Dependable Systems for Sentient Computing

Andrew Rice

Problem

Sentient Computing is **failure sensitive**

Sentient Computing is **failure prone**

Dependability

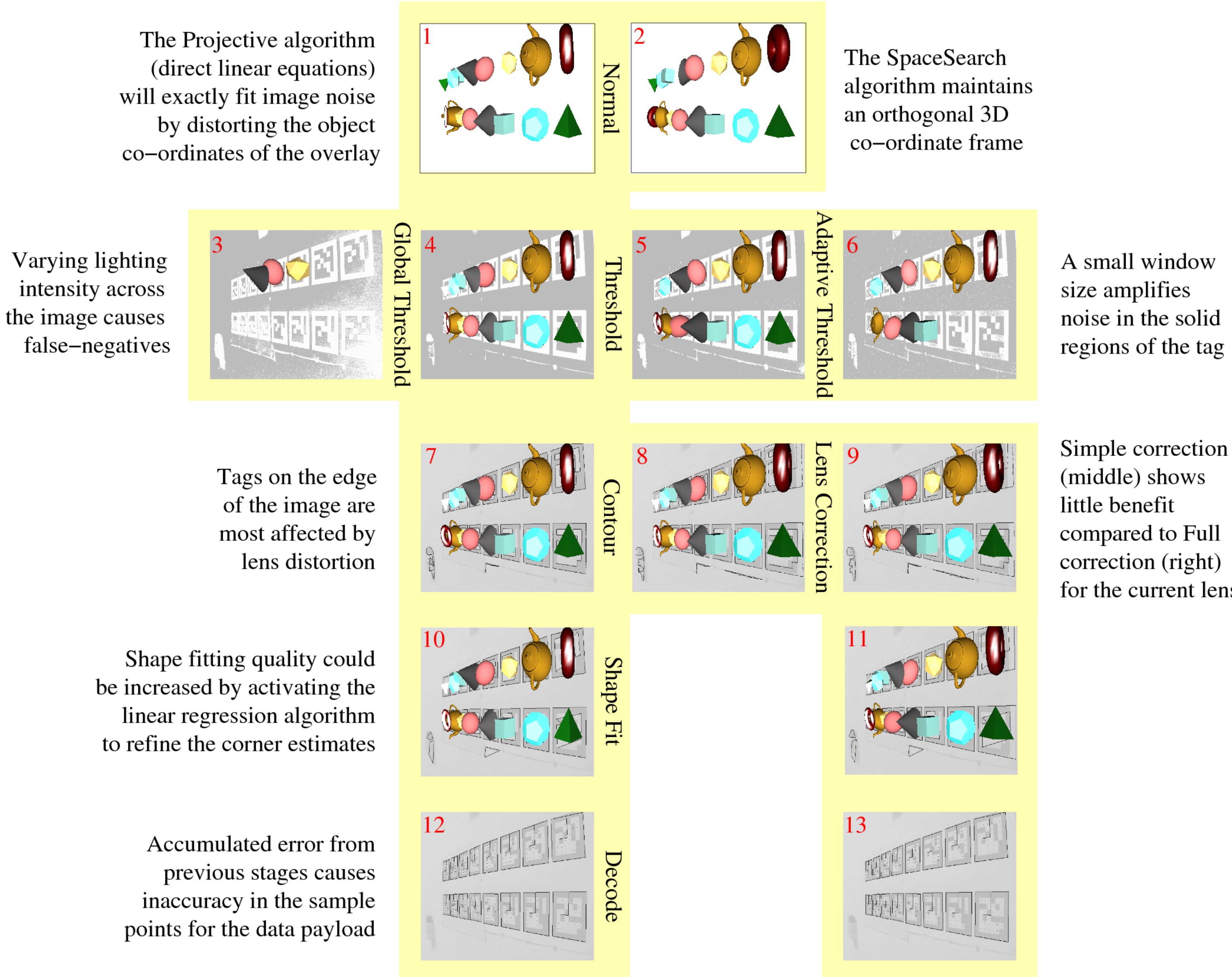
A dependable system can provide, at any time, a specification of current system performance and status

Is it possible to build a dependable location system?

Cantag

Marker Tracking Framework
for investigating dependability

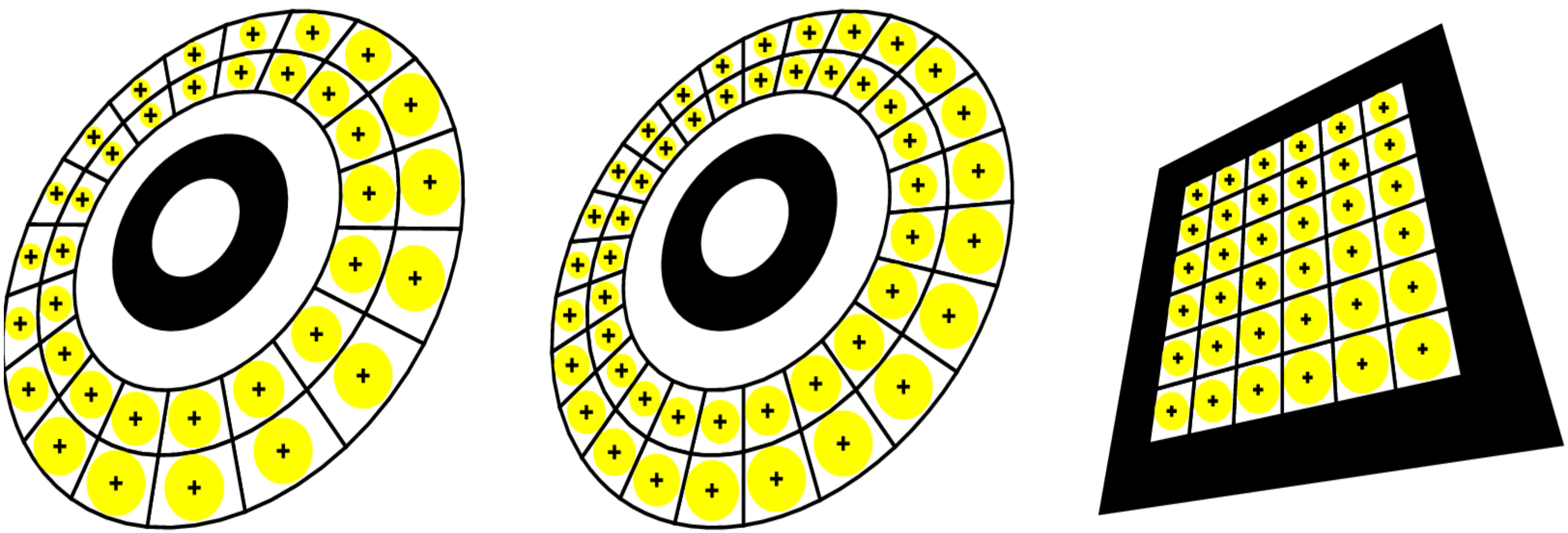
- Reliable Implementation
- Designed for Instrumentation
- Reconfigurable and flexible


The Projective algorithm (direct linear equations) will exactly fit image noise by distorting the object co−ordinates of the overlay
1
2
Normal
The SpaceSearch algorithm maintains an orthogonal 3D co−ordinate frame
Varying lighting intensity across the image causes false−negatives
3
Global Threshold
4
Threshold
5
Adaptive Threshold
6
A small window size amplifies noise in the solid regions of the tag
Tags on the edge of the image are most affected by lens distortion
7
Contour
8
Lens Correction
9
Simple correction (middle) shows little benefit compared to Full correction (right) for the current lens
Shape fitting quality could be increased by activating the linear regression algorithm to refine the corner estimates
10
Shape Fit
11
Accumulated error from previous stages causes inaccuracy in the sample points for the data payload
12
Decode
13

Algorithmic Dependability

How does my system behave in theory?

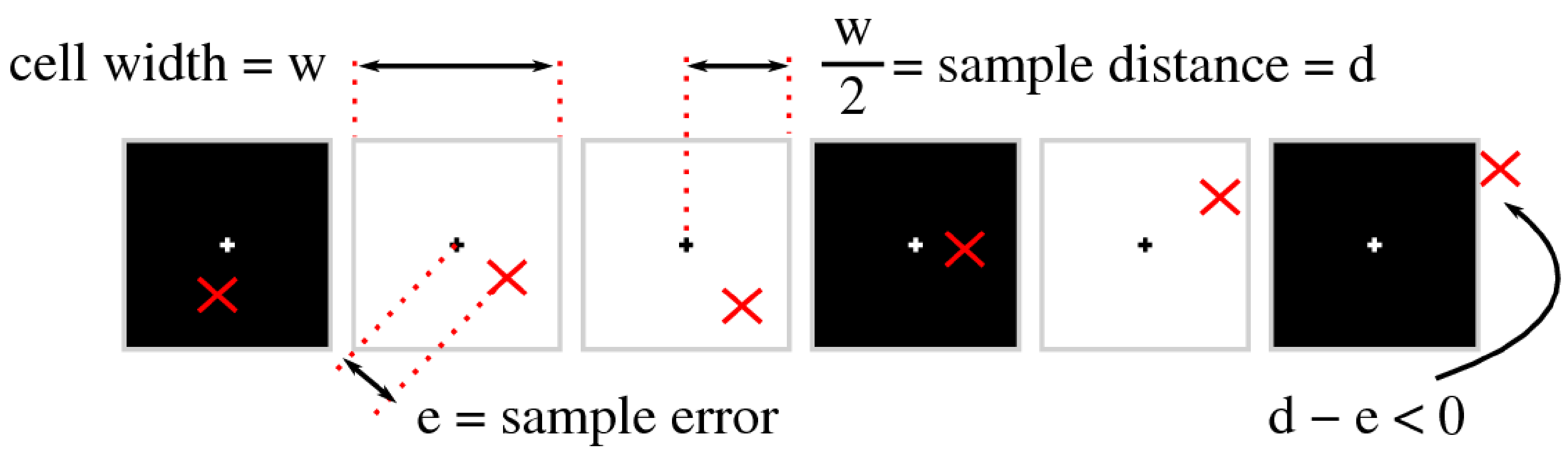
Sample Distance



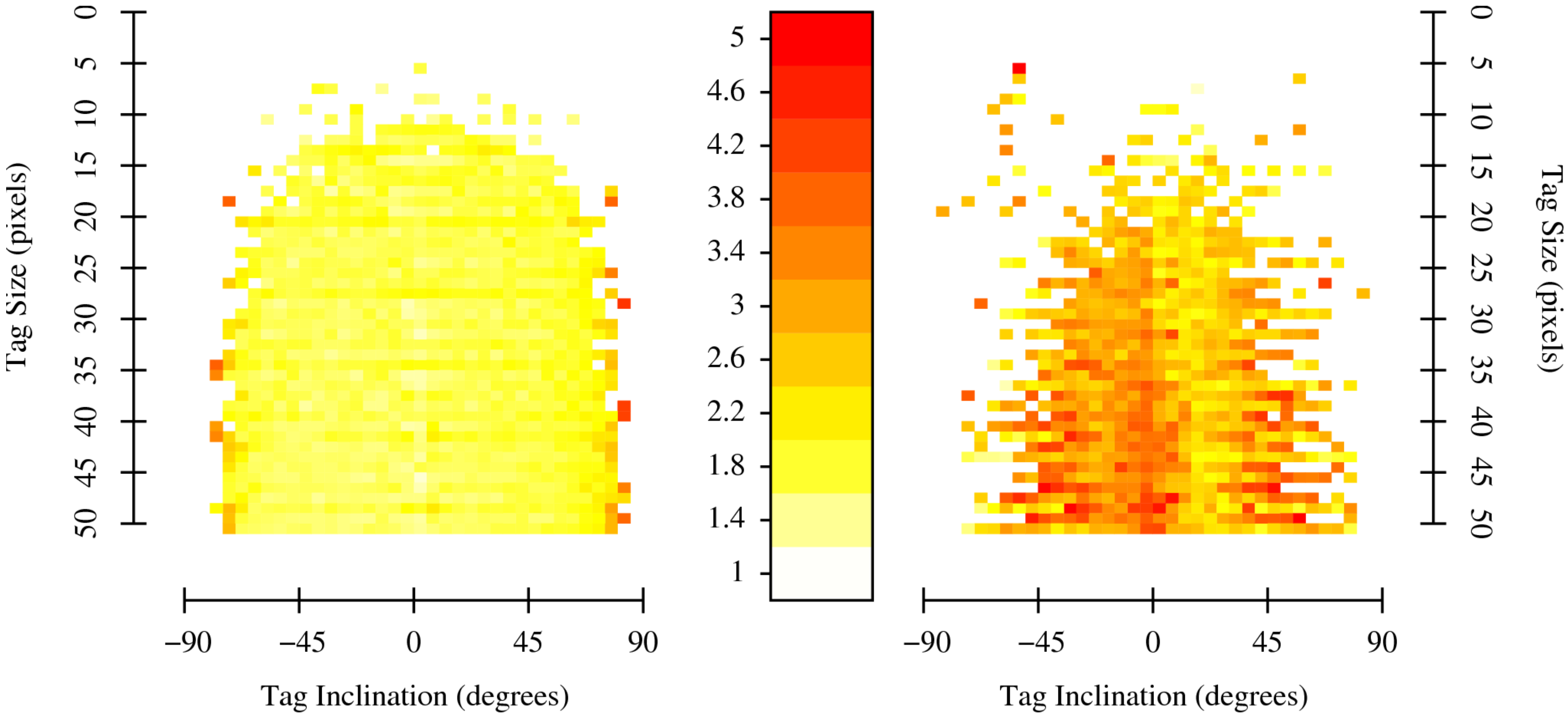
Abstract, information-theoretic limit

Optimising tag design requires maximising the sample distance

Sample Error



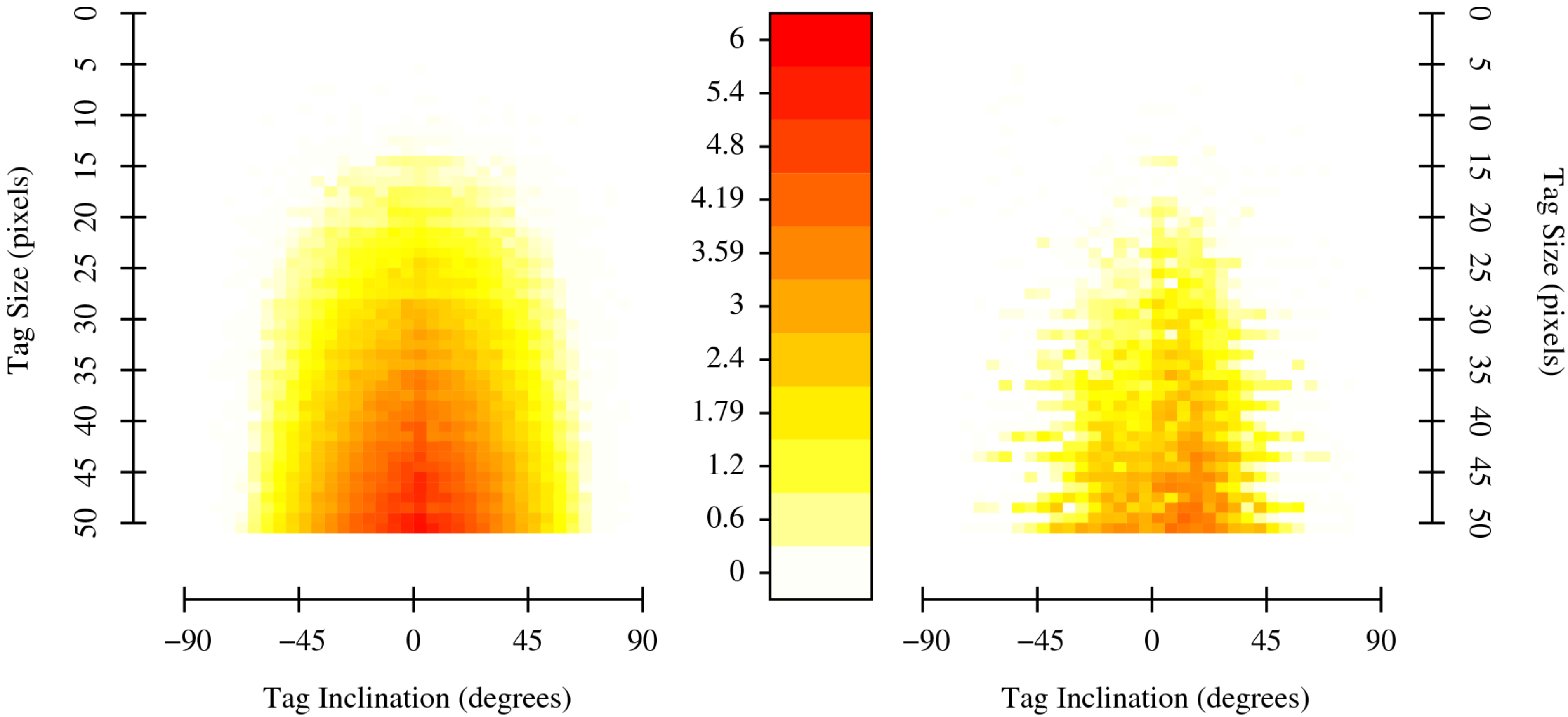
Sample Error



FitEllipseLS

FitEllipseSimple

Sample Strength

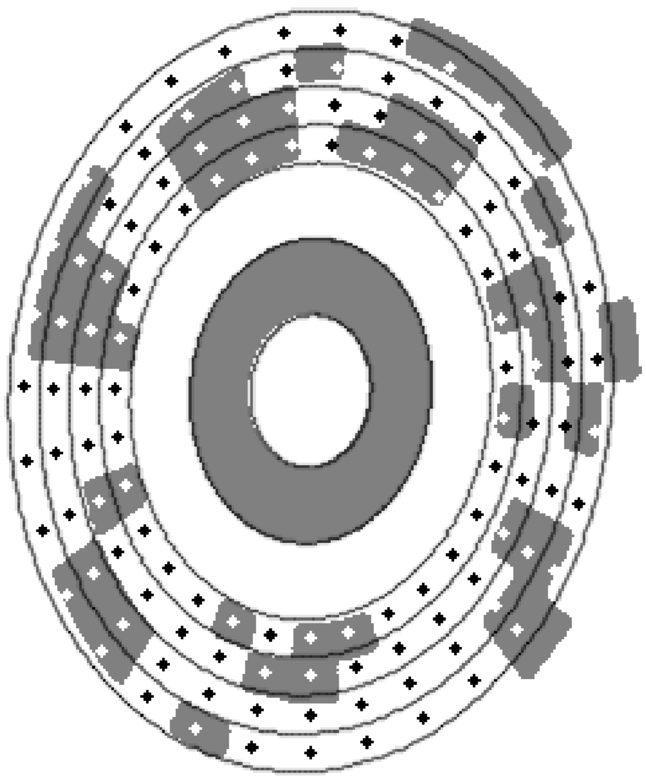


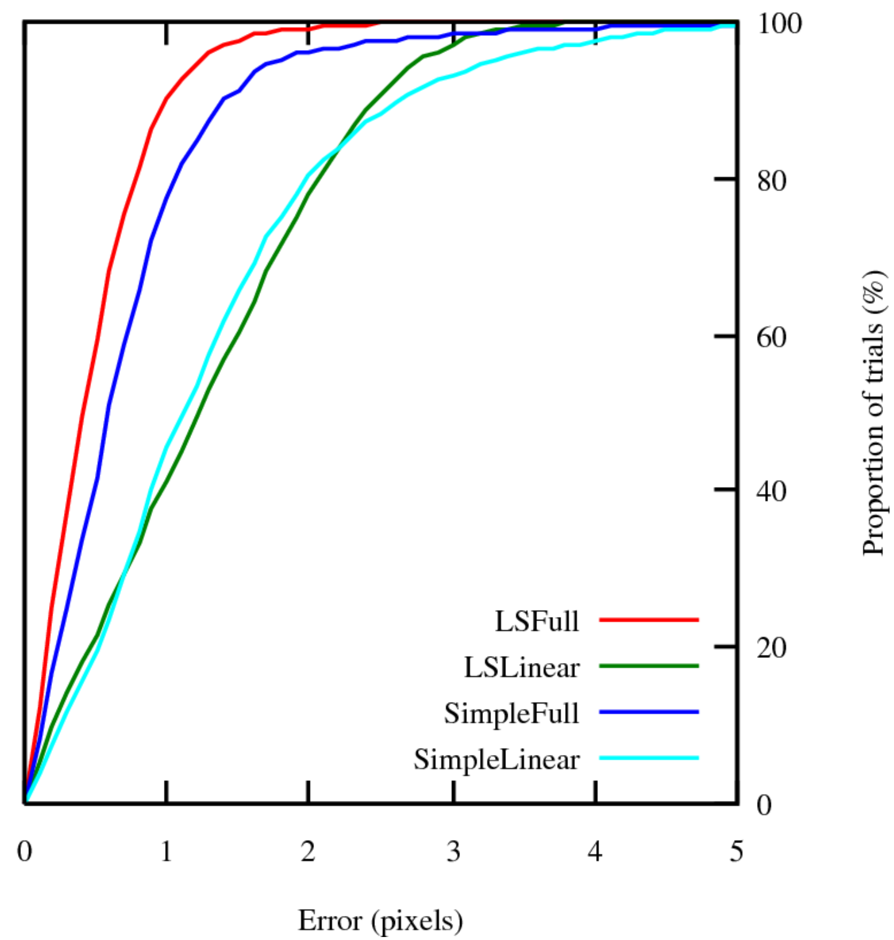
FitEllipseLS

FitEllipseSimple

Predictive Metrics

Estimate sample strength from image



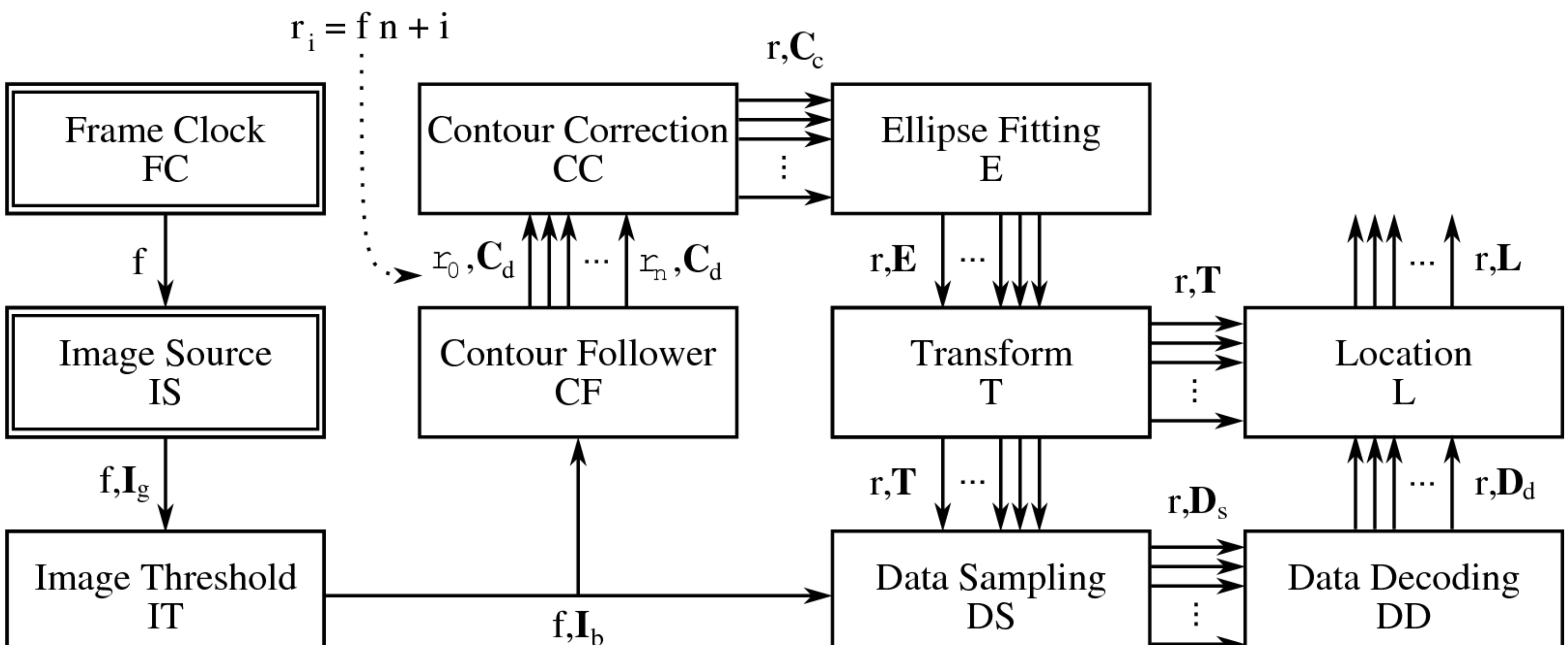


Runtime Dependability

Need to integrate our predictive metrics
with the runtime system

Also need runtime checks of our algorithms,
an asymmetric cost in many cases

The Cantag Pipeline



Inference Rules

$$\text{(IT)}\ \frac{\mathcal{V}^+_{\mathrm{IS}}(f) \qquad \mathcal{C}^+_{\mathrm{IT}}(f)}{\mathcal{V}^+_{\mathrm{IT}}(f)}$$

$$\text{(CF}^+\text{)}\ \frac{\mathcal{V}^+_{\mathrm{IT}}(f) \qquad \mathcal{C}^+_{\mathrm{CF}}(r)}{\mathcal{V}^+_{\mathrm{CF}}(r)}\ f = \lfloor r/n \rfloor \qquad \text{(CF}^-\text{)}\ \frac{\mathcal{V}^+_{\mathrm{IT}}(t) \qquad \mathcal{C}^-_{\mathrm{CF}}(r)}{\mathcal{V}^-_{\mathrm{CF}}(r)}\ t = \lfloor r/n \rfloor$$

$$\text{(CC)}\ \frac{\mathcal{V}^+_{\mathrm{CF}}(r) \qquad \mathcal{C}^+_{\mathrm{CC}}(r)}{\mathcal{V}^+_{\mathrm{CC}}(r)}$$

$$\text{(E}^+\text{)}\ \frac{\mathcal{V}^+_{\mathrm{CC}}(r) \qquad \mathcal{C}^+_{\mathrm{E}}(r)}{\mathcal{V}^+_{\mathrm{E}}(r)} \qquad \text{(E}^-\text{)}\ \frac{\mathcal{V}^+_{\mathrm{CC}}(r) \qquad \mathcal{C}^-_{\mathrm{E}}(r)}{\mathcal{V}^-_{\mathrm{E}}(r)}$$

$$\text{(T)}\ \frac{\mathcal{V}^+_{\mathrm{E}}(r) \qquad \mathcal{C}^+_{\mathrm{T}}(r)}{\mathcal{V}^+_{\mathrm{T}}(r)}$$

$$\text{(DS)}\ \frac{\mathcal{V}^+_{\mathrm{T}}(r) \qquad \mathcal{V}^+_{\mathrm{IT}}(t) \qquad \mathcal{C}^+_{\mathrm{DS}}(r)}{\mathcal{V}^+_{\mathrm{DS}}(r)}$$

$$\text{(DD}^+\text{)}\ \frac{\mathcal{V}^+_{\mathrm{DS}}(r) \qquad \mathcal{C}^+_{\mathrm{DD}}(r)}{\mathcal{V}^+_{\mathrm{DD}}(r)} \qquad \text{(DD}^-\text{)}\ \frac{\mathcal{V}^+_{\mathrm{DS}}(r) \qquad \mathcal{C}^-_{\mathrm{DD}}(r)}{\mathcal{V}^-_{\mathrm{DD}}(r)}$$

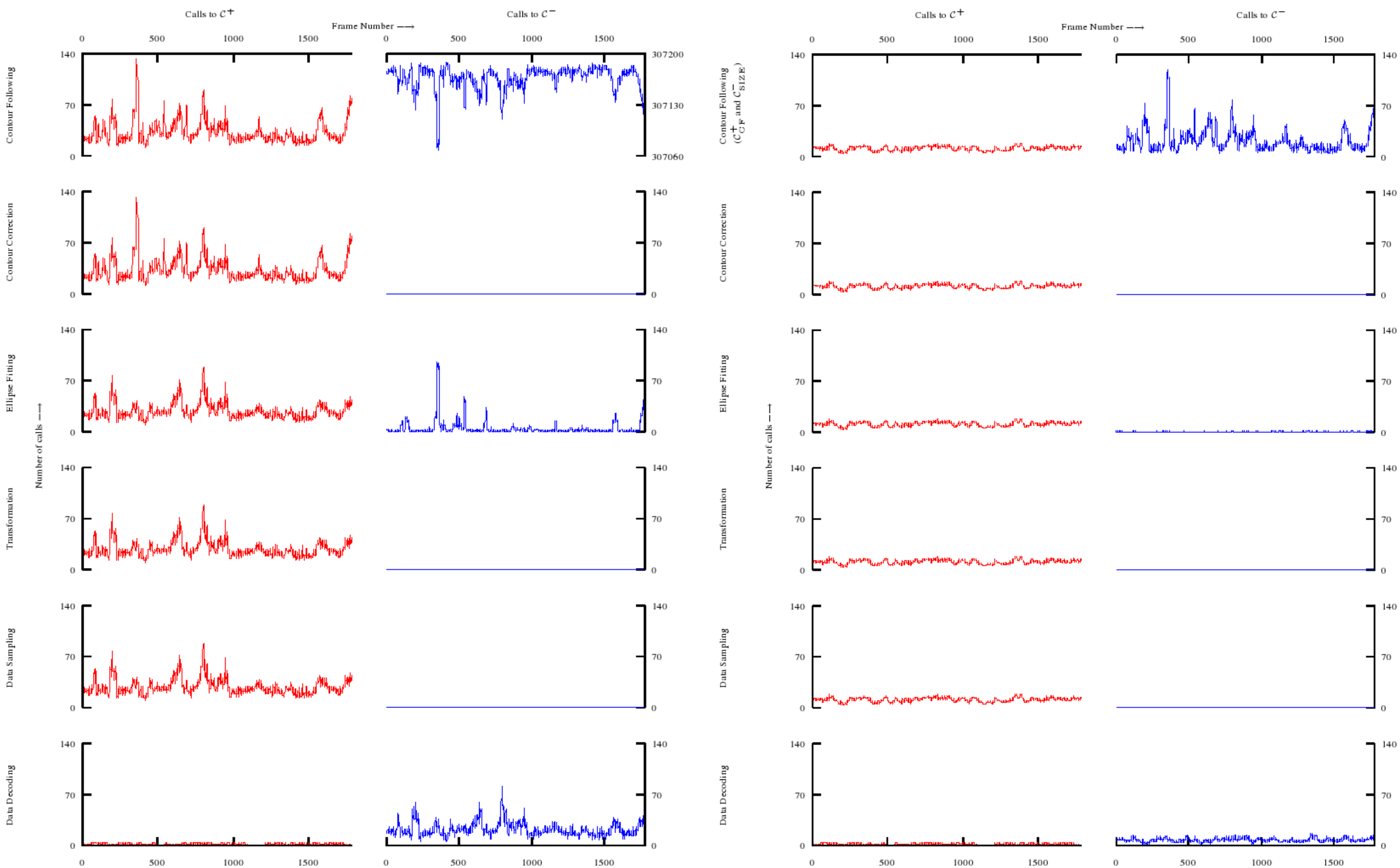
$$\text{(L}^+\text{)}\ \frac{\mathcal{V}^+_{\mathrm{DD}}(r) \qquad \mathcal{V}^+_{\mathrm{T}}(r) \qquad \mathcal{C}^+_{\mathrm{L}}(r)}{\mathcal{V}^+_{\mathrm{L}}(r)} \qquad \text{(L}^-_1\text{)}\ \frac{\mathcal{V}^-_{\mathrm{CF}}(r)}{\mathcal{V}^-_{\mathrm{L}}(r)} \qquad \text{(L}^-_2\text{)}\ \frac{\mathcal{V}^-_{\mathrm{DD}}(r)}{\mathcal{V}^-_{\mathrm{L}}(r)}$$

```
vFCp(_).                                      % Trusted Component
vISp(_).                                      % Trusted Component
n(307200).                                    % 640x480 image

vITp(T) :- vISp(T), cITp(T).                  % Rule IT
vCFp(S) :- n(R), T is S // R,
           vITp(T), cCFp(S).                  % Rule CF+
vCFn(S) :- n(R), T is S // R,
           vITp(T), cCFn(S).                  % Rule CF-
vCCp(S) :- vCFp(S), cCCp(S).                  % Rule CC
vEp(S)  :- vCCp(S), cEp(S).                   % Rule E+
vEn(S)  :- vCCp(S), cEn(S).                   % Rule E+
vTp(S)  :- vEp(S), cTp(S).                    % Rule T
vDSp(S) :- vTp(S), vITp(S),
           cDSp(S).                           % Rule DS
vDDp(S) :- vDSp(S), cDDp(S).                  % Rule DD+
vDDn(S) :- vDSp(S), cDDn(S).                  % Rule DD-
vLp(S)  :- vDDp(S), vTp(S),
           cLp(S).                            % Rule L+
vLn1(S) :- vCFn(S).                           % Rule L1-
vLn2(S) :- vDDn(S).                           % Rule L2-
vLn(S)  :- vLn1(S) ; vLn2(S).
```

Use prolog as automated checker

Costs of Validation



Improving Performance

Reorganise original system to ease the inference burden

Runtime costs are reduced but rule complexity increases

```
theory Bat
imports Main
begin

theorem laststar:
  [| ∀ t. VPP t ⟶ LASTP(t,t) ;
    ∀ t tp. LASTP(t,tp) ∧ VPM (Suc t) ⟶ LASTP(Suc t,tp) ;
    ∀ t te n. SP(n,0) ∧ SP(Suc n,te) ∧ 0 < t ∧ t < te ⟶ VPM t |]
  ⟹
  (∀ t te n. SP(n,0) ∧ SP(Suc n,te) ∧ 0 < t ∧ t < te ∧ VPP 0 ⟶ LASTP(t,0))
  ∧
  (∀ t n te tp. SP(n,0) ∧ SP(Suc n,te) ∧ 0 ≤ t ∧ t < te ∧ VPM 0 ∧ LASTP(0,tp)
     ⟶ LASTP(t,tp))
  apply(subgoal-tac
   ∀ n te.
     (∀ t. SP(n,0) ∧ SP(Suc n,te) ∧ 0 ≤ t ∧ t < te ∧ VPP 0 ⟶ LASTP(t,0))
   ∧
     (∀ t tp. SP(n,0) ∧ SP(Suc n,te) ∧ 0 < t ∧ t < te ∧ VPM 0 ∧ LASTP(0,tp)
       ⟶ LASTP(t,tp))
  )
  apply(blast)
  apply(rule allI, rule allI)
  apply(subgoal-tac
   [| ∀ t. SP(n,0) ∧ SP(Suc n,te) ∧ 0 < t ∧ t < te ⟶ VPM t |]
   ⟹
     (∀ t. SP(n,0) ∧ SP(Suc n,te) ∧ 0 ≤ t ∧ t < te ∧ VPP 0 ⟶ LASTP(t,0))
   ∧
     (∀ t tp. SP(n,0) ∧ SP(Suc n,te) ∧ 0 < t ∧ t < te ∧ VPM 0 ∧ LASTP(0,tp)
      ⟶ LASTP(t,tp))
  )
  apply(blast)
  apply(rule conjI, rule allI)
  apply(induct-tac t)
  apply(simp all)
  apply(rule allI, rule allI)
  apply(induct-tac t)
  apply(simp-all)
  done
end
```

Conclusion

- Recipe for dependability
  1) Provide as reliable an implementation as feasible
  2) Develop predictive metrics
  3) Identify observable metrics and validate them
  4) Integrate into the real system using a Prolog reasoning engine
  5) Add additional inferences for performance improvements---prove these correct in HOL

The End

Andrew Rice, Christopher Cain and John Fawcett. Dependable Coding for Fiducial Tags. In *Proceedings of the 2nd Ubiquitous Computing Symposium*, pages 155--163, 2004.

Andrew Rice, Christopher Cain and John Fawcett. Dependable Coding for Fiducial Tags (Extended Version). In *Ubiquitous Computing Systems*, LNCS 3598, pages 259--274, 2004.

Andrew Rice and Robert Harle. Evaluating Lateration-Based Positioning Algorithms for Fine-Grained Tracking. In *Joint Workshop on Foundations of Mobile Computing* (DIAL-M-POMC), pages 54--61, 2005.

Andrew C Rice, Alastair R Beresford and Robert K Harle. Cantag: an open source software toolkit for designing and deploying marker-based vision systems. In *Fourth Annual IEEE International Conference on Pervasive Computer and Communications,* pages 12--21, 2006.

Andrew C Rice and Alastair R Beresford. Dependability and Accountability for Context-aware Middleware Systems. In *Workshop on Middleware Support for Pervasive Computing* (PerWare), pages 378--382, 2006.

Andrew C Rice, Robert K Harle and Alastair R Beresford. Analysing fundamental properties of marker-based vision system designs. *Pervasive and Mobile Computing*, November 2006.