GEOMETRIC HASHING: AN OVERVIEW

Haim J. Wolfson
Isidore Rigiutsos

What is geometric hashing good for?

- Identification of test objects from a set of know objects using 2D or 3D features
- Object models can be made invariant to translation, rotation, scaling, affine and projective transformations! NEAT!
- Uses an arbitrary feature detector. Works with SIFT etc.
- Quick model identification.

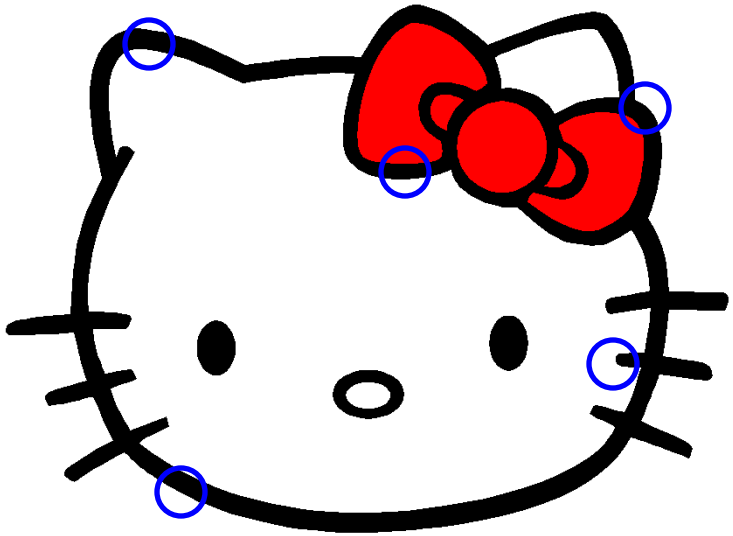
Procedure for Model Building

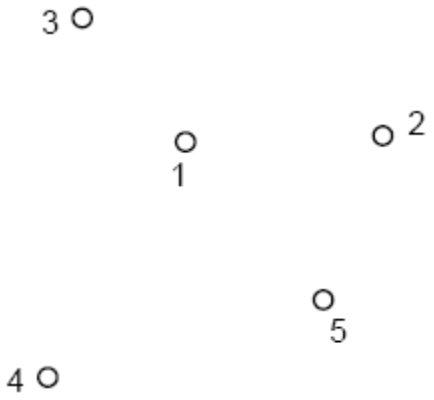
- Building Object Models
  - Extract Features
  - Define a basis with respect to two features
  - Parameterize features with respect to basis.
  - Add the parameterized features to a 2D hash table
  - Repeat with a new basis until every basis is added
  - Do some hash table tweaking to improve worst case performance.
  - Do further hash table tweaking to improve performance in the case of noise.

Procedure for Model Detection.

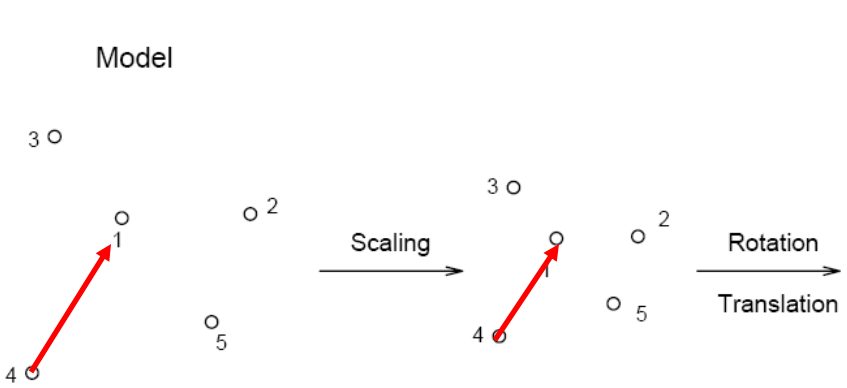
- Model Detection
  - Extract features from the image and parameterize them with respect to a basis.
  - Do a hash table lookup for each feature. Hash table entries vote for the model.
  - For each candidate model complete the necessary transformation, and test against the input.
  - If the match isn't good pick a new basis and restart the procedure.

Feature Extraction

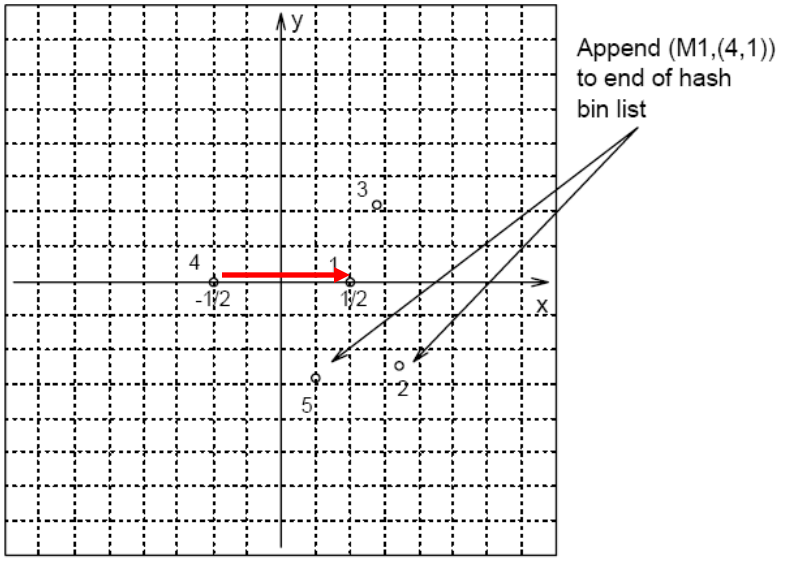




Scale and Create a Basis

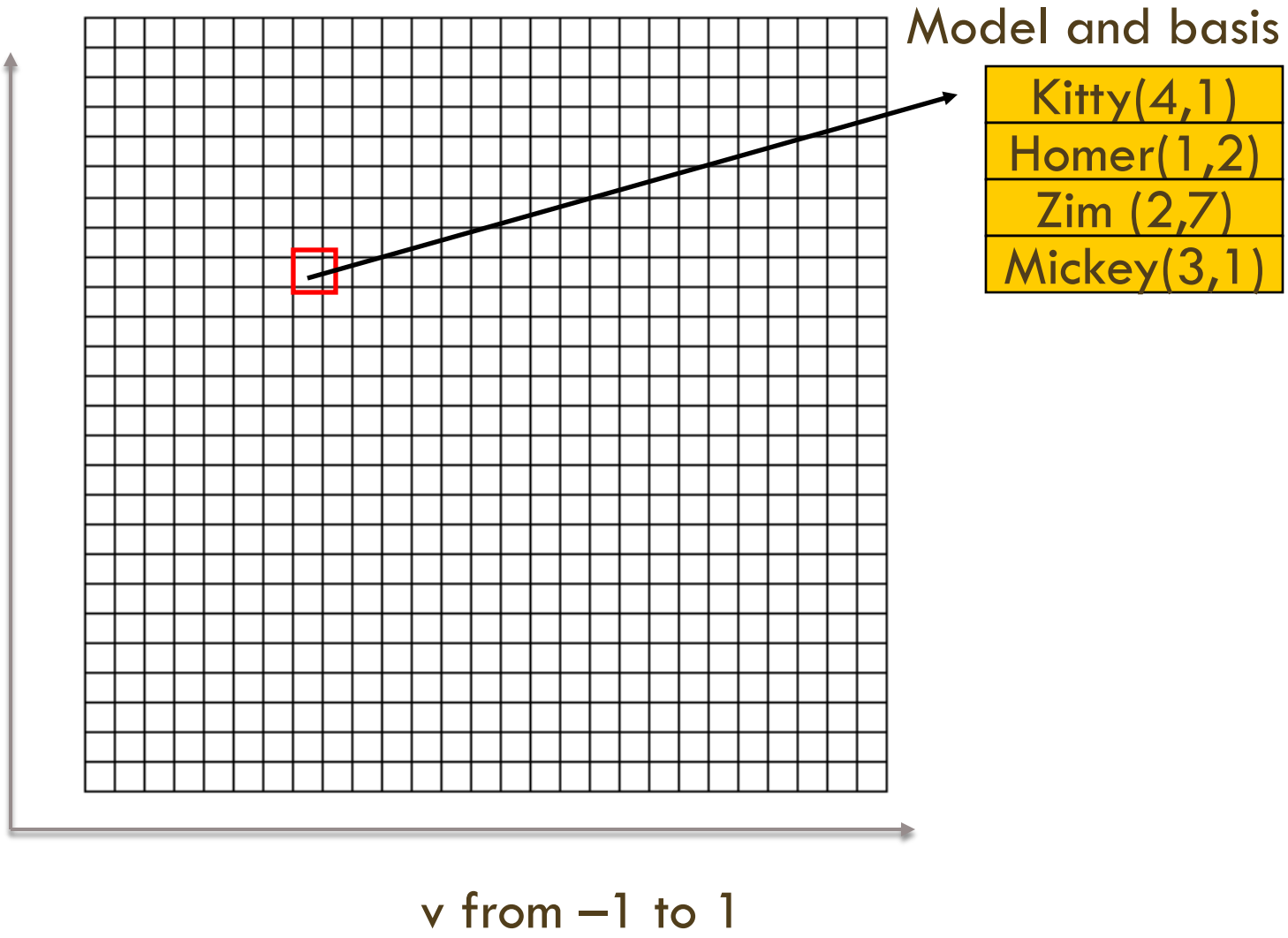


Scale to unit length
(scale invariance)

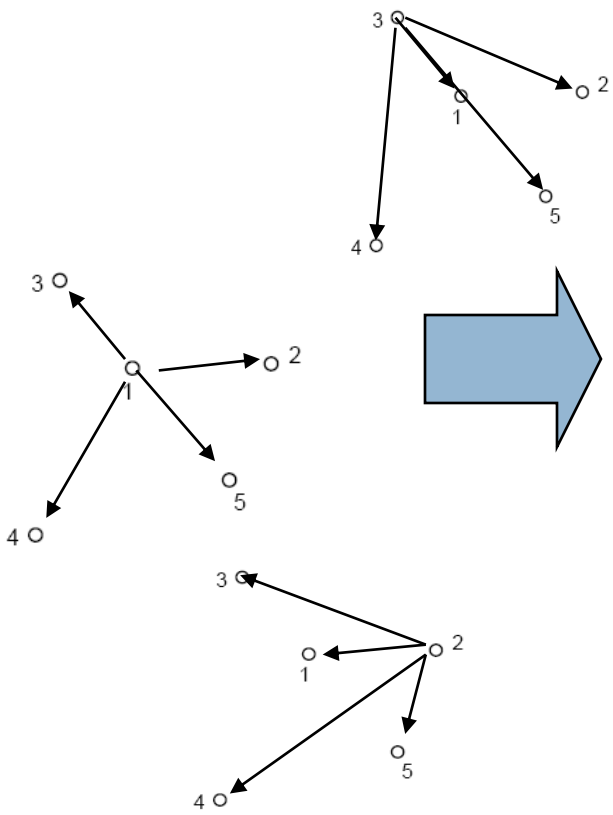


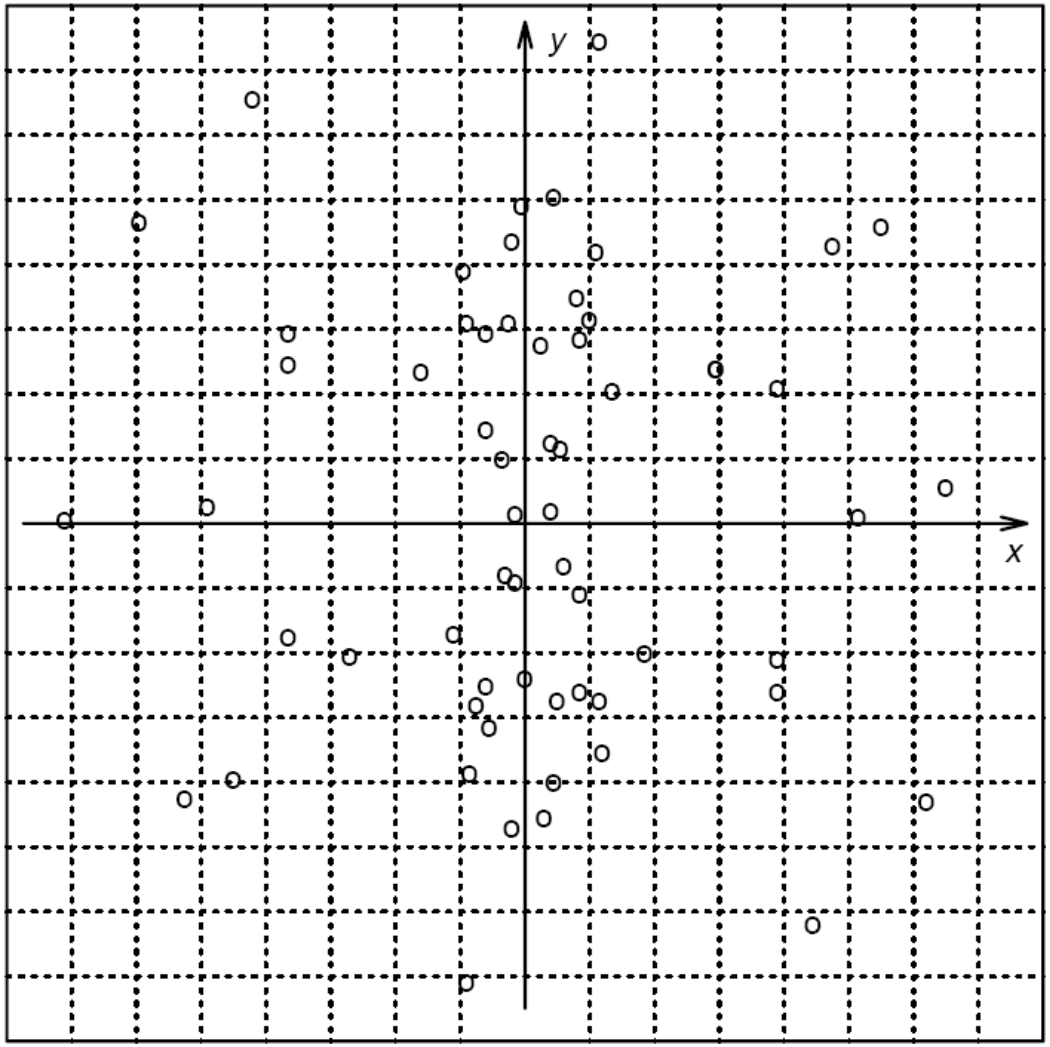
Rotate to +x axis
(rotation invariance)

We also create a 2D hash table



Now repeat for each basis



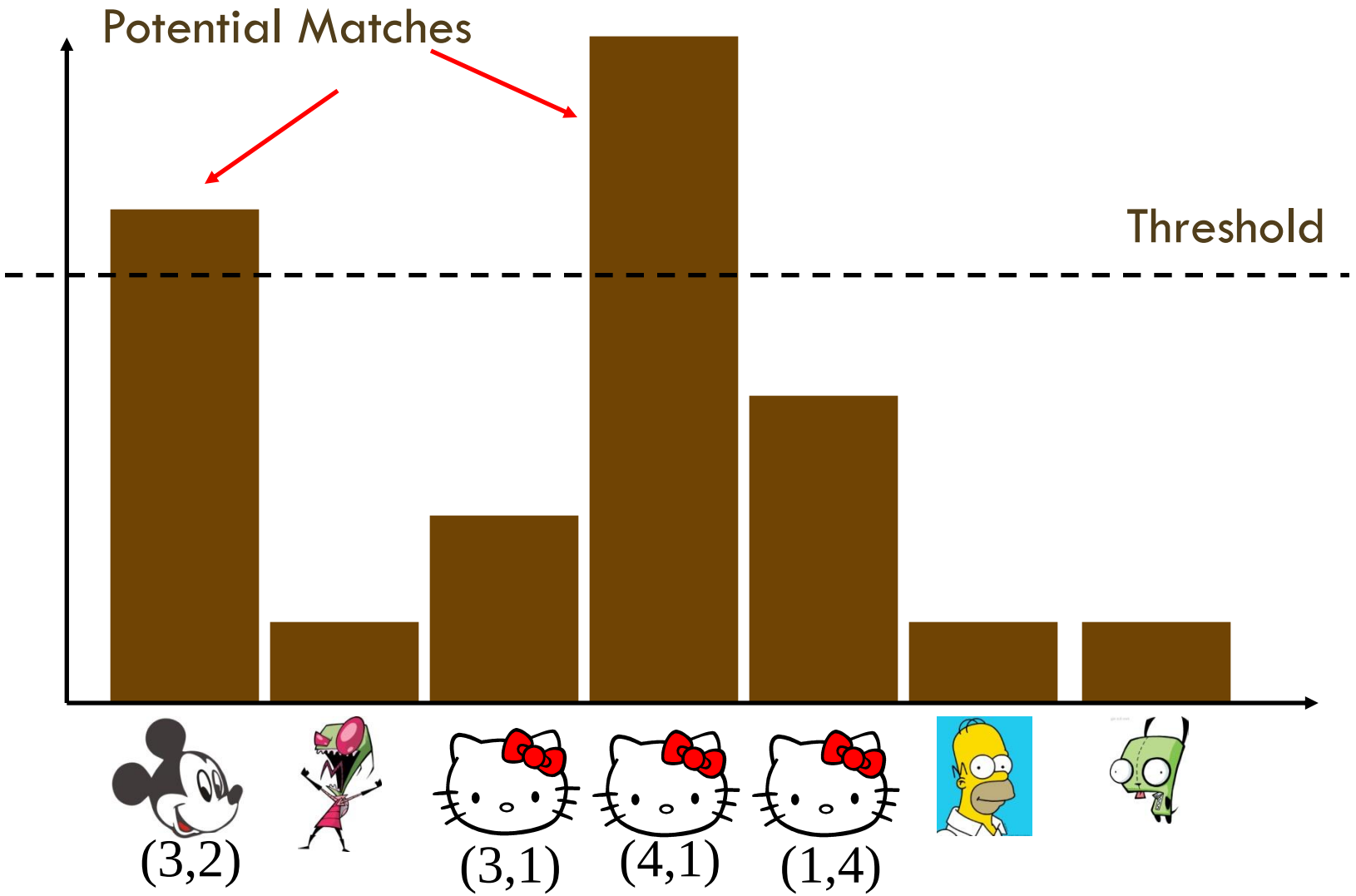


Each arrow is a basis

Recognition of Objects

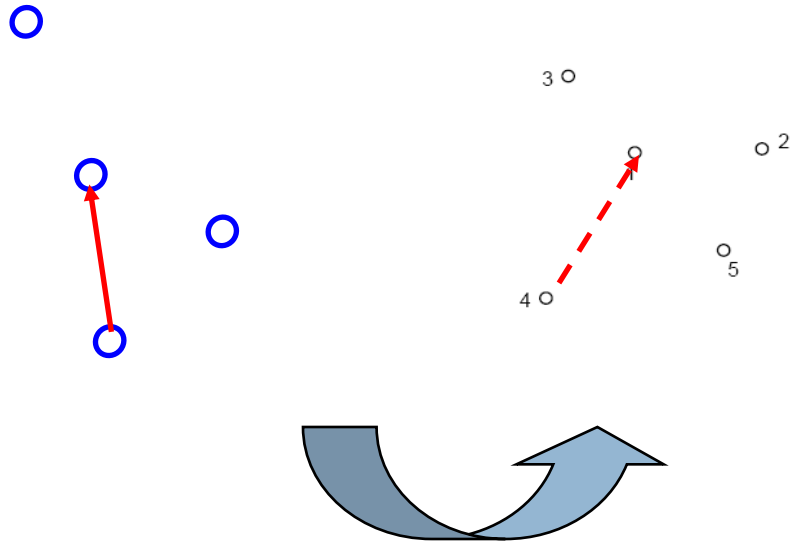
- Find Features
- Select Basis
- Parameterize Features
- Do hash table lookup
- Histogram number of matches per model

Recognition of Objects



Recognition of Objects

Verification



Least Square Transform Recovery
(find the rotation, and translation of the basis)

Geometric hashing has a number of nice properties in 2D

Transformation Recovered:	**What is needed:**
Translation	One point centered at origin
Translation & Rotation	Two points
Translation, Rotation, Scaling	Two points and scale to length of one.
Affine Transformation	Three Points
Projective Transformation	Four Points

Geometric Hashing also works in 3D

Transformation Recovered:	**What is needed:**
Translation	One point centered at origin
Translation & Rotation	Three points –or- Two non co-linear vectors
Translation, Rotation, Scaling	A line and a point or Two non-collinear non-coplanar lines

Run time of Geometric Hashing

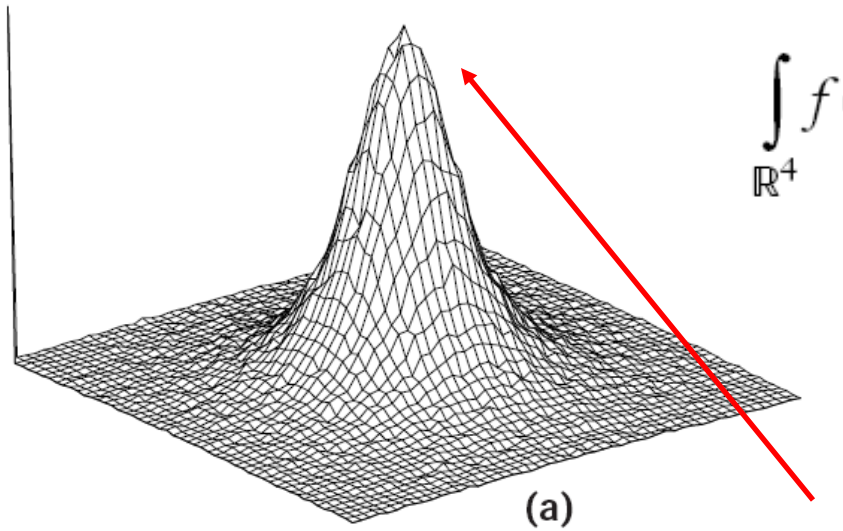
M = number of models
n = number of features per model.
c = the number of elements in a basis
H = hash table access time.
S = number of features in a test image

Computing the hash table runs at $O(Mn^{c+1})$
Looking up a model in the hash table takes $O(HS^{c+1})$

The run time is not great, but tolerable when H is small as H can vary
i.e. $O(1) \leq H \leq O(Mn^{c+1})$

What if we end up with a really bad hash table?

pdf of has entry distribution f(u,v)



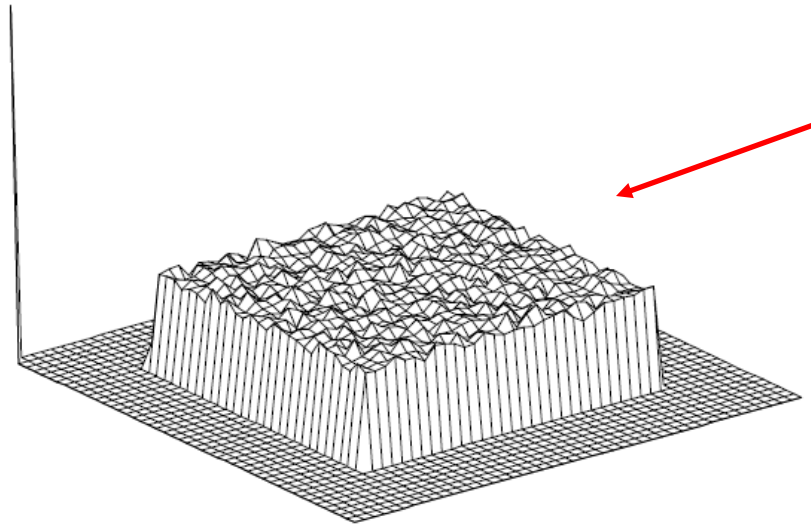
$$\int_{\mathbb{R}^4} f\big(x(u,v), y(u,v)\big) f\big(x_{\mu_1}, y_{\mu_1}\big) f\big(x_{\mu_2}, y_{\mu_2}\big)$$

$$|\mathcal{J}|^{-1}\, dx_{\mu_1} dx_{\mu_2} dy_{\mu_1} dy_{\mu_2},$$

Really bad hash table. Worst case performance is horrible!

We can calculate (f(u,v)) or model (f*(u,v)) the probability distribution function of the entries of the hash table.

Rehashing a Table



Nice uniform hash table

Similarity transform with Gaussian noise has this pdf

$$f(u,v) = \frac{12}{\pi} \cdot \frac{1}{(4(u^2+v^2)+3)^2} \cdot$$

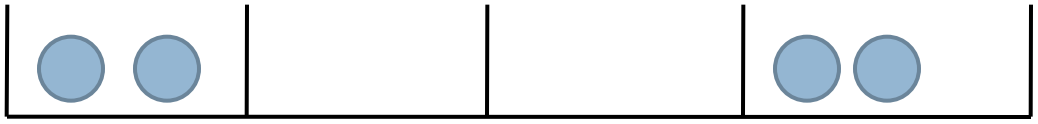
$$h(u,v) = \left(1 - \frac{3}{4(u^2+v^2)+3}, \operatorname{atan}2(v,u)\right)$$

Find a function h(u,v) that can apply to each hash entry to get a nice uniform distribution. This is to say h(u,v) makes f(u,v) or f*(u,v) into an uniform distribution.

Intuition behind rehashing

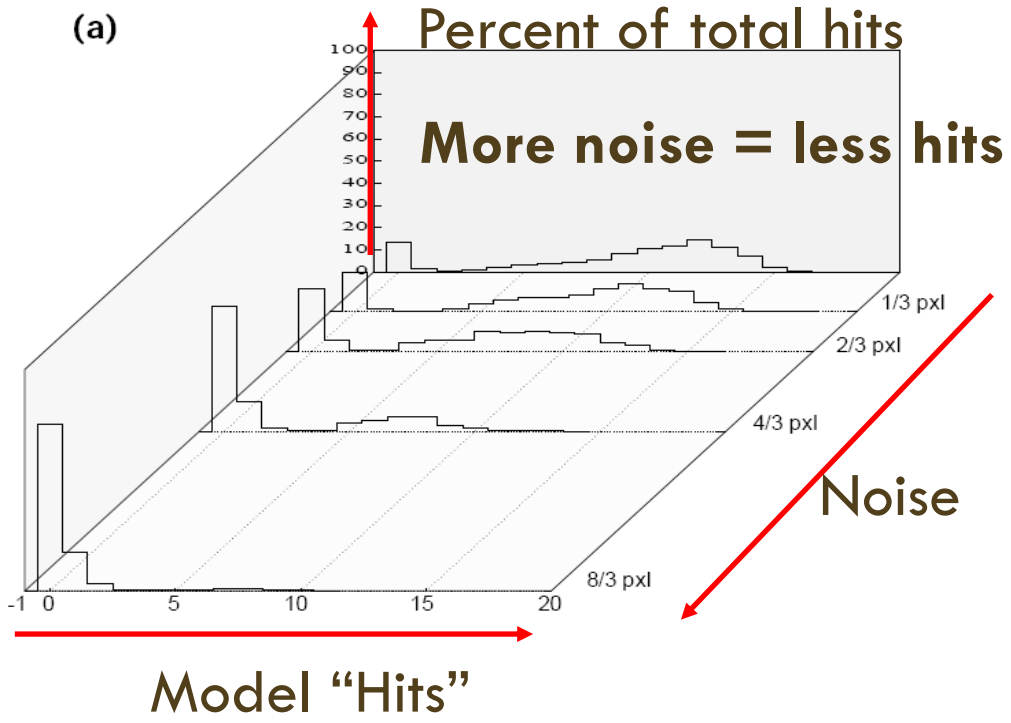
Consider a 1D hash table.

We determine f(u) for this table



And find h(u) that does this:

Modeling and Dealing with Noise



Some noise tolerance is built into the hash table by quantization. Bigger bin sizes result in less noise but lower fidelity. One approach is to weight bins in a neighborhood or weighting the votes of each bin

More about image noise

1. The larger the separation of basis points the smaller the effect of noise spread in the hash table. (Noise is proportionally smaller)

2. The closer a point is to the center of the basis the smaller the spread of noise in the hash table.

3. Short hash table entries contain more information than long ones.

Bayesian Formulation for Weighted Bin Distribution

We can construct a Bayesian formulation of geometric hashing

- x y = value of basis transformation
- u v = hash table bin value
- p_μ p_v = trial basis
- σ = error distribution
- S = number of points in scene

$$\tau = (4(x^2 + y^2) + 3)\,\sigma^2$$

z is the neighborhood function for weighting hash table entries

$$z = \log\left(1 + \frac{(4(u^2+v^2)+3)^2 \left\| p_\mu - p_v \right\|^2}{12S\tau} \cdot \exp\left(\frac{-\left\|(u,v)-(x,y)\right\|^2}{\tau / \left\| p_\mu - p_v \right\|^2}\right)\right) \qquad (5)$$