AIBO Camera Stabilization

Tom Stepleton

Ethan Tira-Thompson

16-720, Fall 2003

AIBO vision is *bumpy*

- Legged locomotion induces vibration.
- Head (camera mount) is a great big cantilevered mass.

Camera problems: the lineup

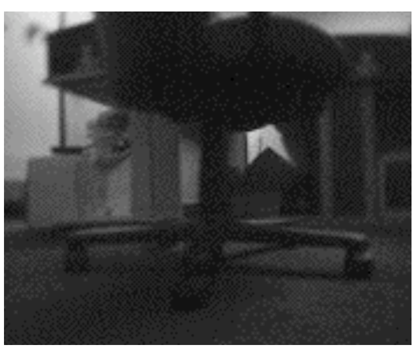
- Obvious problems due to exposure time/cheap optics:

Smear

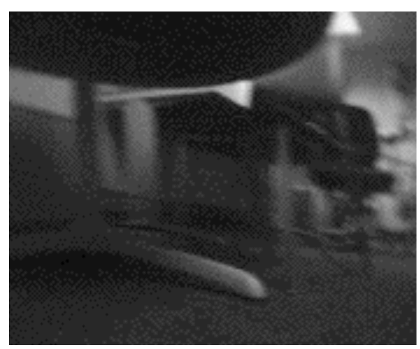
Lens distortion

Camera problems: the lineup

- Subtle problems due to sample rate:



Stretching

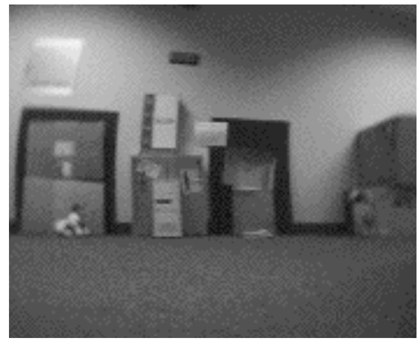


Skew/bending

Camera problems: the lineup

- Subtle problems due to sample rate:

Stretching

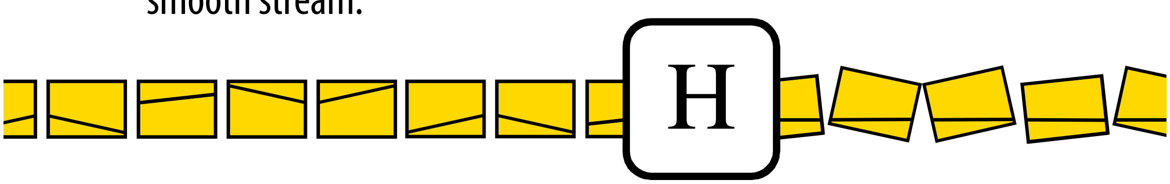


Skew/bending

Goal: Take AIBO from this…

Stabilization Basics

- Compute homographies between successive images in your sequence.
- Transform sequence images one by one to make a continuous, smooth stream.



- Problems: error accumulates, especially with more degrees of freedom (e.g. affine transformations).
    - Many papers are about dealing with this.

Mosaicing Video Sequences
Netzer, Gotsman

- Suggests using a sliding window of multiple images to compute more accurate registration of each frame
  - We ignore this, too computationally intensive, introduces lag in images
- Treat each new image as one of translation, rigid, similarity, affine, or projective transformation. Try each, pick the one with the lowest error, with some bonus to "simpler" transformations.
  - Instead of trying all 5 on each frame at run time, we did some trials and found rigid transformations satisfactory.

Camera Stabilization Based on 2.5D Motion Estimation and Inertial Motion Filtering
Zhu, Xu, Yang, Jin

- Typically, camera movements fall into a few classes of motion. (e.g. panning, dolly, tracking,…) We can pass through movement on the dominant dimension and stabilize on the non-dominant dimension.
  - Since our motions are typically constrained to the horizontal plane, we can compensate for vertical bouncing and rotation, but leave horizontal motion unchecked.

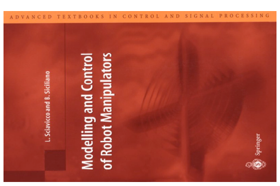
Our approach

- AIBO vibration is very regular.
- Rotation oscillates around 0°.

- Vertical bouncing oscillates around a fixed value.

- Horizontal bouncing oscillates around a value determined by AIBO's turning and sidestepping velocities (which we know!).

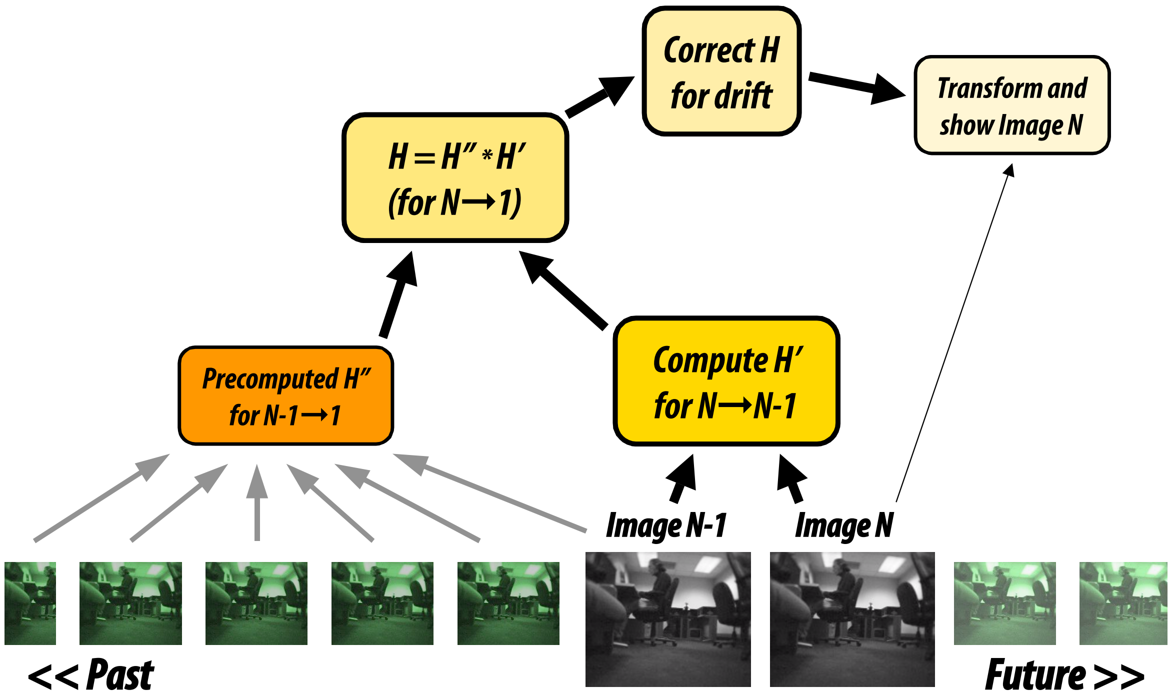
So…



It's a control problem now!

- Over many frames, image motion should tend toward fixed (or predictable) values.
- Use an "image placement controller" that allows high-frequency changes in placement, but enforces this constraint.
- Specifically, we extract and adjust the x,y coordinates and θ rotation used for image registration.

The obligatory flowchart



How to find corresponding points

- **Q:** How do you find corresponding points in Image N and Image N-1?
- **A:** Andrew and Ranjith's **RANSAC** from Assignment 5.
- **Q:** Oh.
- **A:** It's pretty robust, even to blurry, smeared images.

Why find H indirectly?

- We could simply find the H from Image N to the ***corrected, transformed*** Image N-1, right?

- ***Wrong-o!*** The corrected image is jagged, noisy. RANSAC would ***freak***.

- Instead, first find H′ from Image N to the ***normal*** Image N-1.

- Then premultiply it with the accumulated H (H″) from the normal Image N-1 to Image 1.

How do we get x, y, and θ?

- It's easy if our "homography" is just a rigid transform.

arccotan $\rightarrow \theta$

$$\begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

- It's easy to adjust them, too.
- A cop out? Perhaps… it doesn't fix all the aberrations in the AIBO image. It's a start, though.

Finding rigid transformations (1 of 2)

- Step 1: Constrained least squares.

$$\begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \Longrightarrow \begin{bmatrix} a & -b & c \\ b & a & d \\ 0 & 0 & e \end{bmatrix}$$

P: Image N point
u,v: N+1 point

$$\begin{array}{l} -m_1^\top P_i + u_i m_3^\top P_i = 0 \\ -m_2^\top P_i + v_i m_3^\top P_i = 0 \end{array} \Longrightarrow \begin{array}{l} -aP_i(1) + bP_i(2) - cP_i(3) + 0 + u_i e P_i(3) = 0 \\ -bP_i(1) - aP_i(2) + 0 - cP_i(3) + v_i e P_i(3) = 0 \end{array}$$

$$\begin{bmatrix} & & \ddots & & \\ -P_i(1) & P_i(2) & -P_i(3) & 0 & u_i P_i(3) \\ -P_i(2) & -P_i(1) & 0 & -P_i(3) & v_i P_i(3) \\ & & \ddots & & \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \end{bmatrix} = \mathbf{0}$$

The U matrix

Finding rigid transformations (2 of 2)

- Step 2: Throw away image scaling.

$$\begin{bmatrix} a & -b & c \\ b & a & d \\ 0 & 0 & e \end{bmatrix}$$

[a/e b/e] is not constrained by constrained least squares to be of length 1.

- Divide *a*, *b*, *c*, *d*, and *e* by *e*, then by the length of [*a b*].
- Otherwise the image will shrink as you walk forward.

Fighting drift, step by step

- Isolate θ from H
- Apply really simple correction: premultiply H by a rotation matrix that rotates by $-\theta/constant$ (forcing it back toward 0).
- Isolate t_x and t_y from H
- Apply similar correction. We should force t_y toward a predicted value based on turning and sidestepping. We don't right now, forcing it to 0 instead.

Demo time!

- Hallway scene
  - Normal, stabilized, and side-by-side
- Lab scene
  - Normal, stabilized, and side-by-side

Any questions?