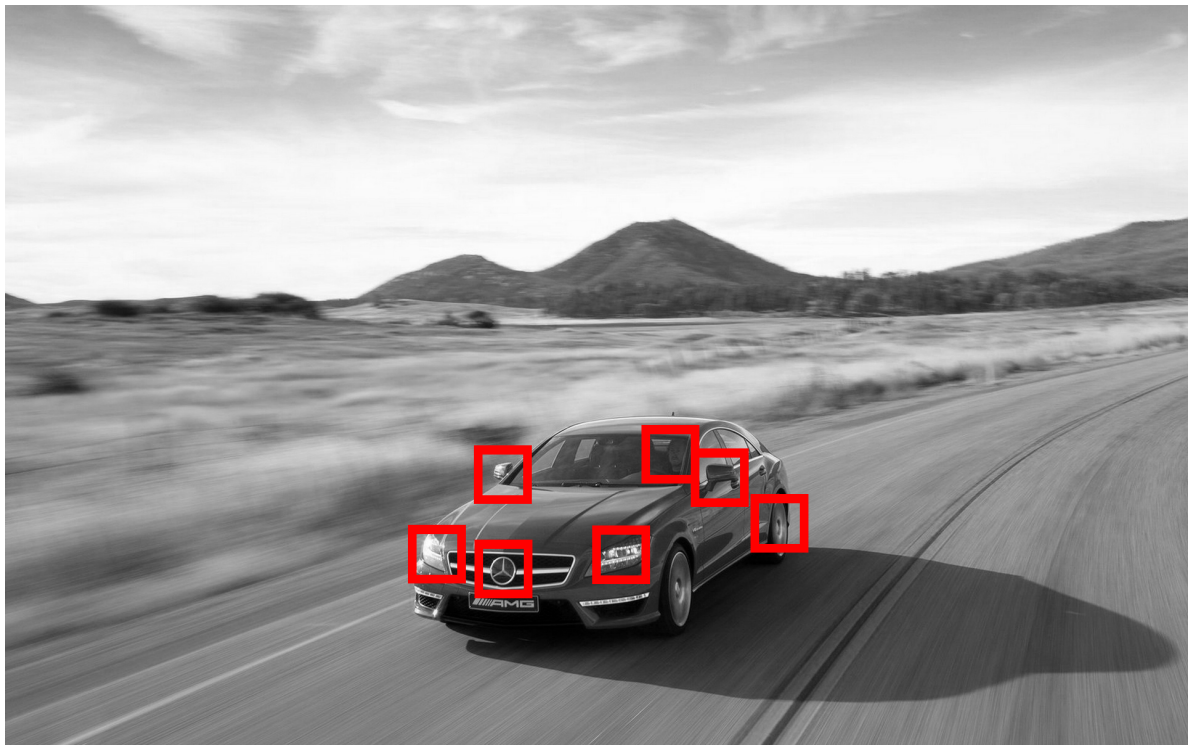
Using Subspace Constraints to Improve Feature Tracking
Presented by Bryan Poling



Based on work by Bryan Poling, Gilad Lerman, and Arthur Szlam

What is Tracking?

Broad Definition

Tracking, or Object tracking, is a general term for following some "thing" through multiple frames of a video sequence, and determining that things location in each frame.

This is a very broad problem, and there are several sub-problems in tracking.

What is Tracking?

Big Question #1

- How do we characterize the target?
  - Color or intensity histogram
  - Texture composition
  - Spatial configuration (sub-image)
  - Outline or Silhouette
  - Target autocorrelation pattern (Irani)

Big Question #2

- What do we assume about the nature of the target and its possible motion?
  - Rigid/Semi-Rigid/Non-Rigid
  - Smooth motion
  - Limited velocity/Limited changes in scale
  - Will the targets characterization evolve?

Feature Tracking

Feature Tracking

Feature tracking is a sub-problem in tracking. We make the following assumptions:

- The target is well-characterized by a template image.
- The targets motion is limited from one frame to the next.
- The targets signature (our characterization of the target) may change is many ways, but its evolution is slow (between successive frames it changes only a small amount).

Why Make These Assumptions?

The assumptions above tend to be valid for small, visually distinctive objects in high-framerate motion video.

Feature Tracking

Lucas Kanade Framework

In feature tracking we try and minimize the "Fit Residual":

$$\epsilon(\mathbf{p}) = \sum_{\mathbf{x}} [I(W(\mathbf{x}, \mathbf{p})) - T(\mathbf{x})]^2 \quad (1)$$

I = Current frame of video (image function)
T = Template for feature (image function)
W = "Warp" function: $\mathbb{R}^2 \times \mathbb{R}^D \rightarrow \mathbb{R}^2$

Feature Tracking

Lucas Kanade Framework

$$\epsilon(\mathbf{p}) = \sum_{\mathbf{x}} [I(W(\mathbf{x}, \mathbf{p})) - T(\mathbf{x})]^2$$

I = Current frame of video (image function)
T = Template for feature (image function)
W = "Warp" function: $\mathbb{R}^2 \times \mathbb{R}^D \rightarrow \mathbb{R}^2$

The Warp function is thought of as a domain transformation for the image function I ($\mathbb{R}^2 \rightarrow \mathbb{R}^2$) with controlling parameter $\mathbf{p}$. The warp function will have a particular form (chosen in advance). We are trying to find the domain transformation that makes the Image and the template agree as much as possible (minimize l_2 distance between them).

Feature Tracking

Lucas Kanade Framework

$$\epsilon(\mathbf{p}) = \sum_{\mathbf{x}} [I(W(\mathbf{x}, \mathbf{p})) - T(\mathbf{x})]^2$$

I = Current frame of video (image function)
T = Template for feature (image function)
W = "Warp" function: $\mathbb{R}^2 \times \mathbb{R}^D \rightarrow \mathbb{R}^2$

Frequently, the warp function is chosen to represent simple translation. The parameter $\mathbf{p}$ is a vector in $\mathbb{R}^2$ and the warp function has the form:

$$W(\mathbf{x}, \mathbf{p}) = \mathbf{x} + \mathbf{p}$$

This is sufficient because we need only model the transformation that the feature experiences between two successive frames.

Feature Tracking

(a) Image, I

(b) Template, T

Feature Tracking

(c) Good Match

(d) Bad Match

(e) Bad Match

Feature Tracking

Lucas Kanade Framework

$$\epsilon(\mathbf{p}) = \sum_{\mathbf{x}} [I(W(\mathbf{x}, \mathbf{p})) - T(\mathbf{x})]^2$$

I = Current frame of video (image function)
T = Template for feature (image function)
W = "Warp" function: $\mathbb{R}^2 \times \mathbb{R}^D \rightarrow \mathbb{R}^2$

Minimization Strategy

The objective function is:

- Non-Convex
- Possibly Non-Smooth

How do we minimize it?

Feature Tracking

Minimization Strategy

- Exhaustive search for minimum in some search window. This is not as bad as it sounds since we know where the feature was in the last frame - one of our assumptions is that it can't go far between 2 frames.
- Pretend $\epsilon(\mathbf{p})$ is "nice" and use calculus-based minimization methods. This should be faster than exhaustive search.

Iterative Gradient-Based Registration

We can use an iterative approach to minimizing $\epsilon(\mathbf{p})$. In each step we compute a good direction to shift $\mathbf{p}$ and a good amount (by using derivative information). We take that step. Repeat until we converge.

Feature Tracking

1^{st}-Order Approach

Your step is -1 times a small multiple of the gradient of $\epsilon(\mathbf{p})$.
This is reliable, but takes lots of iterations to converge.

2^{nd}-Order Approach - Gauss-Newton

Locally fit a quadratic surface to $\epsilon(\mathbf{p})$ (using derivative information). Analytically find the minimum of the quadratic surface. Take the step which moves you to that minimum.
This will typically converge in very few iterations (like 3 or 4), but it can be "finicky" since it uses 2^{nd}-order info on a function that depends on real data. The Lucas Kanade method uses this scheme.

Feature Tracking

Enhancements and Extensions

- A simple modification to the Lucas Kanade formulation allows for processing color video: $\epsilon(\mathbf{p}) = \sum_{\mathbf{x}} |I(W(\mathbf{x}, \mathbf{p})) - T(\mathbf{x})|^2$
- Coarse-2-Fine: Build an image pyramid for the Image and the template. Start with the lowest resolution copy of both and match the template to the image. Successively refine your solution on finer and finer resolution levels. This offers a very big improvement and is essential for a good feature tracker.

Live Demo

Structure

Definition

Assume a single feature is tracked through L frames of video. Let (x_i, y_i) denote the position of the feature in frame i. Then the following vector is called the "trajectory" or the "track" of the feature:

$$\mathbf{v} = \begin{pmatrix} x_1 \\ y_1 \\ \vdots \\ x_L \\ y_L \end{pmatrix} \tag{2}$$

Structure

Definition

Imagine that we track multiple features on a single rigid body, and we consider the set of feature tracks (a set of vectors in $\mathbb{R}^{2L}$). We will call this a "track set" for the rigid object.

Features On a Rigid Body Don't Move Arbitrarily

Not every set of vectors in $\mathbb{R}^{2L}$ is realizable as a track set for a rigid body.

Structure

Definition

For a track set to be consistent with the motion of a rigid body, some relationships between the different tracks must be obeyed. We will refer to a collection of relationships between feature tracks as a "structure observation". It represents a formal and precise way of quantifying some type of structure in a scene.

Affine Camera Model

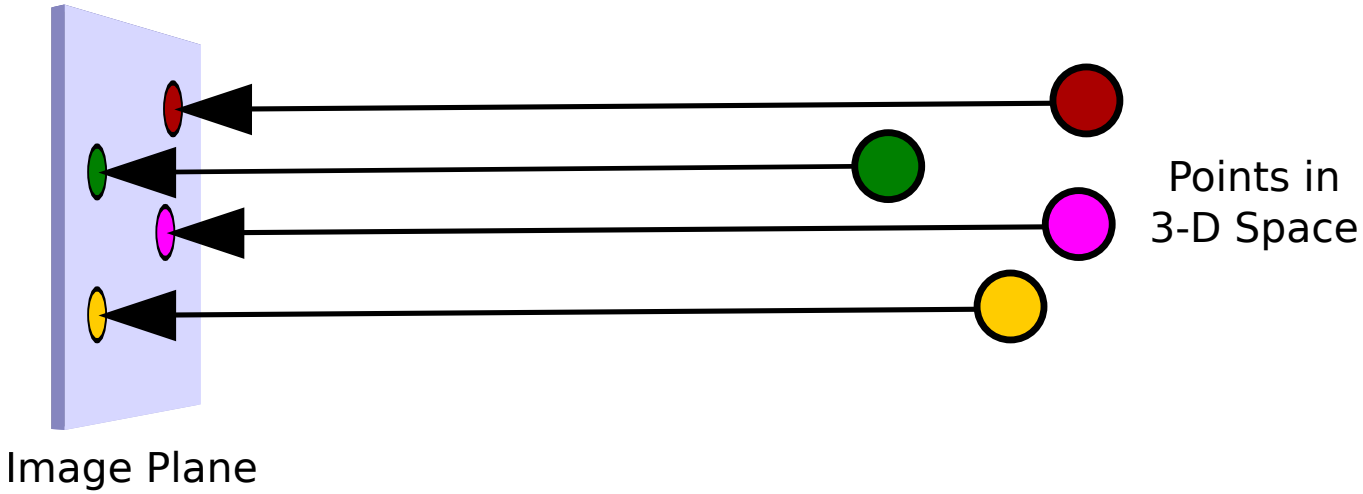
There are multiple structure observations for rigid bodies. We will derive a structure observation based on the affine camera model.

Detour: The Affine Camera Model

The Simplest Camera Model... Orthographic

In The Beginning...

From a theoretical perspective, a camera maps points in 3D space into a plane, called the "image plane", or the "focal plane". A camera model defines this projection operation. The most basic camera model is the orthographic model. This model approximates a cameras projection with orthogonal projection onto a subspace.



The Simplest Camera Model... Orthographic

Homogeneous Coordinates

Homogeneous Coordinates are convenient for expressing and manipulating transformations and projections that occur in camera modeling. The homogeneous vector $[x, y, t]^T$ corresponds to the euclidean vector $(x/t, y/t)^T$. When the last coordinate in a homogeneous vector is 1, the vector is said to be in "standard homogeneous coordinates".

Watch This...

The affine transformation $\mathbf{Y} = \mathbf{RX} + \mathbf{C}$ in Euclidean coordinates becomes the following when using homogeneous coordinates:

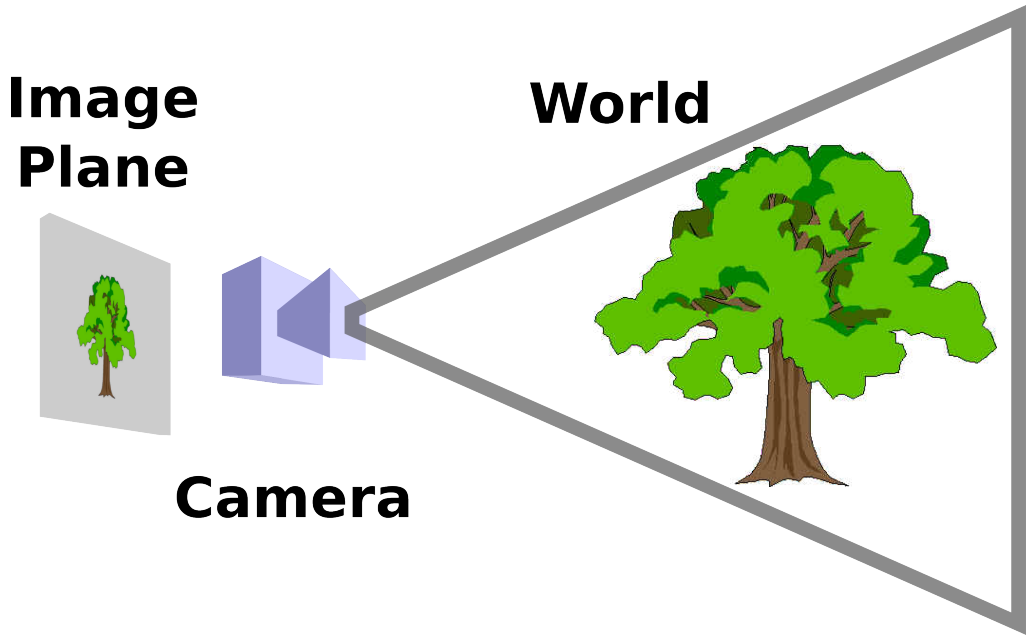
$$\mathbf{Y} = \left(\begin{array}{ccc|c} & \mathbf{R} & & \mathbf{C} \\ 0 & 0 & 0 & 1 \end{array} \right) \mathbf{X} \tag{3}$$

The Simplest Camera Model... Orthographic

Coordinate Frames

There are 3 main coordinate frames used in defining camera models.

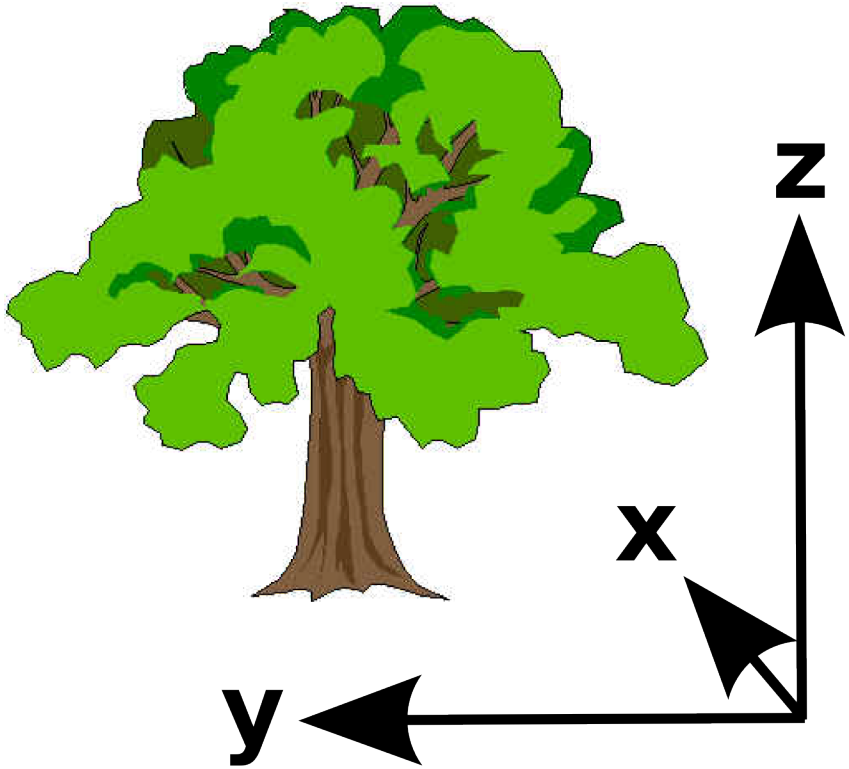
- The World Frame
- The Camera Frame
- Image Coordinates



The Simplest Camera Model... Orthographic

The World Frame

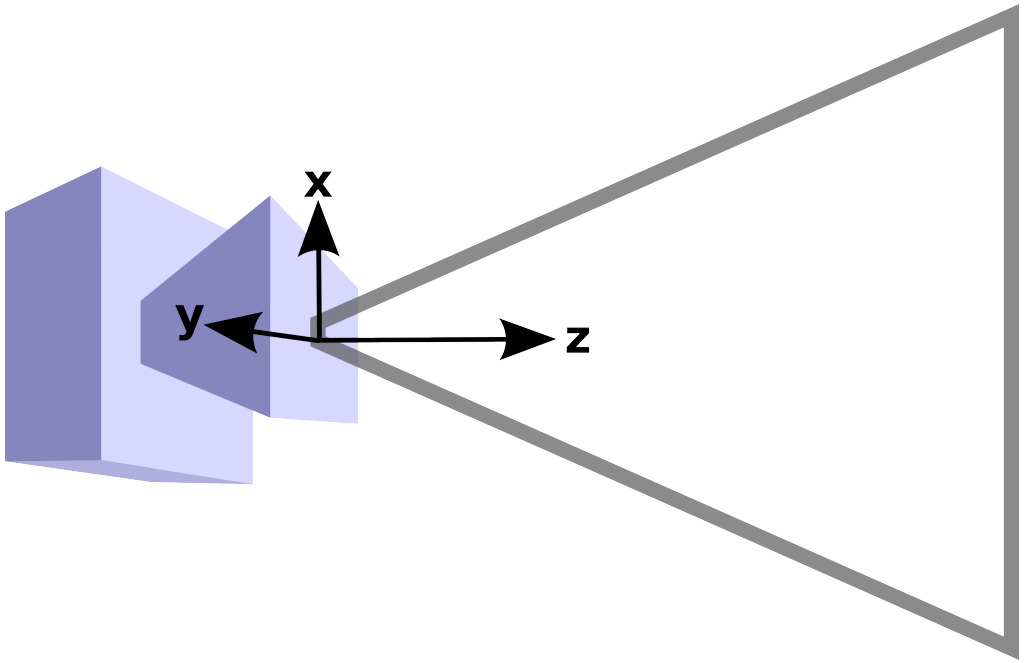
This is a 3D coordinate system fixed in the world we are imaging.



The Simplest Camera Model... Orthographic

The Camera Frame

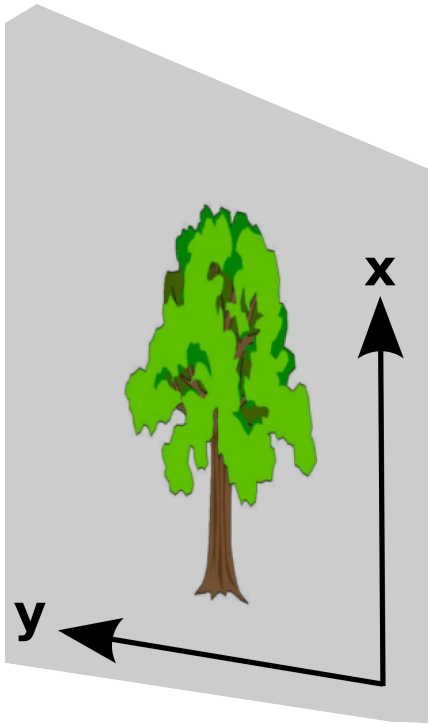
This is a 3D coordinate system fixed to our camera. The z-axis (also called the "optical axis") points at our target.



The Simplest Camera Model... Orthographic

The Image Frame

The image frame is a coordinate system in our focal plane. It's origin can be chosen however is convenient, although often the camera center is chosen (the image of the camera frame origin). The x and y axes are parallel to the images of the x and y camera axes, respectively.



The Simplest Camera Model... Orthographic

The Orthographic Camera

Let $\mathbf{X}_{world}$ represent the position of a point in 3D space in standard homogeneous coordinates, in the world frame. To derive the orthographic camera model, we first express $\mathbf{X}_{world}$ in the camera frame (denoted $\mathbf{X}_{cam}$), and then perform the projection. The world and camera frames are both standard 3D coordinate frames. They are related by some rotation and translation. Assume that $\mathbf{R}$ and $\mathbf{C}$ represent the rotation and translation between these frames so that:

$$\mathbf{X}_{cam} = \left(\begin{array}{ccc|c} & \mathbf{R} & & \mathbf{C} \\ 0 & 0 & 0 & 1 \end{array} \right) \mathbf{X}_{world} \tag{4}$$

The Simplest Camera Model... Orthographic

The Orthographic Camera

Let $\mathbf{x}$ denote the projection of $\mathbf{X}_{cam}$ in the image plane (using homogeneous coordinates). To perform the projection we just eliminate the z-component of the vector. This is accomplished as follows:

$$\mathbf{x} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \mathbf{X}_{cam} \tag{5}$$

The Simplest Camera Model... Orthographic

The Orthographic Camera

Then, using our relationship between the world and camera frames, we get our final camera model:

$$\mathbf{x} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \left(\begin{array}{ccc|c} & \mathbf{R} & & \mathbf{C} \\ 0 & 0 & 0 & 1 \end{array} \right) \mathbf{X}_{world} \tag{6}$$

From The Orthographic To The Affine Camera

Problems With The Orthographic Camera

- Since the orthographic camera is nothing more than orthogonal projection onto a subspace, the overall scale of objects is preserved. This in not realistic though for regular cameras (A photo of a car, for instance, would be physically very large).
- The orthographic camera model defines the origin of your image frame for you. It may not be the origin you want.

From The Orthographic To The Affine Camera

Problems With The Orthographic Camera

- Since the orthographic camera is nothing more than orthogonal projection onto a subspace, the overall scale of objects is preserved. This in not realistic though for regular cameras (A photo of a car, for instance, would be physically very large).
- The orthographic camera model defines the origin of your image frame for you. It may not be the origin you want.

The Fix

To address these problems we apply a scaling and translation after orthographic projection.

From The Orthographic To The Affine Camera

Modified Orthographic Camera

$$\mathbf{x} = \begin{pmatrix} \alpha & 0 & \gamma \\ 0 & \beta & \delta \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \left(\begin{array}{ccc|c} & \mathbf{R} & & \mathbf{C} \\ 0 & 0 & 0 & 1 \end{array} \right) \mathbf{X}_{world}$$

This simply scales the result of orthographic projection by α in the x-direction and β in the y-direction, and then shifts by (γ, δ)

The Affine Camera

We can generalize this camera model by replacing our scaling of the x and y axes with an arbitrary linear transformation **T**:

$$\mathbf{x} = \begin{pmatrix} & \mathbf{T} & \begin{matrix}\gamma\\ \delta\end{matrix} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \left(\begin{array}{ccc|c} & \mathbf{R} & & \mathbf{C} \\ 0 & 0 & 0 & 1 \end{array}\right) \mathbf{X}_{world}$$

Multiplying these matrices out and changing variables gives:

$$\mathbf{x} = \begin{pmatrix} T_{11} & T_{12} & T_{13} & T_{14} \\ T_{21} & T_{22} & T_{23} & T_{24} \\ 0 & 0 & 0 & 1 \end{pmatrix} \mathbf{X}_{world} \tag{7}$$

where each T_{ij} is arbitrary.

From The Orthographic To The Affine Camera

The Affine Camera

If we want to then go to euclidean coordinates in the image plane, we just drop the trailing 1 (this requires that $\mathbf{X}_{world}$ be expressed in *standard* homogeneous coordinates).

$$\mathbf{x} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} T_{11} & T_{12} & T_{13} & T_{14} \\ T_{21} & T_{22} & T_{23} & T_{24} \\ 0 & 0 & 0 & 1 \end{pmatrix} \mathbf{X}_{world} \tag{8}$$

This is written more simply as:

$$\mathbf{x} = \mathbf{P}\mathbf{X}_{world} \tag{9}$$

where $\mathbf{P}$ is an arbitrary 2-by-4 matrix. This $\mathbf{P}$ will be called the "camera projection matrix", or more simply just the "camera matrix" for our camera.

End Detour: Deriving The Structure Observation

Structure

Recall...

We set out to derive a set of relationships that must be obeyed by a set of feature trajectories if they all belong to a single rigid body.

Let us fix our "world frame" to our rigid body. Let $\mathbf{X}_f$ denote the position of feature f in the world frame (we drop the subscript *world* for readability). Because our world frame is fixed to the same body that all of our features belong to, a features position in the world frame does not change with time.

We Have A Moving Rigid Body

Now, imagine that our rigid body is moving with respect to our camera. This means that at each instant we have a different rotation and translation from our camera frame to our world frame. Thus, for each frame of video, we will have a different camera matrix. Let $\mathbf{P}_l$ denote the camera matrix for frame l.

$\mathbf{P}_l\mathbf{X}_f$ gives the euclidean image coordinates of feature f in frame l.

The Trackpoint Matrix

We define a matrix holding the observed image coordinates of each feature in each frame. This is called the "history matrix" or the "trackpoint matrix". Assume we observe F features for L frames of video.

$$\mathbf{M} = \begin{bmatrix} \mathbf{P}_1\mathbf{X}_1 & \mathbf{P}_1\mathbf{X}_2 & \dots & \mathbf{P}_1\mathbf{X}_F \\ \mathbf{P}_2\mathbf{X}_1 & \mathbf{P}_2\mathbf{X}_2 & \dots & \mathbf{P}_2\mathbf{X}_F \\ \vdots & \vdots & & \vdots \\ \mathbf{P}_L\mathbf{X}_1 & \mathbf{P}_L\mathbf{X}_2 & \dots & \mathbf{P}_L\mathbf{X}_F \end{bmatrix} \tag{10}$$

This matrix has dimensions $2L$-by-F.

The Decomposition Of The Trackpoint Matrix

From the way we constructed the trackpoint matrix, there is an obvious decomposition:

$$\mathbf{M} = \begin{bmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \vdots \\ \mathbf{P}_L \end{bmatrix}_{2L \times 4} \begin{bmatrix} \mathbf{X}_1 & \mathbf{X}_2 & \dots & \mathbf{X}_F \end{bmatrix}_{4 \times F} \tag{11}$$

The Decomposition Of The Trackpoint Matrix

From the way we constructed the trackpoint matrix, there is an obvious decomposition:

$$\mathbf{M} = \begin{bmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \vdots \\ \mathbf{P}_L \end{bmatrix}_{2L \times 4} \begin{bmatrix} \mathbf{X}_1 & \mathbf{X}_2 & \dots & \mathbf{X}_F \end{bmatrix}_{4 \times F} \tag{12}$$

From this decomposition we see that $\mathbf{M}$ has rank less than or equal to 4. That is, the columns of the trackpoint matrix live in a 4-dimensional subspace of $\mathbb{R}^{2L}$.

It Gets Better

The last coordinate of each $\mathbf{X}_f$ is 1 (since we used *standard* homogeneous coordinates). Thus the columns of

$$\begin{bmatrix} \mathbf{X}_1 & \mathbf{X}_2 & \dots & \mathbf{X}_F \end{bmatrix}_{4 \times F} \tag{13}$$

actually live is an affine subspace of $\mathbb{R}^4$ of dimension 3.

Thus, the columns of $\mathbf{M}$ actually live in an affine subspace of $\mathbb{R}^{2L}$ of dimension 3 or less.

Structure

Our Structure Observation

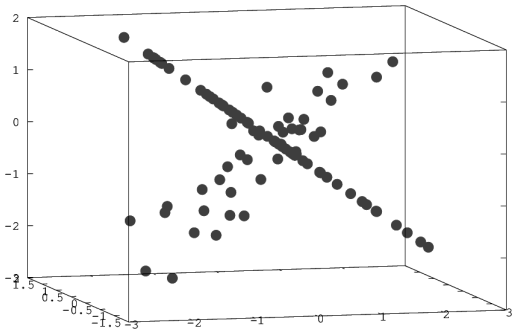
If a set of features belong to the same rigid body, then the associated trajectories are confined to an affine subspace of dimension no more than 3.

Footnote

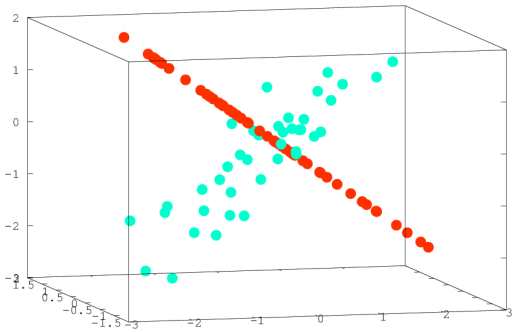
It turns out that in general, track sets from different rigid bodies (moving differently) give rise to different affine subspaces.

Application To Segmentation

This structure observation has been successfully applied to motion segmentation. You track features and generate trajectories from the observed feature positions in each frame. Segmentation is accomplished by identifying different subspaces in the set of trajectories and associating each trajectory to its nearest subspace.



(f) Tracks



(g) Segmented Tracks

(h) Segmentation

Exploiting Structure To Improve Feature Tracking

Exploiting Structure For Better Feature Tracking

The Total Energy Function

The first step is to merge all of the single-feature energy functions used for tracking into a global energy function.

$$C(\mathbf{x}) = \sum_{f=1}^{F} \sum_{\mathbf{u} \in \Omega} \psi \left(T_f(\mathbf{u}) - I(\mathbf{u} + \mathbf{x}_f) \right) \tag{14}$$

Now, we can effect a rigid body requirement by imposing constraints on the minimization of this total energy function. This can be done in a strong sense or a weak sense.

Exploiting Structure For Better Feature Tracking

Causality

In this problem, it is important to keep causality in mind. For our algorithm to be useful practically, it should not require knowledge of the future. If a solution were to require information from L frames into the future, our tracking scheme would have a theoretical latency of L frames.

Exploiting Structure For Better Feature Tracking

First Solution

?

Exploiting Structure For Better Feature Tracking

First Solution

Our first solution might be to track features without a constraint from the last frame to the current. Then fit a 3-dim affine subspace to the trajectories over the last L frames and project the trajectories into that subspace. Take the projected feature positions as the state for the current frame.

Problems With The First Solution

- Our constraint requires explicitly modeling motion. The results are very sensitive to errors in our subspace estimation.
- This is a track-then-fix solution. We start by tracking without constraint. If we make catastrophic errors in the initial tracking, they are hard to fix because they make it harder to model the motion.
- Fixed dimensionality. The 3D constraint does not allow us to infer the precise position of a single feature, given the rest.
- Cannot "encourage" the constraint... can only "demand" it.

Exploiting Structure For Better Feature Tracking

Our Solution

Impose the constraint in the weak sense by adding a penalty term to the energy function. This term "disincentivises" configurations of features that are inconsistent with rigid motion, given knowledge of the past.

$$\bar{C}(\mathbf{x}) = \alpha \sum_{f=1}^{F} \sum_{\mathbf{u} \in \Omega} \psi \left(T_f(\mathbf{u}) - I(\mathbf{u} + \mathbf{x}_f) \right) + P(\mathbf{x}) \qquad (15)$$

The Penalty Term

$P(\mathbf{x})$ is an estimate of, or proxy for, the dimensionality of the set of feature trajectories over the last several frames of video (including the current frame).

Exploiting Structure For Better Feature Tracking

$$\bar{C}(\mathbf{x}) = \alpha \sum_{f=1}^{F} \sum_{\mathbf{u} \in \Omega} \psi \left(T_f(\mathbf{u}) - I(\mathbf{u} + \mathbf{x}_f) \right) + P(\mathbf{x})$$

The Penalty Term Details

- Past feature locations are treated as constants, so P is a function only of the current state, $\mathbf{x}$.
- The terms before $P(\mathbf{x})$ are called "fit terms". We added α as a coefficient to the sum of the fit terms to weight the penalty.
- Several forms of P were tested, with different levels of success.

Exploiting Structure For Better Feature Tracking

$$\bar{C}(\mathbf{x}) = \alpha \sum_{f=1}^{F} \sum_{\mathbf{u} \in \Omega} \psi \left(T_f(\mathbf{u}) - I(\mathbf{u} + \mathbf{x}_f) \right) + P(\mathbf{x})$$

Forms for $P(\mathbf{x})$

Let $\mathbf{M}$ denote the trackpoint matrix for the last L frames of video (plus the current frame).

- $P(\mathbf{x}) = \|\mathbf{x} - \Pi(\mathbf{x})\|$ where $\Pi(\mathbf{x})$ is the projection of $\mathbf{x}$ onto the best-fit 3D affine subspace to the columns of $\mathbf{M}$ (in l_2 sense).
- $P(\mathbf{x}) = \|\mathbf{M}\|_*$
- $P(\mathbf{x}) = d_\epsilon(\mathbf{M})$ where d_ϵ denotes empirical dimension.

Exploiting Structure For Better Feature Tracking

$$\bar{C}(\mathbf{x}) = \alpha \sum_{f=1}^{F} \sum_{\mathbf{u} \in \Omega} \psi \left(T_f(\mathbf{u}) - I(\mathbf{u} + \mathbf{x}_f) \right) + P(\mathbf{x})$$

Forms for $P(\mathbf{x})$

Each form for $P(\mathbf{x})$ tested offered some improvement over not using a constraint. The form that appeared to work best was $P(\mathbf{x}) = d_\epsilon(\mathbf{M})$. Recall the definition of $d_\epsilon(\mathbf{M})$:

$$d_\epsilon(\mathbf{M}) := \frac{\|\boldsymbol{\sigma}\|_\epsilon}{\|\boldsymbol{\sigma}\|_{\left(\frac{\epsilon}{1-\epsilon}\right)}} \tag{16}$$

where $\boldsymbol{\sigma}$ is the vector of singular values of $\mathbf{M}$.

Exploiting Structure For Better Feature Tracking

Optimization Strategy

We employed a 1^{st}-order decent method. The simplest such method is gradient decent. Unfortunately, there can be large differences in magnitude in the gradients of the fit terms (strong features can have much sharper gradients than weak features).

- If we could measure gradients exactly this would not be a problem.
- We must numerically approximate gradients (since our energy function is built on measured data). The approximation error on strong features can be larger than the entire gradient contribution from weak features.
- With pure gradient decent, the algorithm has a tendency to ignore weak features completely.

Exploiting Structure For Better Feature Tracking

Optimization Strategy

Our fix was to alternate between gradient decent steps and coordinate decent steps.

- Gradient decent steps ensure we continue moving in the direction that decreases the total energy the fastest.
- Coordinate decent steps ensure that week features do not get ignored.

Exploiting Structure For Better Feature Tracking

The Down Side

A 2^{nd}-order method (like in Lucas Kanade) has more expensive iterations, but requires fewer iterations to converge than our method. Ignoring the penalty term, the cost of our method is a small fixed multiple of the cost of Lucas Kanade. The penalty term was not significant in the problem sizes we worked with.

The Up Side

In exchange for the added computational cost, we get (much) more reliable tracking of low-quality features.

Experiments

Penalty Strength

By selecting the coefficient α, we can make the penalty strong or weak. A strong constraint will "demand" that the features move in a manner consistent with rigid motion. A week constraint will only "encourage" this behavior.

We observed (surprisingly) that the weak constraint was actually very useful.

Experiments

On The Week Constraint

With the week constraint:

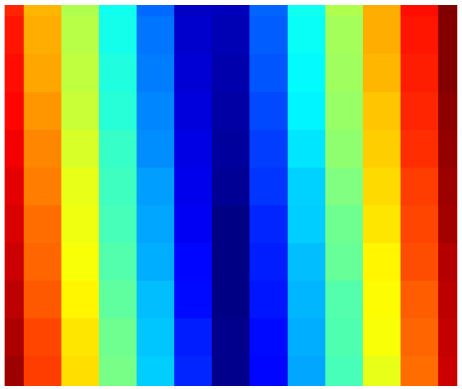
- If a feature has a very sharp single-feature energy, the penalty is effectively ignored in determining that features position.
- If a feature has a single-feature energy surface that is not sharp (in one or more directions) then the penalty becomes significant in those directions only.

Effectively, strong features are tracked normally, while weak features get assistance in ambiguous directions from the rest of the track set. This happens smoothly and automatically - there is not decision making in the algorithm.

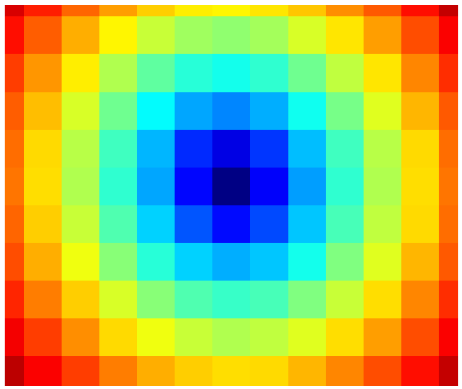
Experiments

Energy Surface For An Edge Feature

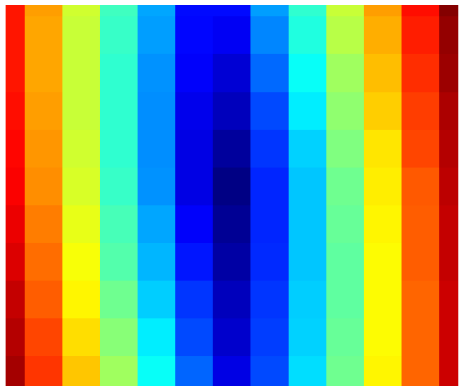
Here we show the energy surface projected onto the 2 dimensions which correspond to the position of a single edge feature. The fit term is ambiguous in the vertical direction.



(i) Fit Term



(j) Penalty Term



(k) Combined Energy

Experiments

Occlusion?

When a single feature becomes occluded, its template will not match the image at any feasible location. The fit term of the energy function becomes large and effectively random. Ideally, the penalty term would dominate the fit term and the feature would ideally be tracked according to the penalty term. The position of the feature would be inferred by the motion of the rest of the trackset. Some modifications are needed in order to achieve this:

- Detect Occlusion and stop updating template.
- Clip fit term values at a max level. This will prevent the randomness of the fit term from dominating the weak penalty term.

I have not tried this yet... it may not work.

Performance With Weak Constraint

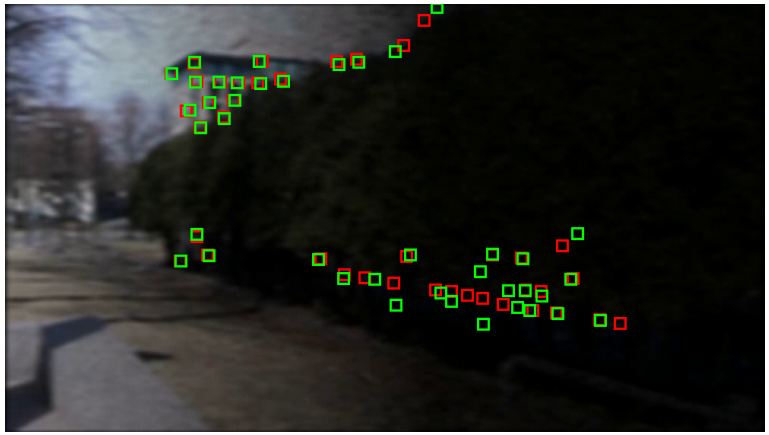
Table: Average track duration in frames (higher is better)

		Average Over All Files
Method	Lucas Kanade (OpenCV)	8.6
	Gradient Descent	36.1
	LDOF	22.3
	Rank-const: Emp Dim	67.0
	Rank-const: Nuc Norm	67.1
	Rank-const: Exp Fact	66.3

Experiments

Sample With Weak Constraint

Red boxes highlight groundtruth feature positions. Green boxes show tracker output. These images show tracker state 10 frames after initialization.



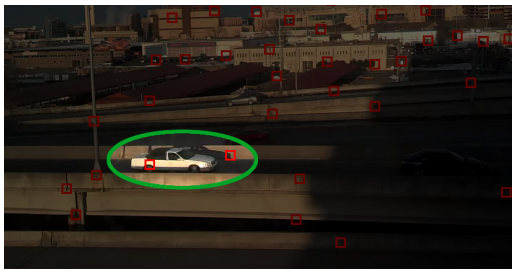
(l) Lucas Kanade

(m) Our Method - Weak

Experiments

On The Strong Constraint

Since the strong constraint forces rigidity, it makes it possible to track through occlusion. Regardless of whether or not a single feature is strong, if its motion is not consistent with the rest of the trackset, it is advantageous to adjust that features location according to the penalty term.



(n) Frame 1

(o) Frame 10 Strong

(p) Frame 10 Weak

Other Details Of Tracker

Other Details

- Tracker is multi-resolution (coarse-to-fine)
- State initialization is done via coarse global motion estimation.
- Feature templates are updated periodically and in batches.

Conclusions Of Rank-Regularized Tracking

Conclusions Of Rank-Regularized Tracking

- Exploiting structure observations can improve feature tracking performance.
- Our method exploits structure by adding a penalty term to the optimization objective.
- Encouraging conformance with a structure observation can work as well as, or almost as well as forcing conformance. It is also applicable in more problems since it only effects weak features.
- 1^{st}-order optimization (gradient/coordinate decent) gives more reliable results than 2^{nd}-order methods, and only incurs a modest increase in computational expense.

Thank You!