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.

Varying lighting intensity across the image causes false-negatives.

Tags on the edge of the image are most affected by lens distortion.

Shape fitting quality could be increased by activating the linear regression algorithm to refine the corner estimates.

Accumulated error from previous stages causes inaccuracy in the sample points for the data payload.

The SpaceSearch algorithm maintains an orthogonal 3D co-ordinate frame.

A small window size amplifies noise in the solid regions of the tag.

Simple correction (middle) shows little benefit compared to Full correction (right) for the current lens.

Decode
Algorithmic Dependability

How does my system behave in theory?
Abstract, information-theoretic limit

Optimising tag design requires maximising the sample distance
Sample Error

cell width = w

\( \frac{w}{2} = \text{sample distance} = d \)

e = sample error

d - e < 0
Sample Error

FitEllipseLS

FitEllipseSimple
Sample Strength

Tag Size (pixels)

Tag Inclination (degrees)

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

Frame Clock (FC)

Image Source (IS)

Image Threshold (IT)

Contour Correction (CC)

Contour Follower (CF)

Ellipse Fitting (E)

Transform (T)

Location (L)

Data Sampling (DS)

Data Decoding (DD)

\[ r_i = f n + i \]

\[ r, C_c \]

\[ r, E \]

\[ r, T \]

\[ r, D_s \]

\[ r, D_d \]
Inference Rules

\[
\begin{align*}
\text{(IT)} & \quad \psi_{IT}(f) \quad \psi_{IT}(f) \\
\text{(CF+)} & \quad \psi_{IT}(f) \quad \psi_{CF}(r) \quad c_{CF}(r) \\
\text{(CF-)} & \quad \psi_{IT}(t) \quad \psi_{CF}(r) \quad c_{CF}(r) \\
\text{(CC)} & \quad \psi_{CF}(r) \quad \psi_{CC}(r) \\
\text{(E)} & \quad \psi_{CC}(r) \quad \psi_{CE}(r) \\
\text{(E-)} & \quad \psi_{CE}(r) \quad \psi_{E}(r) \\
\text{(T)} & \quad \psi_{E}(r) \quad \psi_{T}(r) \\
\text{(DS)} & \quad \psi_{T}(r) \quad \psi_{IT}(t) \quad c_{DS}(r) \\
\text{(DD+)} & \quad \psi_{DS}(r) \quad \psi_{DD}(r) \\
\text{(DD-)} & \quad \psi_{DS}(r) \quad \psi_{DD}(r) \\
\text{(L+)} & \quad \psi_{DD}(r) \quad \psi_{L}(r) \quad c_{L}(r) \\
\text{(L-)} & \quad \psi_{CF}(r) \quad \psi_{L}(r) \\
\text{(L2)} & \quad \psi_{DD}(r) \quad \psi_{L}(r)
\end{align*}
\]

\[
\begin{align*}
vFCp(\_). & \quad n(307200). \\
vISp(\_). & \quad n(307200). \\
vITp(T) & :- vISp(T), cITp(T). \\
vCFp(S) & :- n(R), T \text{ is } S // R, vITp(T), cCFp(S). \\
vCFn(S) & :- n(R), T \text{ is } S // R, vITp(T), cCFn(S). \\
vCCp(S) & :- vCFp(S), cCCp(S). \\
vEp(S) & :- vCCp(S), cEp(S). \\
vEn(S) & :- vCCp(S), cEn(S). \\
vT(S) & :- vEp(S), cT(S). \\
vDp(S) & :- vT(S), vITp(S), cDp(S). \\
vDDp(S) & :- vDp(S), cDDp(S). \\
vDDn(S) & :- vDp(S), cDDn(S). \\
vLP(S) & :- vDDp(S), vT(S), cLP(S). \\
vLn1(S) & :- vCFn(S). \\
vLn2(S) & :- vDn(S). \\
vLn(S) & :- vLn1(S); vLn2(S).
\end{align*}
\]

Use prolog as automated checker
Costs of Validation

Charts showing the costs of various validation processes, including Call to C, Frame Number, and Call to C++.
Improving Performance

Reorganise original system to ease the inference burden

Runtime costs are reduced but rule complexity increases
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


