラーなどのために、わずかにずれる）。

どうもまわりくどいという感じだ。エッジがX, Y, Z軸に並行ということは、面がいずれかの軸に垂直ということだ。これを利用したSmartCanvasの方が安定したアルゴリズムだと直感的に思う。なぜなら、得られた3D座標は、確実にいずれかの軸に垂直になっているから。

# 使い方

スケッチを描画し、Tool->Reconstructで3Dモデルに変換される。

**バグなど：**

* Undoは動かない。
* スケッチのロード・セーブもたぶん動かない。
* スケッチが終わって3D生成が終わるまではカメラを回転してはいけない。
* Axis-alignedでないエッジ（黒色のエッジ）があっては駄目。
* 曲がったエッジは駄目。

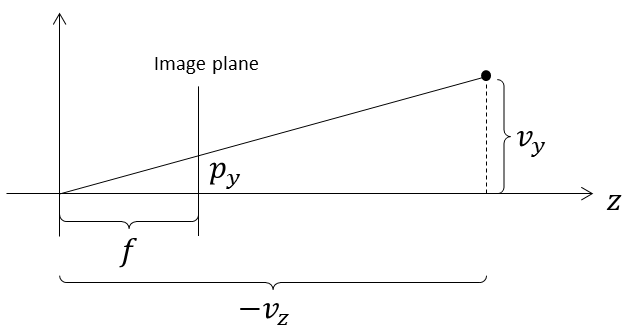
# Reconstruction

## Projection equation

3Dの点が、スクリーン座標（以後、スクリーン座標は、[-1,1]の値とする）をとおくと、例の図を描けば分かるとおり、

(1)

ただし、はアスペクト比だ。このアスペクト比で割ることで、たとえ横長の画面だとしても、X軸座標も[-1,1]の範囲に収めるわけだ。



右手座標系にするため、以後、Z座標の符号を反転する。

式(1)を整理して、

両辺をで割ると、

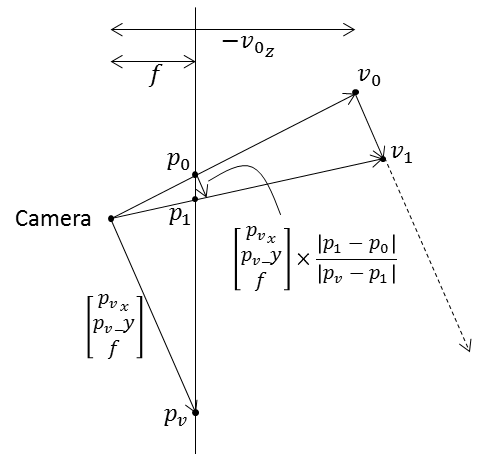
これを行列で表すと、

(2)

ここで、をにしたのは右手座標系にするため。つまり、カメラの前面にあるオブジェクトのZ座標は負の値となるので、正負を反転している。

## Vanishing Points

この論文のアプローチは、スケッチの線は概ねいずれかの軸に平行であるという前提に基づいている。軸に平行な線は、Image plane上ではVanishing pointへ向かう。これを利用した制約を使う。



上図を見ると分かるとおり、

整理すると、

これを行列の形で表すと、

(3)

## Structure

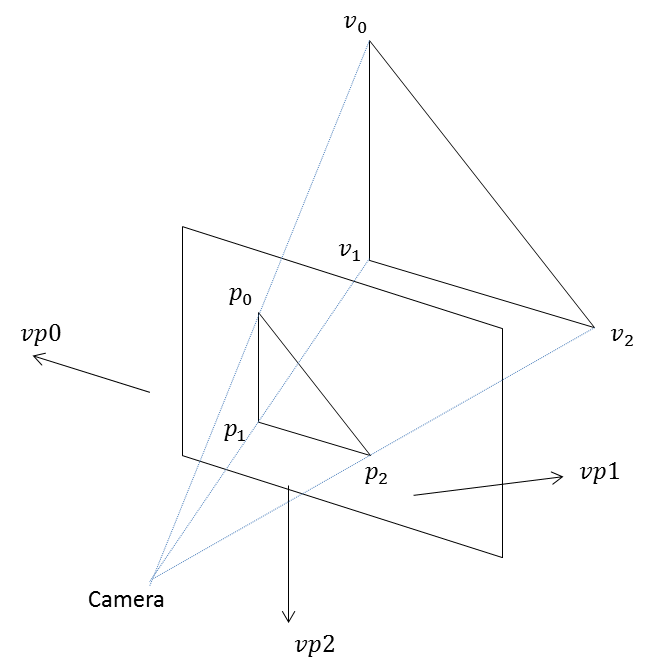
実装していない。

そもそも、与えられたスケッチに対して、primitive形状との2Dトポロジーマッチングを行うとのことだが、どうやってマッチングを行うかの詳細が記載されていない。Geometric graphのマッチングは簡単ではないはず。

しかも、スケッチが複数のprimitive形状から構成されている時や、一部が隠れている場合などは、そもそもマッチングできない。この辺りについても、一切説明がない。

## まとめ

以上から、例えば以下のような三角形に関する制約をの形で表す。



後は、行列をSVD分解しての最終列がの解だ！

なお、右辺が0なので、のスケールは求められない。幾何的には、2Dに射影された像は、カメラに近くにあるか、遠くにあるか、判定しようがないということだ（たとえエッジの直交などを考慮しても）。

そこで、現状は、スケッチの頂点のY座標が最も小さい頂点の世界座標系でのY座標が0であるという仮定をつかってスケールを計算することにする。

まず、ビュー変換行列をとすると、その逆行列は

よって、

つまり、

ここで、としたいので、

したがって、