Institute of Systems and Robotics
University of Coimbra
Department of Electrical and Computer Engineering

Contactless Registration

Estimation of Normals using Affine Correspondences

Diogo Emanuel Ribas Vaz

December 19, 2018

Contents

1 Introduction

The main goal of this report is to summarize the work done by Diogo Vaz in order to develop a contactless registration algorithm to be used in arthroscopic orthopedic surgery context. The algorithm should perform an automatic alignment of a 3 dimensional bone model, obtained before the surgery by Computed Tomography or Magnetic Resonance Imaging, and the respective real bone of a patient during the surgery.
Diogo's work focused on creating a method to estimate a sparse 3D model of the patient anatomy during the medical procedure. This method should compute points and surface normals by exploiting the geometric information encoded in affine correspondences.
The registration itself is done with Carolina's registration algorithm which receives as input the pre-operative model (obtained with CT or MRI) and the intra-operative model (obtained with Diogo's method). The output is a rigid transformation that aligns the two models.
The proposed reconstruction method was implemented in two different approaches: **feature matching** and **tracking**, explained respectively in sections 2 and 3.

2 Sparse Model Computation - Feature Matching Approach (FMA)

The registration based on feature matching requires feature detection in each image and establishment of feature correspondences between different images. Thereby, the generic steps of the method are:

1. Feature detection on images;
2. Feature matching between pairs of images;
3. Filtering out matches outside the boundary, over markers or not verifying the epipolar constraint;
4. Refinement of affine correspondences;
5. Estimation of points and normals;
6. Refinement of normals.

Since these steps could be implemented in several different ways, four implementation were created and are explained in the next subsections.

2.1 Version 1 (FMA-V1)

In the context of arthroscopy, the acquired images have a significant radial distortion. On the other hand, the majority of feature detectors and features descriptors are suitable for images without radial distortion and their use with radial distorted images leads to a poor description of features and hence a reduction of matches.
In this regard, in the first version of pipeline, the radial distortion is corrected and the pipeline is designed to be applied in undistorted images. This version has the following steps:

1. Feature detection is done on undistorted images, using a covariant feature detector from VLFeat library or the standard sift algorithm. Both detectors return features composed by a point and a 2x2 matrix. However, while the first returns an affine matrix, the second returns a similarity matrix;
2. The matches are computed, using a standard matcher based on the squared distance of SIFT feature descriptors;

3. All matches outside the boundary, over markers or disrespecting the epipolar constraint are discarded;
4. The remaining matches are refined, using an affine tracker. Note that when using standard sift algorithm, this step is more than a simple refinement because the initialization is a similarity matrix (a small subset of the generic affine matrices) instead of an affine matrix. Some degrees of freedom of the affine transformation are computed for the first time in this step;
5. Knowing the motion and intrinsic parameters of the camera and an affine correspondence, it is possible to compute a 3D point and the surface normal in this point;
6. Refinement of normals with a homography-based tracker.

All these steps are represented in figure 1.

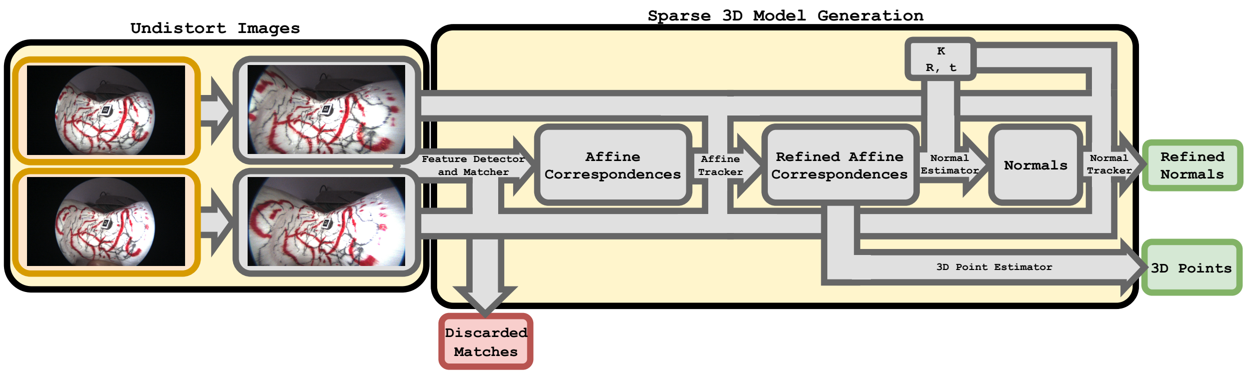

Figure 1: Schematic representation of the first version of pipeline.

2.2 Version 2 (FMA-V2)

The second version of the pipeline results from a small modification of version 1. The affine tracker formulation for undistorted images was replaced by a new affine tracker to be directly applied on distorted images and all the other algorithm steps were maintained. Figure 2 shows the second version structure.

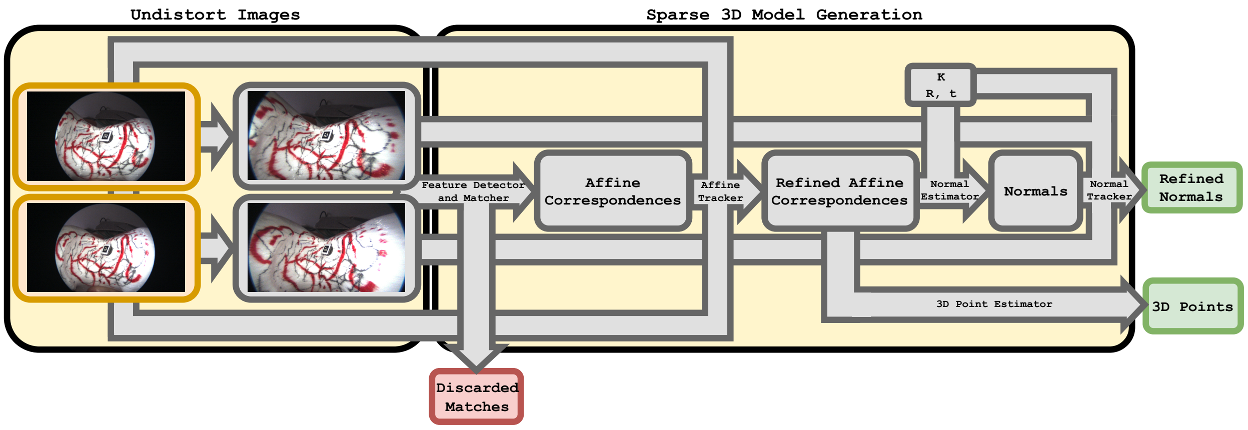

Figure 2: Schematic representation of the first version of pipeline.

2.3 Version 3 (FMA-V3)

As mentioned in subsection 2.1, commonly, feature detectors and descriptors are only applicable on undistorted images and, in this application, images have distortion. However there are some algorithms to perform these tasks directly on distorted images, avoiding distortion correction. One of them is the sRD-SIFT, a modified sift algorithm to deal with radial distortion during detection and description stages of SIFT algorithm.
The third version of the method is similar to second, but introduces sRD-SIFT to detect and describe features on distorted images. This version is represented in figure 3.

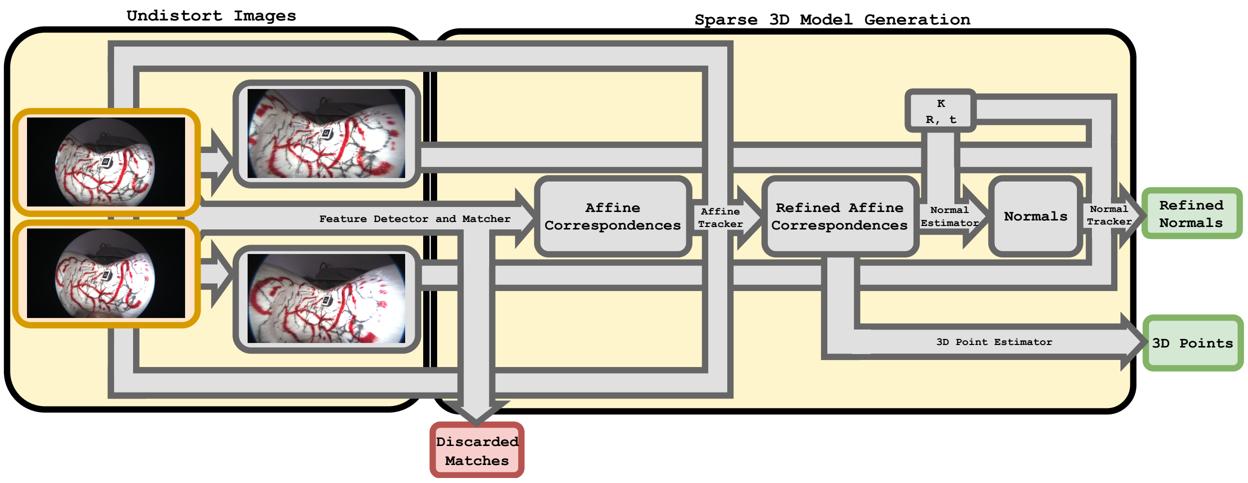

Figure 3: Schematic representation of the first version of pipeline.

2.4 Version 4 (FMA-V4)

At last, in version 4 the homography-based tracker used to refine normals passes to operate over distorted images. Thereby, in this version, the distorted images are no longer necessary, because the whole pipeline can be used directly on undistorted images. The schematic representation in figure 4 depicts the final structure of the algorithm.

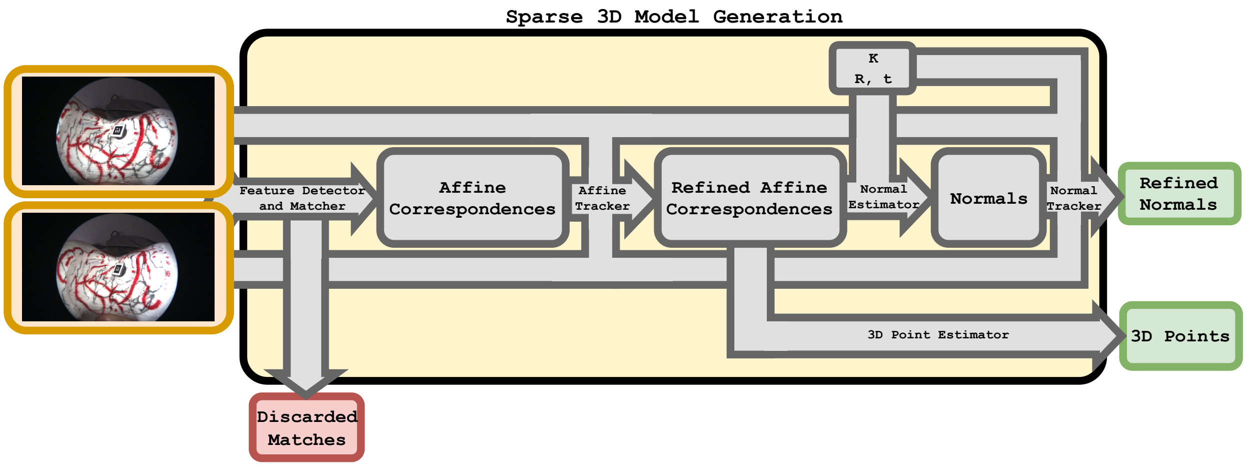

Figure 4: Schematic representation of the first version of pipeline.

3 Sparse Model Computation - Tracking Approach (TA)

In arthroscopy context, the matching process can be a hard task since bone has a low-textured surface and there are suspended particles in the liquid inside the knee during the surgery. In order to avoid the matching process, it was developed a tracker-based approach. In this case, given a set of features on an image, an affine tracker is used to compute the correspondent features on a second image, estimating one affine transformation per feature and hence an affine correspondence. The steps of this approach are:

1. Feature detection on undistorted images with sRD-SIFT, whenever the number of tracked features are lower than a certain threshold;
2. Feature tracking using an affine tracker in order to compute affine correspondences;
3. Filtering out tracklets out of the boundary, over the marker or disrespecting the epipolar constraint;
4. Estimation of 3D points and normals, using information encoded in the affine tracklets;
5. Refinement of normals using an homography-based tracker.

4 Formulation of Trackers

A tracker to perform image alignment is an algorithm that apply image transformations on a template to minimize the difference between that template and an image. In practice, a tracker has the goal of finding an image warp $W(\mathbf{p}, \mathbf{x})$ that minimizes the cost function

$$\sum_{\mathbf{x}} \left[I(W(\mathbf{x}, \mathbf{p})) - T(\mathbf{x})\right]^2 . \tag{1}$$

During this section, all formulations of trackers used in the sparse model computation are detailed, as well as the adaption for radial distorted images.

4.1 Adaptation for Radial Distorted Images

4.2 Affine KLT Tracker

The affine KLT tracker was implemented with an inverse composition formulation to take advantage of its computational efficiency as reported in ?. In ?, it is demonstrated how to formulate a tracker based on affine transformations.
Assuming an affine warp with the following parameterization:

$$\mathbf{W}(\mathbf{x}, \mathbf{p}) = \begin{bmatrix} p_1 + 1 & p_3 & p_5 \\ p_2 & p_4 + 1 & p_6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix} \tag{2}$$

being $\mathbf{p} = [\, p_1 \quad p_2 \quad p_3 \quad p_4 \quad p_5 \quad p_6]^T$ a vector of warp parameters and $\mathbf{x} = [\, x_1 \quad x_2]^T$ a pixel to be warped, the least-squares minimization can be done by an iterative update of warp parameters ($\mathbf{p}$) with

$$\boldsymbol{\Delta}\mathbf{p} = H^{-1} \sum_{\mathbf{x}} \left[\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]^T \left[I(\mathbf{W}(\mathbf{x}, \mathbf{p})) - T(\mathbf{x})\right] \tag{3}$$

where H is the Hessian matrix

$$H = \sum_{\mathbf{x}} \left[\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]^T \left[\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right] \tag{4}$$

and $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ is the jacobian of the warp evaluated at $\mathbf{p} = \mathbf{0}$.

$$\frac{\partial \mathbf{W}}{\partial \mathbf{p}} = \begin{bmatrix} x & 0 & y & 0 & 1 & 0 \\ 0 & x & 0 & y & 0 & 1 \end{bmatrix} = \left.\frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right|_{\mathbf{p}=\mathbf{0}} \tag{5}$$

A significant part of expression 3 can be computed before the iterative part of the algorithm, because the terms $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ (evaluated at $\mathbf{p} = \mathbf{0}$), ∇T and hence H^{-1} are constant. This is the reason for the efficiency of inverse compositional formulations.
After computing the update, the warp estimate should be updated with the equation below.

$$\mathbf{W}(\mathbf{x}, \mathbf{p}) \leftarrow \mathbf{W}(\mathbf{x}, \mathbf{p}) \circ \mathbf{W}(\Delta\mathbf{x}, \mathbf{p}) \Leftrightarrow \mathbf{W}(\mathbf{x}, \mathbf{p}) = \mathbf{W}(\mathbf{W}(\Delta\mathbf{x}, \mathbf{p}), \mathbf{p}) \tag{6}$$

To summarize, the algorithm steps are the following:

- Pre-computation part:
  1. Compute template gradients ($\nabla T(\mathbf{x})$);
  2. Evaluate the warping jacobian ($\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$) at $(\mathbf{x}, \mathbf{0})$;
  3. Compute steepest descent images ($\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}}$);
  4. Compute the inverse of Hessian matrix (H^{-1}).
- Iterative part:
  1. Warp I with $\mathbf{W}(\mathbf{x}, \mathbf{p})$ to compute $I(\mathbf{W}(\mathbf{x}, \mathbf{p}))$;
  2. Compute update $\Delta\mathbf{p}$ using equation 3;
  3. Update the warp as described in equation 6.

4.3 Affine Motion-Constrained KLT Tracker (Forward Additive)

The tracker explained in subsection 4.2 assumes that the affine transformation between the template and the image can be anyone. However this affine transformation is constrained by the camera motion and only a small part of the affine transformations verify the motion constraints. The affine motion-constrained KLT tracker uses the known motion to ensure the estimated affine transformation is coherent with the motion constraints.
Given an affine correspondence $(A, \mathbf{x}, \mathbf{y})$

$$A = \begin{bmatrix} a_1 & a_3 \\ a_2 & a_4 \end{bmatrix}, \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \text{ and } \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \tag{7}$$

and the essential matrix E

$$E = \begin{bmatrix} e_1 & e_4 & e_7 \\ e_2 & e_5 & e_8 \\ e_3 & e_6 & e_9 \end{bmatrix} \tag{8}$$

the following matrix equation is verified:

$$\begin{bmatrix} y_1 + a_1 x_1 & y_2 + a_2 x_1 & 1 & a_1 x_2 & a_2 x_2 & 0 & a_1 & a_2 & 0 \\ a_3 x_1 & a_4 x_1 & 0 & y_1 + a_3 x_2 & y_2 + a_4 x_2 & 1 & a_3 & a_4 & 0 \\ x_1 y_1 & x_1 y_2 & x_1 & x_2 y_1 & x_2 y_2 & x_2 & y_1 & y_2 & 1 \end{bmatrix} \mathbf{e} = \mathbf{0} \tag{9}$$

being $\mathbf{e} = [e_1 \quad e_2 \quad e_3 \quad e_4 \quad e_5 \quad e_6 \quad e_7 \quad e_8 \quad e_9]^T$.
Assuming that the essential matrix and $\mathbf{x}$ are known, it is possible to manipulate equation 9 in order to put the parameters of A and $\mathbf{y}$ in evidence, as shown below.

$$\underbrace{\begin{bmatrix} x_1e_1+x_2e_4+e_7 & x_1e_2+x_2e_5+e_8 & 0 & 0 & e_1 & e_2 & e_3 \\ 0 & 0 & x_1e_1+x_2e_4+e_7 & x_1e_2+x_2e_5+e_8 & e_4 & e_5 & e_6 \\ 0 & 0 & 0 & 0 & x_1e_1+x_2e_4+e_7 & x_1e_2+x_2e_5+e_8 & x_1e_3+x_2e_6+e_9 \end{bmatrix}}_{C} \mathbf{c} = \mathbf{0} \tag{10}$$

where $\mathbf{c} = [a_1 \quad a_2 \quad a_3 \quad a_4 \quad y_1 \quad y_2 \quad 1]^T$. Since $\mathbf{y} = A\mathbf{x} + \mathbf{b}$, equation 10 can be written as

$$\underbrace{\begin{bmatrix} 2x_1e_1+x_2e_4+e_7 & 2x_1e_2+x_2e_5+e_8 & x_2e_1 & x_2e_2 & e_1 & e_2 & e_3 \\ x_1e_4 & x_1e_5 & x_1e_1+2x_2e_4+e_7 & x_1e_2+2x_2e_5+e_8 & e_4 & e_5 & e_6 \\ x_1^2e_1+x_1x_2e_4+x_1e_7 & x_1^2e_2+x_1x_2e_5+x_1e_8 & x_1x_2e_1+x_2^2e_4+x_2e_7 & x_1x_2e_2+x_2^2e_5+x_2e_8 & x_1e_1+x_2e_4+e_7 & x_1e_2+x_2e_5+e_8 & x_1e_3+x_2e_6+e_9 \end{bmatrix}}_{D} \mathbf{d} = \mathbf{0} \tag{11}$$

with $\mathbf{d} = [a_1 \quad a_2 \quad a_3 \quad a_4 \quad b_1 \quad b_2 \quad 1]^T$.
The equations of system 11 are constraints introduced by the camera motion to the general affine transformations. The null-space of D ($N = Null(D)$) constitutes a basis of the space of affine motion-constrained transformations, allowing to obtain a new parameterization of them. Thereby, this subset of transformations can be written as linear combination of the columns of N,

$$\mathbf{d} = \underbrace{\begin{bmatrix} n_1 & n_4 & n_7 & n_{10} \\ n_2 & n_5 & n_8 & n_{11} \\ n_3 & n_6 & n_9 & n_{12} \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{N} \mathbf{r} \tag{12}$$

and $\mathbf{r} = [p_1 \quad p_2 \quad p_3 \quad p_4]^T$. From last equation of system 12, $p_4 = 1$, so the final parameterization is

$$\mathbf{a} = \begin{bmatrix} n_1 & n_4 & n_7 & n_{10} \\ n_2 & n_5 & n_8 & n_{11} \\ n_3 & n_6 & n_9 & n_{12} \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix} \tag{13}$$

where $\mathbf{a} = [a_1 \quad a_2 \quad a_3 \quad a_4 \quad b_1 \quad b_2]^T$ and $\mathbf{p} = [p_1 \quad p_2 \quad p_3]^T$.
Since this subset of affine transformations is not a group, the affine motion-constrained tracker cannot be implemented with an inverse compositional formulation. For this reason, it was implemented with a forward additive formulation. Given the affine motion-constrained warp function

$$\mathbf{W}(\mathbf{x}, \mathbf{p}) = \begin{bmatrix} n_1p_1 + n_4p_2 + n_7p_3 + n_{10} & n_3p_1 + n_6p_2 + n_9p_3 + n_{12} & p_2 \\ n_2p_1 + n_5p_2 + n_8p_3 + n_{11} & p_1 & p_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{14}$$

the minimization of cost function 1 can be done by an iterative update of warp parameters ($\mathbf{p}$) with

$$\mathbf{\Delta p} = H^{-1} \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^T [I(\mathbf{W}(\mathbf{x}, \mathbf{p})) - T(\mathbf{x})] \tag{15}$$

where H is the Hessian matrix

$$H = \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]^T \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right] \tag{16}$$

and $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ is the jacobian of the warp. In contrast to inverse compositional formulation, the terms of equation 15 are not constant during the iterative part. The warp parameters are updated using equation 17.

$$\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p} \tag{17}$$

The algorithm steps are presented below.

- Iterative part:
  1. Warp I with $\mathbf{W}(\mathbf{x}, \mathbf{p})$ to compute $I(\mathbf{W}(\mathbf{x}, \mathbf{p}))$;
  2. Warp gradients of ∇I with $\mathbf{W}(\mathbf{x}, \mathbf{p})$
  3. Evaluate the Jacobian $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ at $(\mathbf{x}, \mathbf{p})$
  4. Compute update $\Delta\mathbf{p}$ using equation 15;
  5. Update the warp as described in equation 17.

Since this tracker is specialized for affine motion-constrained transformations, the initial affine transformation provided as input of the tracker should be projected to the affine motion-constrained space before the tracking process.

4.4 Affine Motion-Constrained KLT Tracker (Inverse Compositional)

In order to create an inverse compositional formulation of the affine motion-constrained tracker, the model must be modified. Considering that a transformation can be written as

$$A_n = A_{n-1} A_i^{-1} \Leftrightarrow A_i = A_n^{-1} A_{n-1} \tag{18}$$

where A_n, A_{n-1} and A_i are respectively a final transformation, an initial transformation and an adjustment, an inverse compositional formulation turns out to be reasonable to find matrix A_i. Based on the new parameterization, introduced in equation 13, $\mathbf{p}_{n-1}$ is the vector of parameters of A_{n-1} (also represented as $A_{n-1}(\mathbf{p}_{n-1})$), $\mathbf{p}_n$ is the vector of parameters of A_n (or $A_n(\mathbf{p}_n)$) and $\mathbf{p}_n = \mathbf{p}_{n-1} + \mathbf{p}_i$, where $\mathbf{p}_i$ is the ajustment vector. The tracker optimizes vector $\mathbf{p}_i$.
Assuming this model, the warping function is shown in equation 19.

$$\mathbf{W}(\mathbf{x}, \mathbf{p}_i) = A_n^{-1}(\mathbf{p}_n) A_{n-1}(\mathbf{p}_{n-1})\mathbf{x} = A_n^{-1}(\mathbf{p}_{n-1} + \mathbf{p}_i) A_{n-1}(\mathbf{p}_{n-1})\mathbf{x} \tag{19}$$

The updates of $\mathbf{p}_i$ are computed using equation 3 and applied with equation 6. Although this tracker has been formulated as an inverse compositional tracker, the jacobian must be modified in each iteration because it is dependent of $\mathbf{p}_{n-1}$ and this vector changes in each iteration. For this reason, this formulation is not so efficient as a common inverse compositional formulation.

4.5 Homography-based KLT Tracker (Inverse Compositional)

The tracker proposed in this subsection is based on homographic transformations in order to refine surface normals. This idea consists of assuming that a surface is locally planar in the neighborhood of a point (figure 5). Given a 3D point $\mathbf{x}_p$, over a locally planar surface, that is viewed by cameras C_0

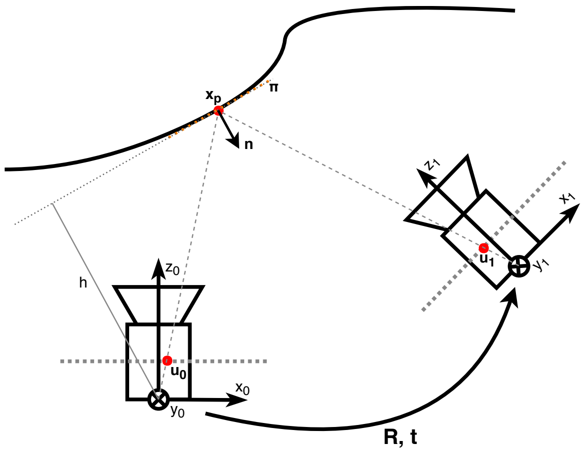

Figure 5: Schematic representation of geometric parameters to describe two cameras and a locally planar surface.

and C$_1$ and projected respectively at pixels with coordinates $\mathbf{u}_0$ and $\mathbf{u}_1$ ($\mathbf{u}_{0h}$ and $\mathbf{u}_{1h}$ in projective coordinates), equation 20 is verified.

$$\mathbf{u}_{1h} = \underbrace{K_1(R\mathbf{x}_p^T\mathbf{n} + \mathbf{t}\mathbf{n}^T)K_0^{-1}}_{H}\mathbf{u}_{0h} \tag{20}$$

K_1 and K_0 are respectively the intrinsic matrices of cameras C$_1$ and C$_0$, R and $\mathbf{t}$ are the rotation matrix and the translation vector between cameras C$_0$ and C$_1$ and $\mathbf{n}$ is the surface normal vector. With the purpose of formulating an inverse compositional tracker, a homography can be written as

$$H_n = H_{n-1}H_i^{-1} \Leftrightarrow H_i = H_n^{-1}H_{n-1} \tag{21}$$

where H_n, H_{n-1} and H_i are respectively a final homography, an initial homography and an adjustment. The tracker is used to estimate the adjustment H_i to modify H_{n-1}, in each iteration. The optimization of normals can be done through two possible approaches: **refining only the normal vector** and **refining the distance to the origin ($h = |\mathbf{x}_p^T\mathbf{n}|$) and the normal vector**.

4.5.1 Refinement of Normal Vector

This refinement consists of optimizing the normal vector direction. For this, it is modified using

$$\mathbf{n}_n = \mathbf{n}_{n-1} + \underbrace{\alpha\bar{\mathbf{x}}_\alpha + \beta\bar{\mathbf{x}}_\beta}_{\mathbf{n}_i} \tag{22}$$

where $\bar{\mathbf{x}}_\alpha$ and $\bar{\mathbf{x}}_\beta$ are two unit vectors, which are perpendicular to $\mathbf{x}_p$ and to each other.
Assuming that camera calibration and motion are known, the homography can be parameterized with the normal vector, being $\mathbf{n}_n$ the vector of parameters of H_n (represented by $H_n(\mathbf{n}_n)$) and $\mathbf{n}_{n-1}$ the vector of parameters of H_{n-1} (or $H_{n-1}(\mathbf{n}_{n-1})$). Based on this parameterization the warping function can be written as

$$\mathbf{W}(\mathbf{x}, \mathbf{p}_i) = H_n(\mathbf{n}_n)^{-1}H_{n-1}(\mathbf{n}_{n-1}) = H_n(\mathbf{n}_{n-1} + \mathbf{n}_i)^{-1}H_{n-1}(\mathbf{n}_{n-1}) \tag{23}$$

being $\mathbf{p}_i = [\alpha \quad \beta]^T$, the vector of parameters optimized by the tracker. Similarly to the inverse compositional formulation of the affine motion-constrained tracker the update is computed with equation 3 and applied using equation 6, recomputing the jacobian in each iteration.

4.5.2 Refinement of Distance to the Origin and Normal Vector

This refinement consists of optimizing the locally planar surface parameters: distance to the origin (moving $\mathbf{x}_p$) and normal direction. In order to obtain homography parameterization for that goal, equation 20 is manipulated as shown in equation 24.

$$\mathbf{u}_{1h} = \underbrace{K_1(R + \mathbf{t}\mathbf{n}'^T)K_0^{-1}}_{H}\mathbf{u}_{0h} \text{ with } \mathbf{n}' = \frac{\mathbf{n}}{\mathbf{x}_p^T \mathbf{n}} \tag{24}$$

Assuming that camera calibration and motion are known, homographies can be defined by the scaled normal vector ($\mathbf{n}'$). The update of scaled normals is done using expression below

$$\mathbf{n}'_n = \mathbf{n}'_{n-1} + \mathbf{n}'_i \tag{25}$$

where $\mathbf{n}'_i$ is the adjustment of parameters. So, defining $\mathbf{n}'_n$ as the vector of parameters of H_n (represented by $H_n(\mathbf{n}'_n)$) and $\mathbf{n}'_{n-1}$ as the vector of parameter of H_{n-1} (or $H_{n-1}(\mathbf{n}'_{n-1})$), the warping function is

$$\mathbf{W}(\mathbf{x}, \mathbf{p}_i) = H_n(\mathbf{n}'_n)^{-1} H_{n-1}(\mathbf{n}'_{n-1}) = H_n(\mathbf{n}'_{n-1} + \mathbf{p}_i)^{-1} H_{n-1}(\mathbf{n}'_{n-1}) \tag{26}$$

with $\mathbf{p}_i = \mathbf{n}'_i$. The tracker performs the optimization of $\mathbf{p}_i$ using equations 3 and 6.