Brief discussion on 3D pose estimation based on article: ”3D pose estimation of visual markers”

Author: Antonio Badal Regàs

Aziel Martins de Freitas Júnior

SENAI CIMATEC

Contents

Introduction

Motivation

Study arised from the need to understand, analyze and improve a computer vision software based on the *ARToolKitPlus* software library.

ARToolKit

- First article published in 1999;
- Intended for augmented reality in collaborative office work;
- One user would wear a HMD (*head mounted display*) to provide others augmented reality visuals.

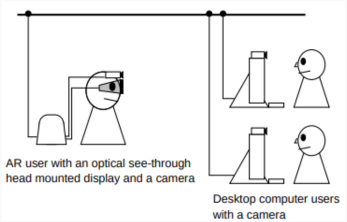

AR user with an optical see-through head mounted display and a camera
Desktop computer users with a camera

Figure 1: Intended usage.

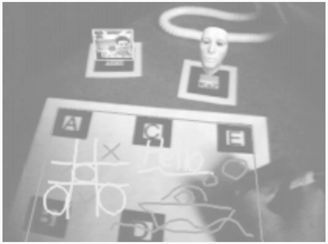


Figure 2: HMD image.

MBV: Marker-based vision systems

- Dependent on **fiducial markers**;
- *ARtag* are commonly used;
- Tag family for this study was not specified;
- *ARToolKitPlus* system was used because of it's accurate readings.

Square and circular ARtags

Cons: square AR tags have the disadvantages of occlusion and minimum size detection when in comparison.

Pros: square have low false negative and very low confusion rate when used in groups.

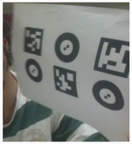


Figure 3: Examples of tags used by the author. Only the square ones were used in the article.

Concept - space transformation

The act of fitting a space image information into a plane;

Typical of pinhole cameras;

Two space transformations **may happen** to the acquired image: a rotation and a translation.

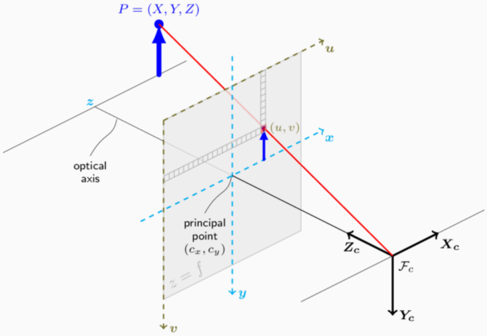


Figure 4: Projection example.

Core concept - space transformation

Transformation represented by:

$$v_i = K \times \mathcal{R} \times p_i + \mathcal{T}$$

v_i is each projected p_i point of the real world object. K depends on the camera lenses. The world object **may have suffered** a rotation $\mathcal{R}$ and a translation $\mathcal{T}$:

$$s \cdot \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = K_{3\times 3} \times \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \times \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} T_x \\ T_y \\ T_z \end{pmatrix} \tag{1}$$

Goal - obtain $\mathcal{R}$ and $\mathcal{T}$ with iterative error minimization via Newton-Raphson's method.

Goals

- Obtain applicable information on marker identification;
- Examine software problems and improve item;
- Test, criticize and improve.

Problem presentation

Pose jump

What is pose jump?

- Tag kept in place while RViz shows tag frame in two different places and orientations;

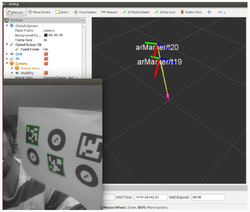


Figure 5: First tag placement.

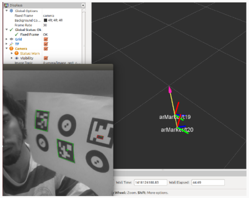
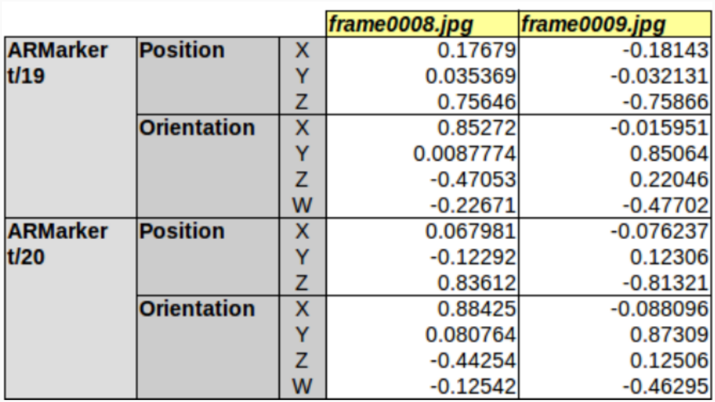


Figure 6: Second tag placement.

Pose jump table



			frame0008.jpg	frame0009.jpg
ARMarker t/19	Position	X	0.17679	-0.18143
		Y	0.035369	-0.032131
		Z	0.75646	-0.75866
	Orientation	X	0.85272	-0.015951
		Y	0.0087774	0.85064
		Z	-0.47053	0.22046
		W	-0.22671	-0.47702
ARMarker t/20	Position	X	0.067981	-0.076237
		Y	-0.12292	0.12306
		Z	0.83612	-0.81321
	Orientation	X	0.88425	-0.088096
		Y	0.080764	0.87309
		Z	-0.44254	0.12506
		W	-0.12542	-0.46295

Figure 7: Different placement in two different moments.

Placement over time

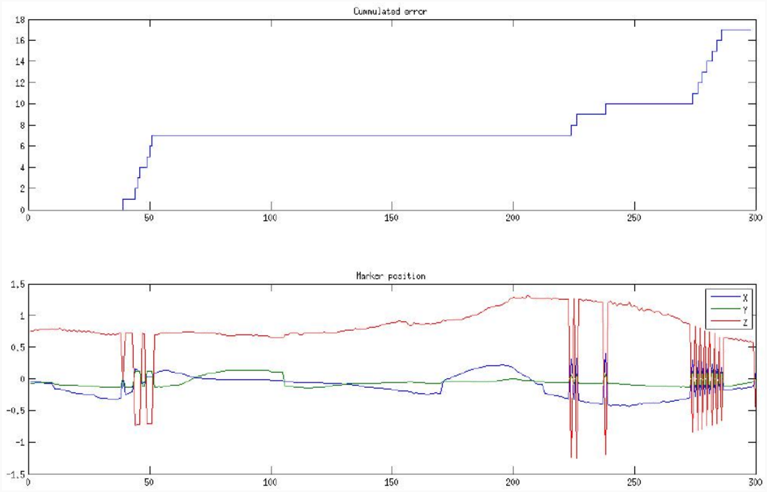


Figure 8: Frame placement along 300 frames.

Symmetry hint

Clearer sign that symmetry is part of the problem.

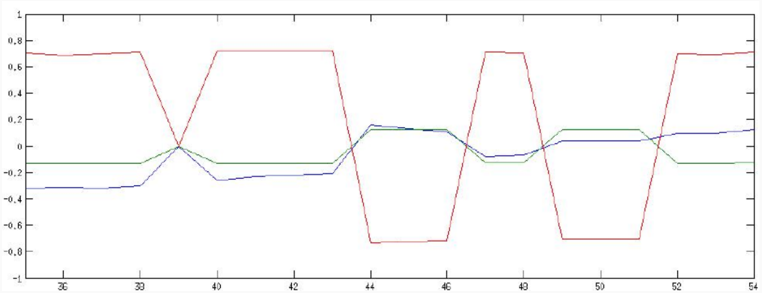


Figure 9: Abrupt changes of Z are origin symmetric.

Software operation and possible issues

Operation steps

- Camera calibration;
- Marker detection;
- Pose calculation;
- Pose visualization.

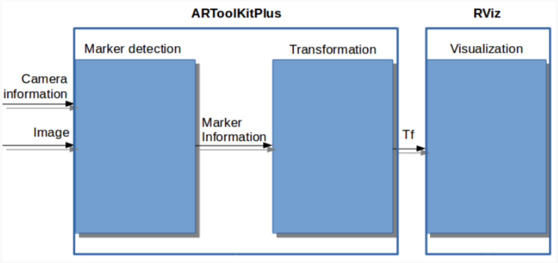


Figure 10: Steps to the output.

Core process: marker detection

- Thresholding;
- Labeling;
- Contour detection;
- Vertex localization;
- Tag identification and localization.

Thresholding

Establish intensity limits to tag readability;

Each pixel is evaluated:

$$r + g + b < \text{threshold} \times 3$$

Saves processing efforts on Labeling step.

Labeling - image segmentation

Evaluation of image continuity by pixel surroundings analysis.

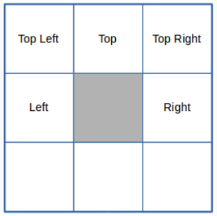


Figure 11: Matrix for labeling process.

A pixel shares it's neighbors label whenever they are continuous.

Contour detection

Intuitive contour discerning method.

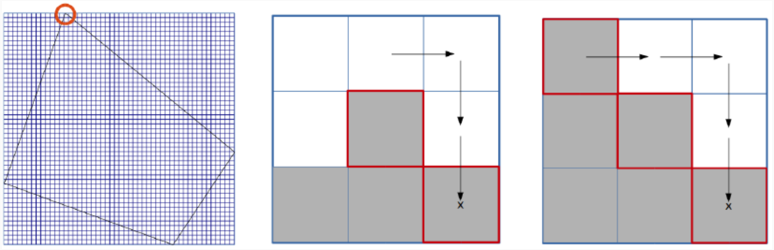


Figure 12: Contour detection process.

Vertex location

Most distant point pair is located;
Line is traced between them;
Other vertices are found.

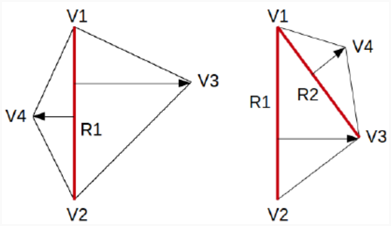
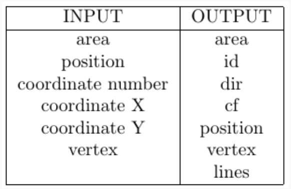


Figure 13: Vertex location.

Why all this?

The computations made before provide the inputs;
If evaluation results in a valid tag, there will be outputs;
The outputs will then provide for the pose estimation.



INPUT	OUTPUT
area	area
position	id
coordinate number	dir
coordinate X	cf
coordinate Y	position
vertex	vertex
	lines

Figure 14: Partial table for evaluated tag.

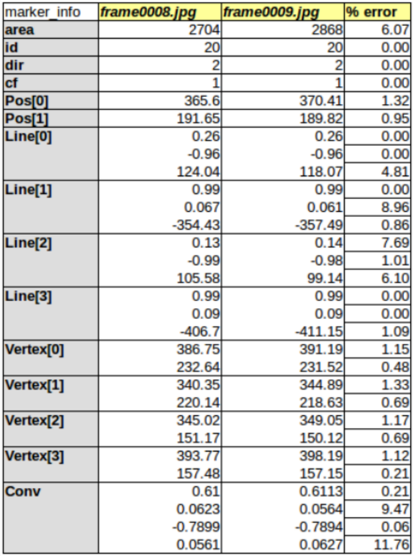
Iterative pose transformation

Newton-Raphson iterative method for minima is applied;
Targets: $\mathcal{R}$ and $\mathcal{T}$ of each vertex.

- Initial guess for $\mathcal{R}$ and $\mathcal{T}$;
- Guess is applied to the 3D-2D transformation equation;
- The residuals x and y for each vertex are calculated;
- Each pair x_i, y_i of residue for each vertex is used to compute the corrections to translation and rotation;
- Iteration ends when residues are small enough.

The culprit

The author excludes camera calibration, marker detection and quaternion transformation as reasons for the pose jump based on the low **% error** values seen in the table. That leaves the iteration as the reason.



marker_info	*frame0008.jpg*	*frame0009.jpg*	% error
area	2704	2868	6.07
id	20	20	0.00
dir	2	2	0.00
cf	1	1	0.00
Pos[0]	365.6	370.41	1.32
Pos[1]	191.65	189.82	0.95
Line[0]	0.26	0.26	0.00
	-0.96	-0.96	0.00
	124.04	118.07	4.81
Line[1]	0.99	0.99	0.00
	0.067	0.061	8.96
	-354.43	-357.49	0.86
Line[2]	0.13	0.14	7.69
	-0.99	-0.98	1.01
	105.58	99.14	6.10
Line[3]	0.99	0.99	0.00
	0.09	0.09	0.00
	-406.7	-411.15	1.09
Vertex[0]	386.75	391.19	1.15
	232.64	231.52	0.48
Vertex[1]	340.35	344.89	1.33
	220.14	218.63	0.69
Vertex[2]	345.02	349.05	1.17
	151.17	150.12	0.69
Vertex[3]	393.77	398.19	1.12
	157.48	157.15	0.21
Conv	0.61	0.6113	0.21
	0.0623	0.0564	9.47
	-0.7899	-0.7894	0.06
	0.0561	0.0627	11.76

Figure 15: Pose jump error accounted.

Fix 1: error threshold

The iteration error threshold should be turned off so the optimization algorithm does one attempt only;

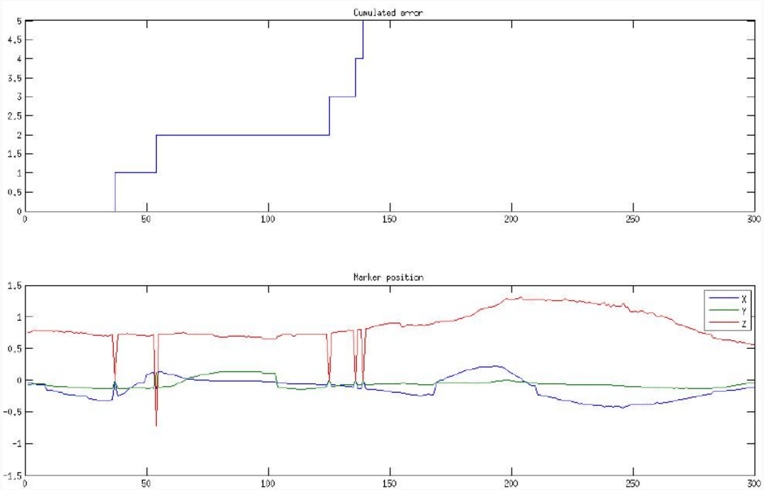


Figure 16: Less position jumps registered.

Fix 2: limit iteration space

Iteration algorithm attempts 30 sweeps of 15°;

Reduction to sweeps of 5°;

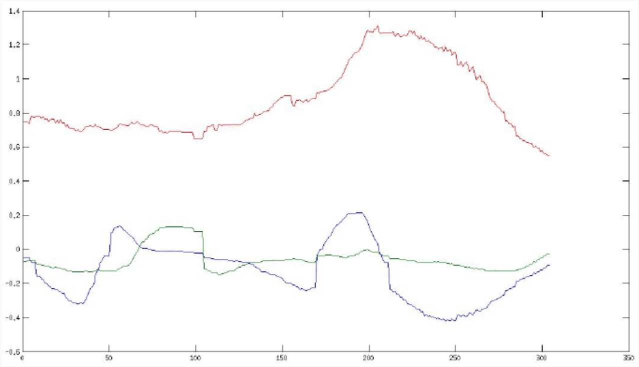


Figure 17: No position jump. Result of both fixes.