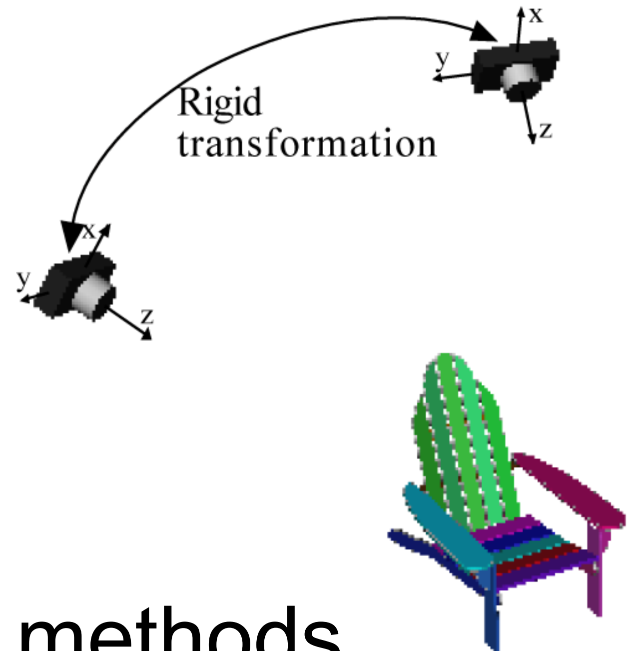
# Projective Surface Matching of Colored 3D Scans


| | |
|---|---|
| **Kari Pulli** | Nokia Research Center & MIT CSAIL |
| **Simo Piiroinen** | Nokia |
| **Tom Duchamp** | University of Washington (Math & CS) |
| **Werner Stuetzle** | University of Washington (Stat & CS) |

# Rigid registration

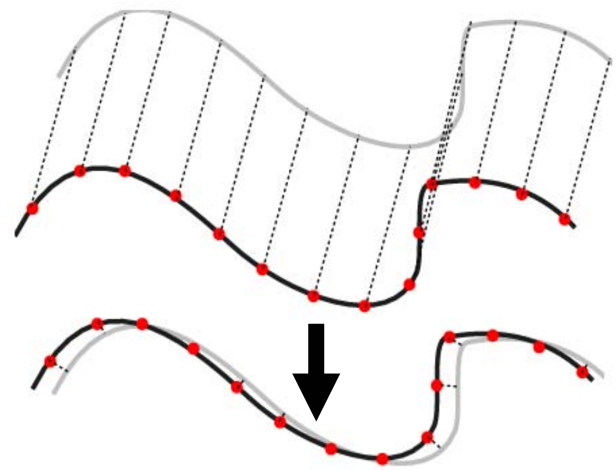
- Assume at least 2 scans
  - of the same object or scene
  - some overlap
- Find a rigid 3D transformation
  - so that samples corresponding to the same surface point are co-located




- Afterwards we can use various methods to reconstruct a single representation for
  - geometry
  - material properties

# ICP: match and align

- Correspondences
  - Heuristics to match scan points
    - take into account Euclidean distance, colors, local features, …
    - drop matches if over threshold, not coherent, at mesh boundary, …



- Alignment
  - find a rigid 3D motion that further reduces the distance between pairings
    - point-point: distance of paired points
    - point-plane: distance from a point to tangent plane
- Iterate until convergence

# Importance of color

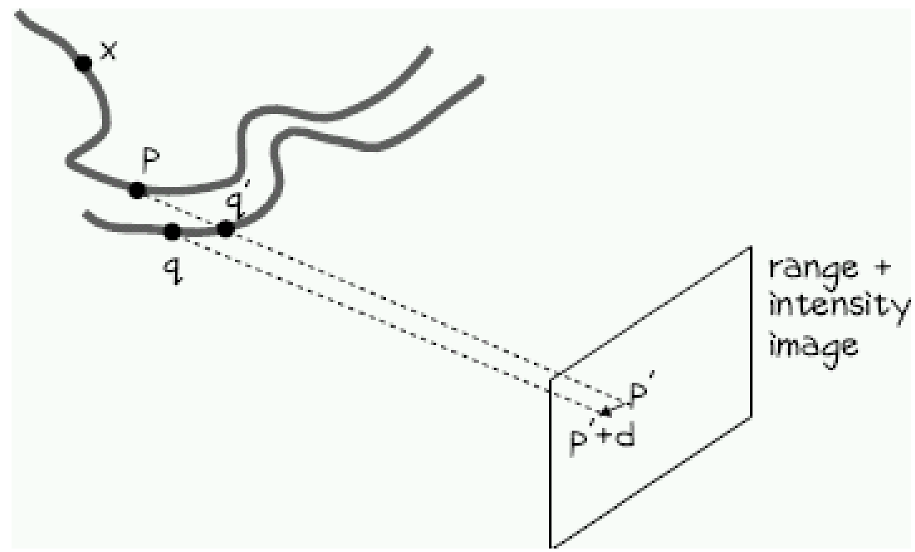
- Color can disambiguate
  - geometry of overlap may not constrain registration
    - sweep surfaces (planar in one direction), surfaces of revolution (cylindrical)
- If alignment only due to geometry
  - color reconstruction is likely to be blurred

# Projection for matching

- Projection used originally for acceleration
  - [Blais & Levine ’95]
  - no search of closest points
    - no building data structures and searching
    - usually O(log n) for each match
  - simply project one scan on the other scanner
    - requires knowledge of scanner calibration
    - usually O(1) for each match
  - then search for the best alignment

# Refine the projection match

- [Weik '97]
  - project point *p* and its intensity to *p'*
  - compare intensities of *p'* and *q'*
    - linearize: evaluate gradient on the image plane
    - take one step *d* to improve intensity match
  - move to *p'+d* on the plane, match *p* with with *q*
  - better matches lead to faster convergence



# Use more context

- [Pulli '97]
  - processing each pixel in isolation yields mismatches
    - esp. where color data is flat
    - where local illumination differs
  - 2D alignment
    - project other scan to camera image plane
    - 2D image alignment (planar projective mapping)
  - 3D alignment
    - match 3D points falling at the same pixel
    - 3D alignment using pairs

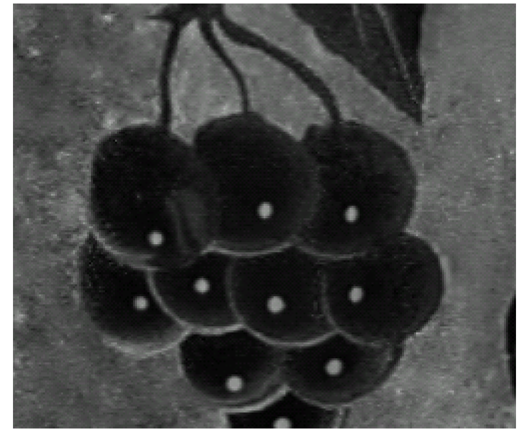
3d meshes

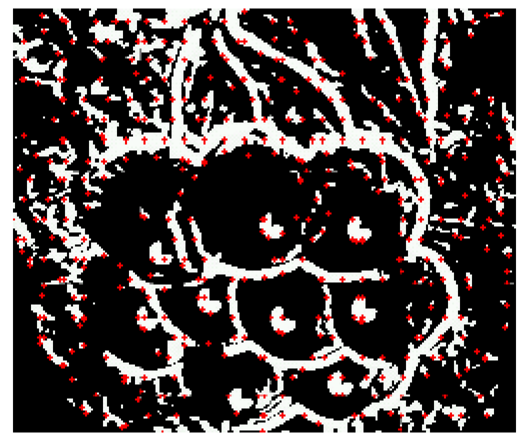
register 2D pictures, match corres-ponding 3D points

align paired points

# One more variation

- [Bernardini et al. ’01]
  - match only points with local intensity changes
    - avoids mismatches at flat colored areas
    - use cross-correlation
      - less affected by illumination changes
    - local search for local maximum
  - and align 3D points corresponding to matched 2D points





# What are we minimizing?

- Want to minimize the matching error of
  - geometry
  - colors
- But the errors are minimized separately
  - no single function to minimize
  - no guarantee of convergence
    - e.g., if there's any distortion of the geometry, the color and geometry errors are minimized at different poses
      - bounce back and forth
- We realized this in Dec '97
  - the new method mostly defined during '98
    - as well as the first implementation
    - but didn't have time to finish until last fall

# Define the error function

- With idealized data and pose
  - points in different scans
    - project to the same spot on any plane
    - have the same depth and color
    - do not project outside the silhouette of the other
- Recipe for the algorithm
  - project scans
  - within overlap
    - minimize color and range
  - outside
    - minimize distance to silhouette
  - component weights

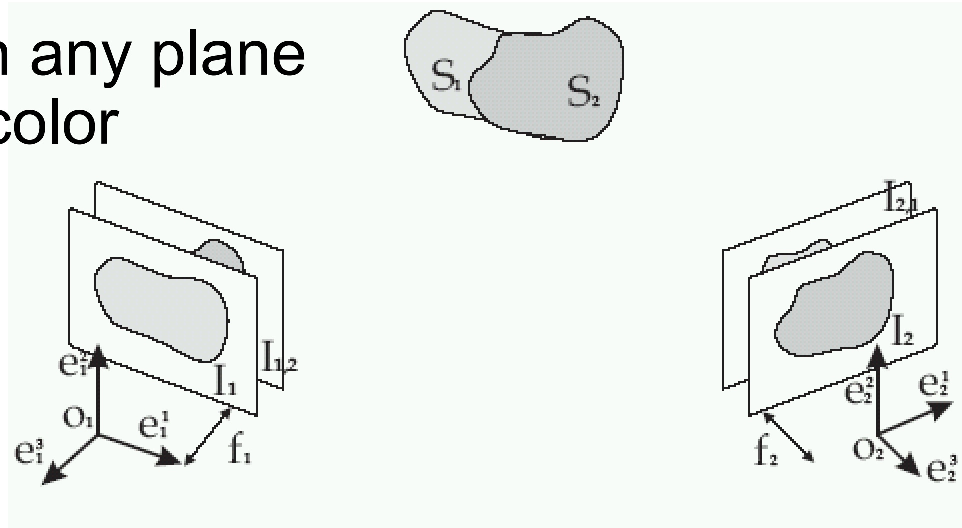
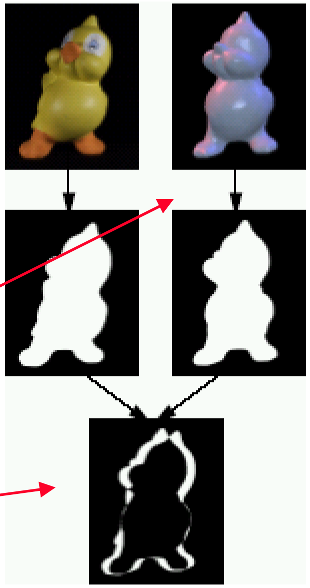


Figure 1. Two scans and their virtual cameras.

$$L(\mathbf{T}) = \int_{\Omega\cap\bar{\Omega}_T} \left((r-\bar{r}_T)^2 + \kappa_1\|\mathbf{c}-\bar{\mathbf{c}}_T\|^2\right) + \kappa_2 \int_{\bar{\Omega}_T} d^2(\mathbf{u},\Omega).$$

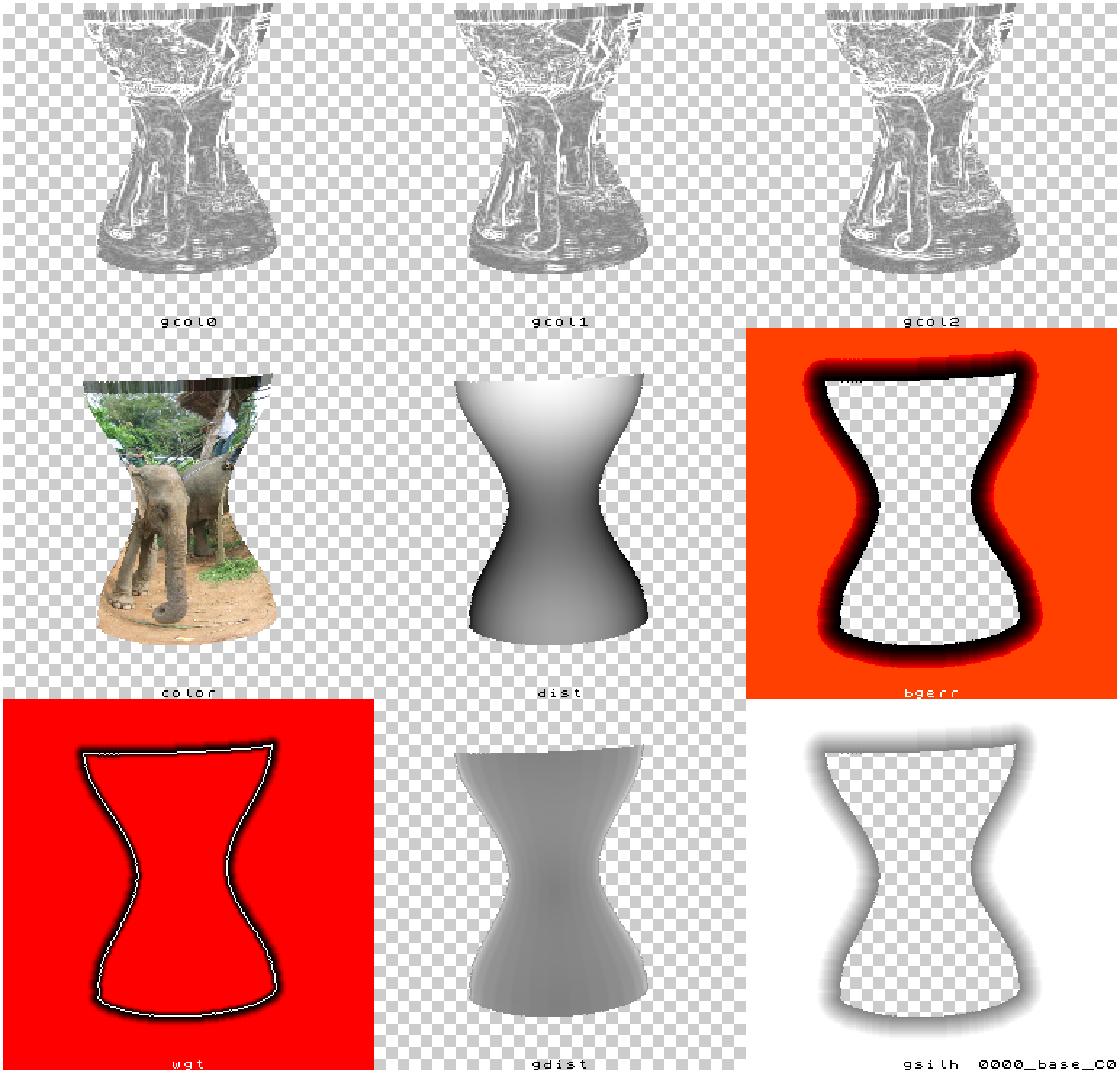
# Silhouettes

- Assumptions with using silhouettes
  - see the full object
  - no missing data
  - can separate background from foreground
- But when you can use them, they provide strong constraints
  - [Lensch et al. '01]
  - registered a surface model to an image
    - extract image foreground
    - render the model white-on-black
    - evaluate with graphics HW (XOR)
    - minimize with Simplex



# Implementation

- Create textured meshes
  - associate a virtual pinhole camera with each scan
    - simple camera calibration to estimate camera location
- Render from camera viewpoint
  - only the “other” view changes
  - “this” view needs to be rendered just once
    - the virtual pinhole camera anchored with the view
- For each pixel
  - evaluate error (color, range, silhouette)
  - estimate image gradient (of each component)
  - analyze pixel flow as a function of 3D transformation T
    - propagate 2D error + gradient to 3D motion
- Minimize with Levenberg-Marquardt
  - just need error and its gradient w.r.t. T




Example:

color gradients (RGB)

color

range

silhouette penalty

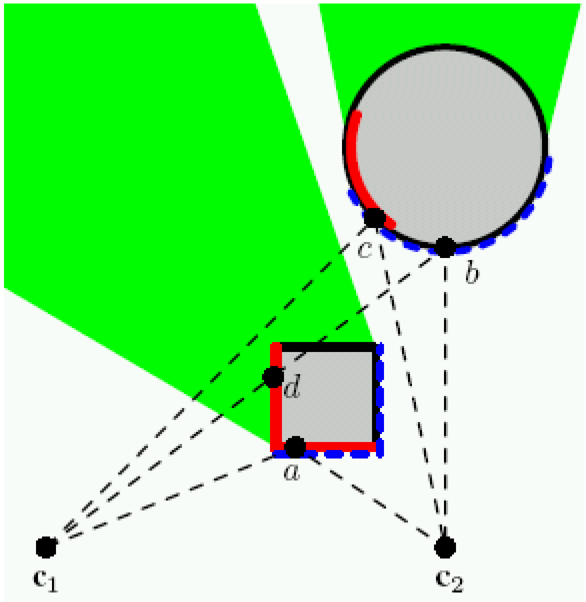
downweight at silhouette

gradients of distance, silhouette

# Color errors: start & end

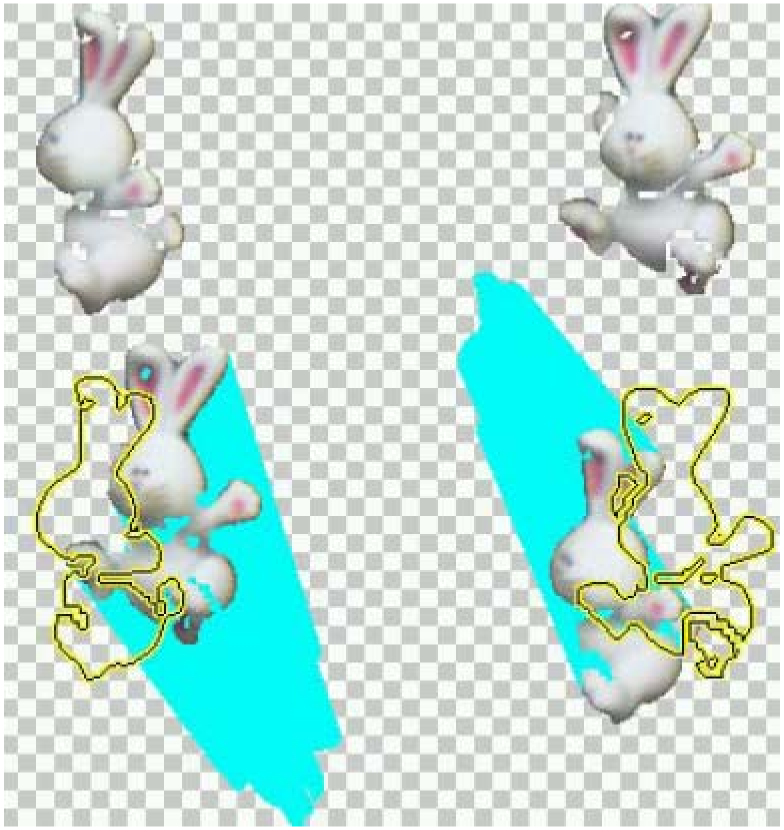
# Prevent false matches

- Surface seen in one camera (b)
  - may remain occluded in another
- Threshold approach
  - pair too far behind: mismatch (b,d)
    - [Weik '97, Pulli '97, Bernardini '01, …]
    - need a larger threshold at start than in the end
- Extruded silhouettes
  - like shadow volumes
  - disallow matches covered by extended silhouettes (d,b)
  - conservative: disallows also (c)
  - but no need to choose a threshold



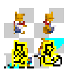
# False match prevention

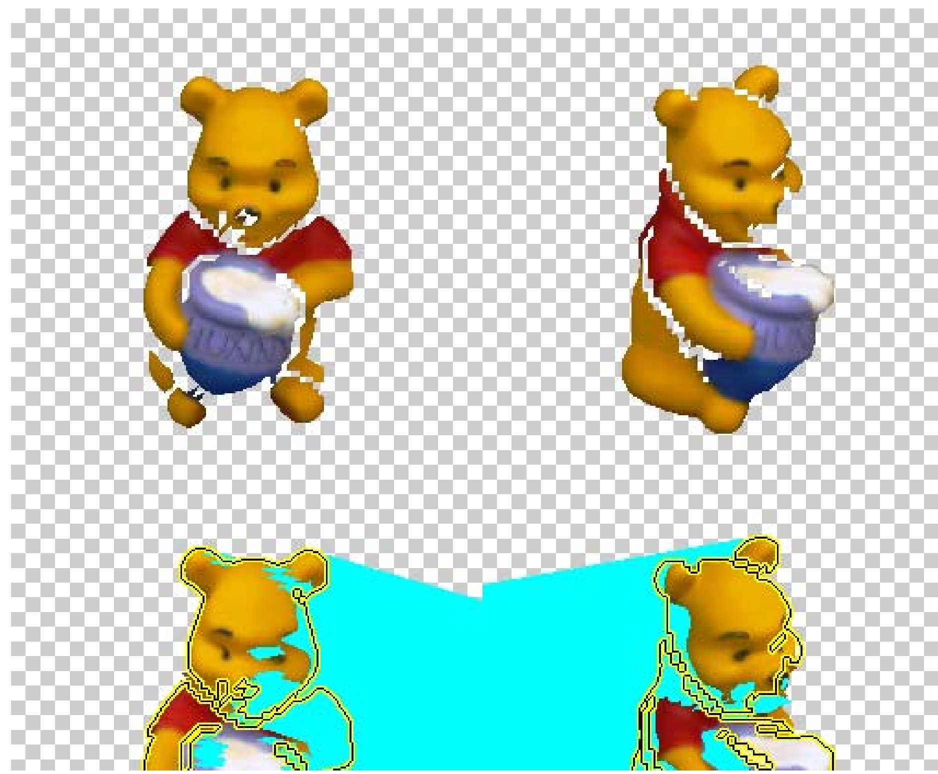
- Implemented as
  - silhouette edges extruded as cyan polygons
  - skip any cyan pixels



# Hierarchical minimization

- Faster
- More robust
  - error function is smoothed
  - fewer local minima





# Summary

- Define good color & range registration
  - project to image plane
  - match projections
- Minimize the defined registration error
  - Levenberg-Marquardt
    - just for the small improvement around the current pose estimate
  - error gradients
    - direct numerical evaluation (sanity check)
    - lift 2D errors and gradients to 3D
    - not much difference in performance, method reported on the paper requires fewer evaluations => faster

# Equations: how to read and use

- Levenberg-Marquardt needs
  - evaluate error
    - easy: just the different at a pixel (or dist. to silhouette)
  - Jacobian of error function w.r.t. 3D xform params
    - a matrix with 6 columns (d = [a b g x y z] 3 rot, 3 trans)
    - a bit different for different components (depth, color, silh.)

- Let’s just analyze one color component
  - how much does the error change as a function of a rigid 3D motion?
    - gradient on the image plane (estimate numerically)
      times
      image flow of a surface point due to 3D motion

$$\nabla_{\mathbf{d}} \bar{\mathbf{c}}_{\mathbf{d}} = -\nabla_{\mathbf{u}} \bar{\mathbf{c}} \cdot \nabla_{\mathbf{d}} \psi_{\mathbf{d}}$$

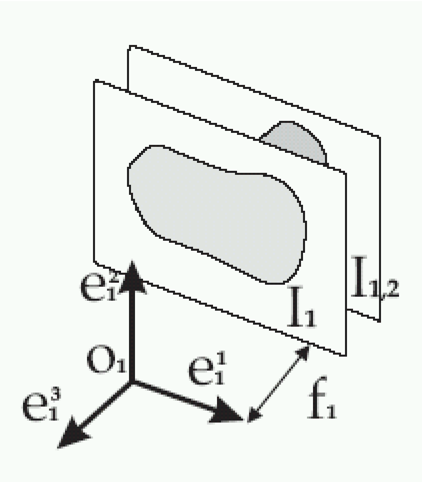
2-vector
row/col gradient

2x6 - matrix
image flow due to xform d

# Projection 2D-3D equations

- Project image point **u** (at distance r) using camera at **o** with focal length *f*

$$\mathbf{P}(\mathbf{u}) = \mathbf{o} + \frac{r(\mathbf{u})}{f}(u\,\mathbf{e}^1 + v\,\mathbf{e}^2 + f\mathbf{e}^3)$$




and translate it around **m** by rotation **R** and translation **t**

$$\mathbf{T}(\mathbf{x}) = \mathbf{R}\cdot(\mathbf{x} - \mathbf{m}) + \mathbf{m} + \mathbf{t}$$

# Projection 3D-2D equations

- Project 3D point **x** to image point *(u,v)*

$$\mathbf{\Pi}(\mathbf{x}) = f\left(\frac{(\mathbf{x}-\mathbf{o})\cdot\mathbf{e}_1}{(\mathbf{x}-\mathbf{o})\cdot\mathbf{e}_3}, \frac{(\mathbf{x}-\mathbf{o})\cdot\mathbf{e}_2}{(\mathbf{x}-\mathbf{o})\cdot\mathbf{e}_3}\right) = (u,v)$$

- Motion of point's projection when it moves in 3D

$$\nabla_{\mathbf{x}}\mathbf{\Pi} = \begin{bmatrix}\frac{\partial u}{\partial \mathbf{x}} & \frac{\partial v}{\partial \mathbf{x}}\end{bmatrix}^T$$

  - its *u* component (*v* is similar)

$$\begin{aligned}\frac{\partial u}{\partial \mathbf{x}} &= f\frac{\mathbf{e}_1}{(\mathbf{x}-\mathbf{o})\cdot\mathbf{e}_3} - f\frac{(\mathbf{x}-\mathbf{o})\cdot\mathbf{e}_1}{(\mathbf{x}-\mathbf{o})\cdot\mathbf{e}_3}\frac{\mathbf{e}_3}{(\mathbf{x}-\mathbf{o})\cdot\mathbf{e}_3} \\ &= \frac{1}{(\mathbf{x}-\mathbf{o})\cdot\mathbf{e}_3}(f\mathbf{e}_1 - u\mathbf{e}_3) \\ &= \frac{1}{\mathbf{r}(u,v)}(f\mathbf{e}_1 - u\mathbf{e}_3).\end{aligned}$$

# Combine: Image flow

- How does a pixel move when a mesh (camera) is moved by a small transformation d?
  - project pixel from image plane to 3D
  - change in 3D by a small transformation
  - that transformation projected to image plane

$$\nabla_{\mathbf{d}}\psi = \nabla_{\mathbf{x}}\Pi \cdot \nabla_{\mathbf{d}}(\mathbf{T} \circ \bar{\mathbf{P}})$$

$$\nabla_{\mathbf{x}}\Pi = \frac{1}{r(\mathbf{u})}\begin{bmatrix} f\mathbf{e}^1 - u\mathbf{e}^3 \\ f\mathbf{e}^2 - v\mathbf{e}^3 \end{bmatrix}$$

2x3 - matrix
from 3D to 2D

$$\nabla_{\mathbf{d}}\mathbf{T}(\mathbf{x}) = \begin{bmatrix} 0 & z - m_z & -(y - m_y) & 1 & 0 & 0 \\ -(z - m_z) & 0 & (x - m_x) & 0 & 1 & 0 \\ (y - m_y) & -(x - m_x) & 0 & 0 & 0 & 1 \end{bmatrix}$$

3x6 - matrix
rotate 3D point x around “mass center” m, translate

# Get the whole Jacobian

- Jacobian for a color component (e.g., red)

$$\nabla_{\mathbf{d}} \bar{\mathbf{c}}_{\mathbf{d}} = -\nabla_{\mathbf{u}} \bar{\mathbf{c}} \cdot \frac{1}{r(\mathbf{u})} \begin{bmatrix} f\mathbf{e}^1 - u\mathbf{e}^3 \\ f\mathbf{e}^2 - v\mathbf{e}^3 \end{bmatrix} \begin{bmatrix} 0 & z - m_z & -(y - m_y) & 1 & 0 & 0 \\ -(z - m_z) & 0 & (x - m_x) & 0 & 1 & 0 \\ (y - m_y) & -(x - m_x) & 0 & 0 & 0 & 1 \end{bmatrix}$$

1x6
row vector

- Need to modify for range and normal
  - since the transformation changes those as well
    - not just where they project to

$$\nabla_{\mathbf{d}} \bar{\mathbf{c}}_{\mathbf{d}} = -\nabla_{\mathbf{u}} \bar{\mathbf{c}} \cdot \nabla_{\mathbf{d}} \psi_{\mathbf{d}}, \quad (12)$$

$$\nabla_{\mathbf{d}} \bar{r}_{\mathbf{d}} = -\nabla_{\mathbf{u}} \bar{r} \cdot \nabla_{\mathbf{d}} \psi_{\mathbf{d}} + \mathbf{e}^3 \cdot \left(\nabla_{\mathbf{d}} \mathbf{T} \circ \bar{\mathbf{P}}\right), \quad (13)$$

$$\nabla_{\mathbf{d}} \bar{\mathbf{n}}_{\mathbf{d}} = -\nabla_{\mathbf{u}} \bar{\mathbf{n}} \cdot \nabla_{\mathbf{d}} \psi_{\mathbf{d}} + \nabla_{\mathbf{d}} \mathbf{R} \cdot \bar{\mathbf{n}}. \quad (14)$$

# L-M for incremental changes

- Use L-M to calculate small xform **d**
  - those equations were for projecting scan A on B
    - we want to use all information, so project A on B as well as B on A
    - when projecting B on A, just flip the sign of the gradient
  - append transformation **d** to the current estimate of the registration pose
  - draw a new image
  - repeat until convergence