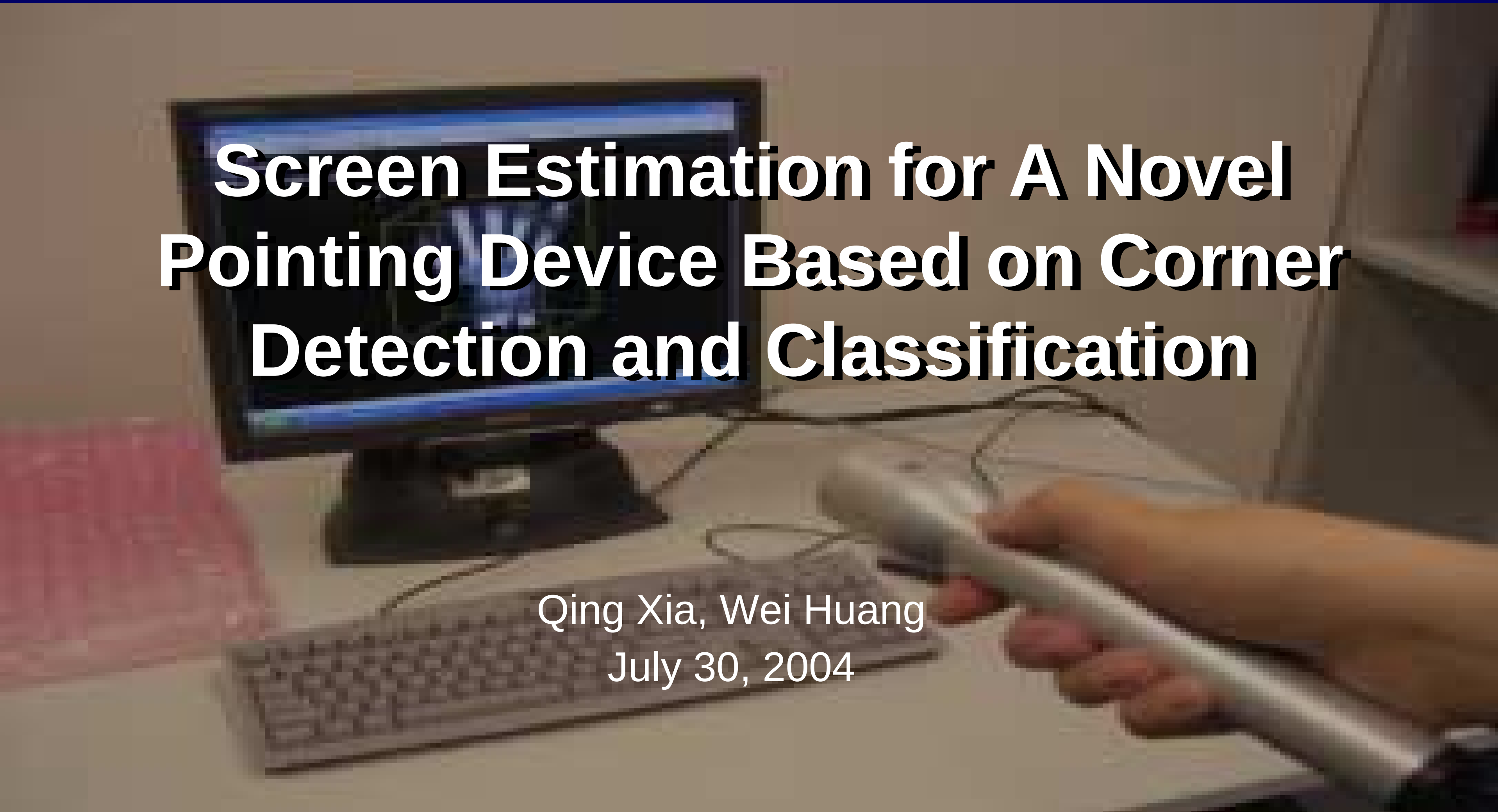
PHILIPS

TU Delft
Delft University of Technology



Screen Estimation for A Novel Pointing Device Based on Corner Detection and Classification

Qing Xia, Wei Huang

July 30, 2004

Overview

- Part I: Problem Definition
- Part II: Models and Algorithms
- Part III: System Implementation Design
- Part IV: System Tests and Conclusions

Part I: Problem Definition

- Introduction and Problem Definition
- Models and Algorithms Overview

Introduction and Problem Definition

Pointing devices

- Microsoft X-Wand
- Infrared pointing device
- HeadMouse
- CyberlinkMindMouse

UI-Wand project

- Camera-Based Approach
- Related Works

UI-Wand hardware components

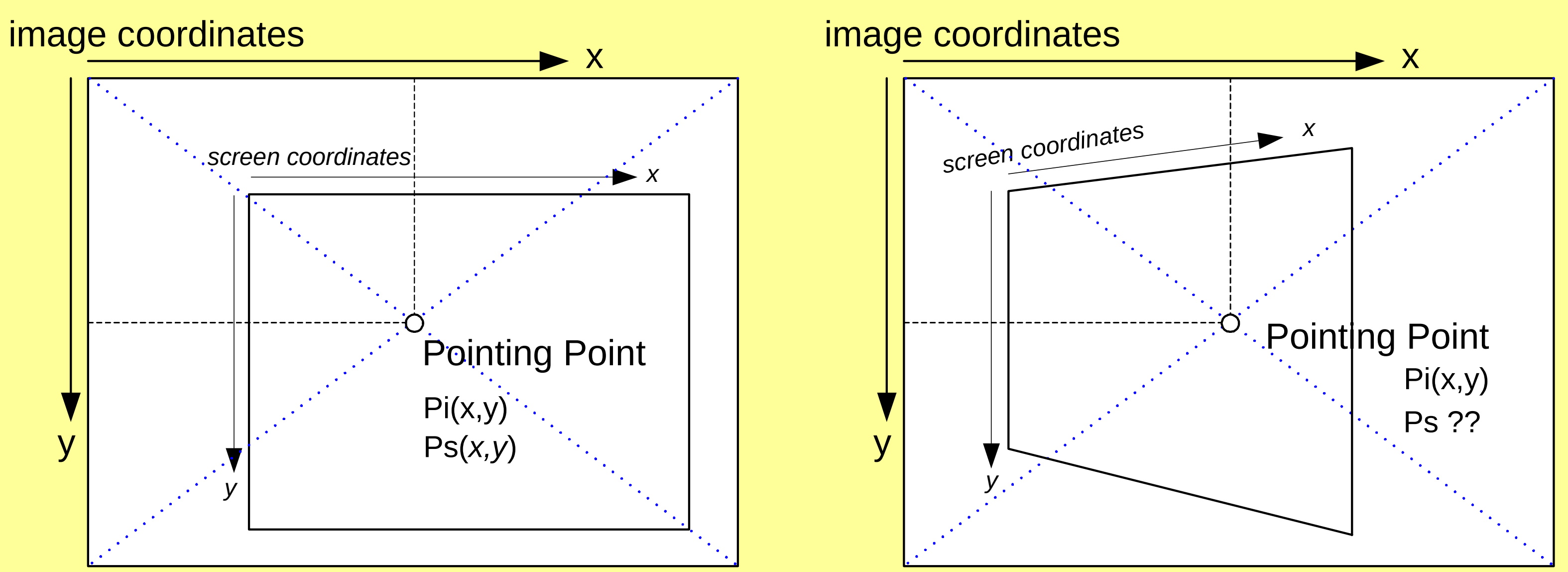
Project goals

- Screen positioning by screen corner detection
- Gesture recognition
- Robustness and stability
- Develop utilities to evaluate the system
- Real-time speed (10 frames/sec)
- Development based on Vispirin

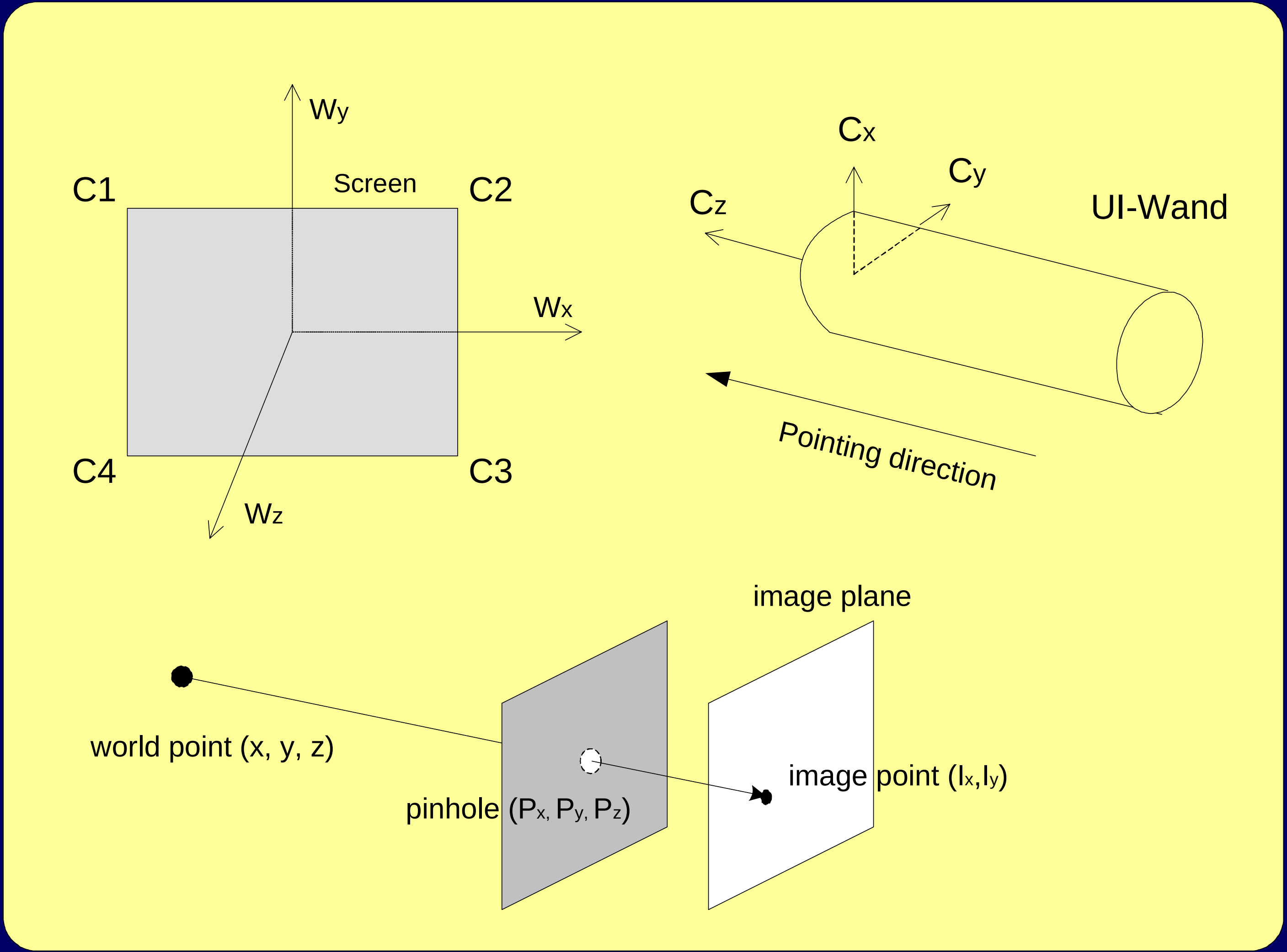
Models and Algorithms Overview

Pointing Projection

- How to calculate Ps?



Pointing Projection



Pointing Projection

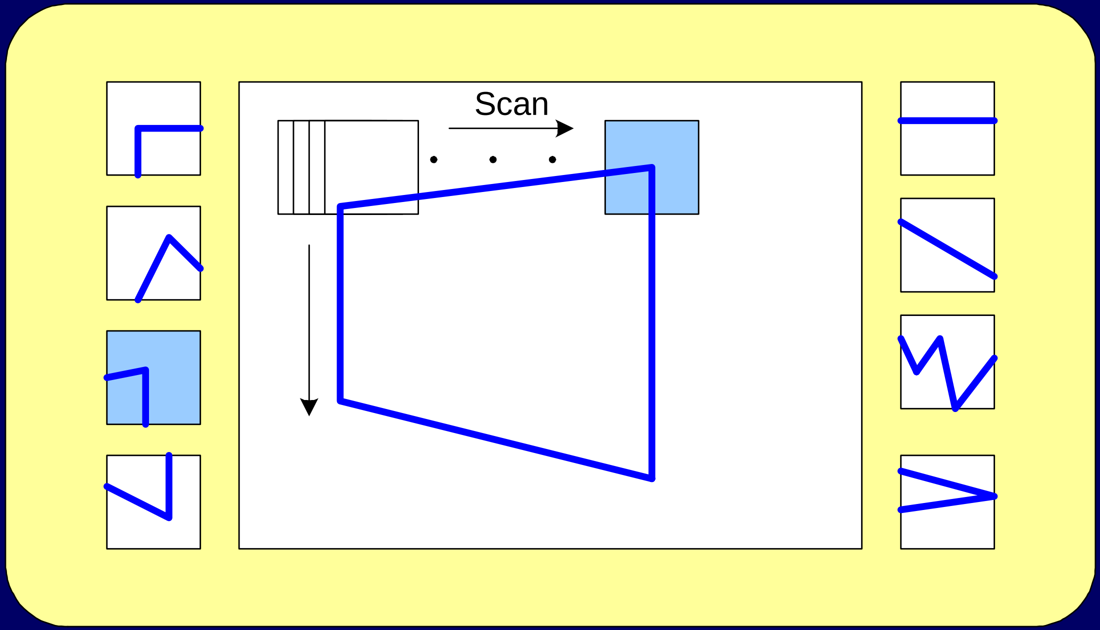
- Problem is to estimate a new pointing device position

$$\pi_{P,O} : (x, y, z) \rightarrow (I_x, I_y)$$

$$(P', O') = \operatorname{argmin}_{(P,\ O)} \sum_{i=1}^{4} \left(\pi_{P,O}(C_i) - D_i\right)^2$$

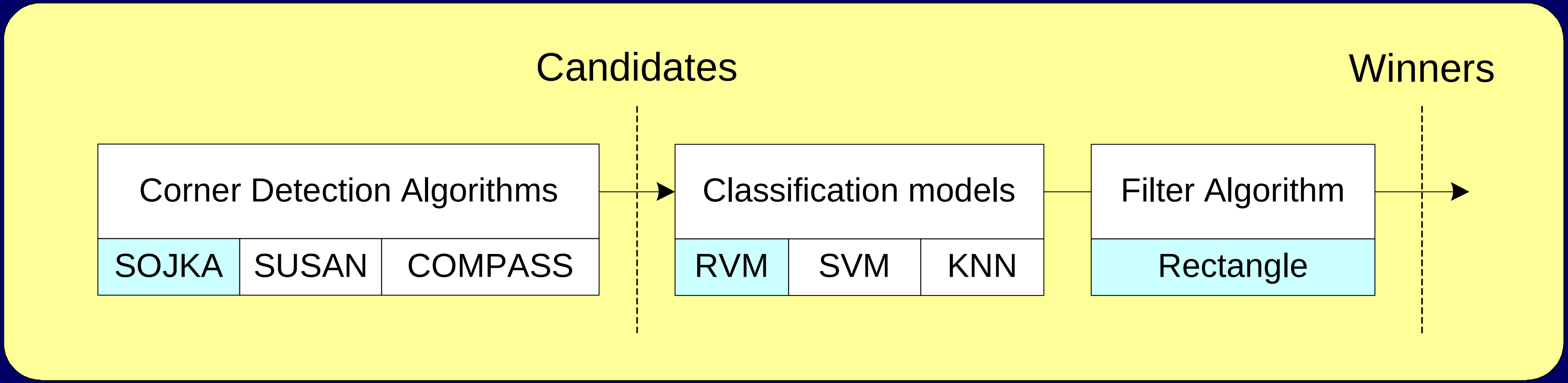
Screen corner detection

- Direct corner detection (gradient of brightness)
- Classification



Screen corner detection

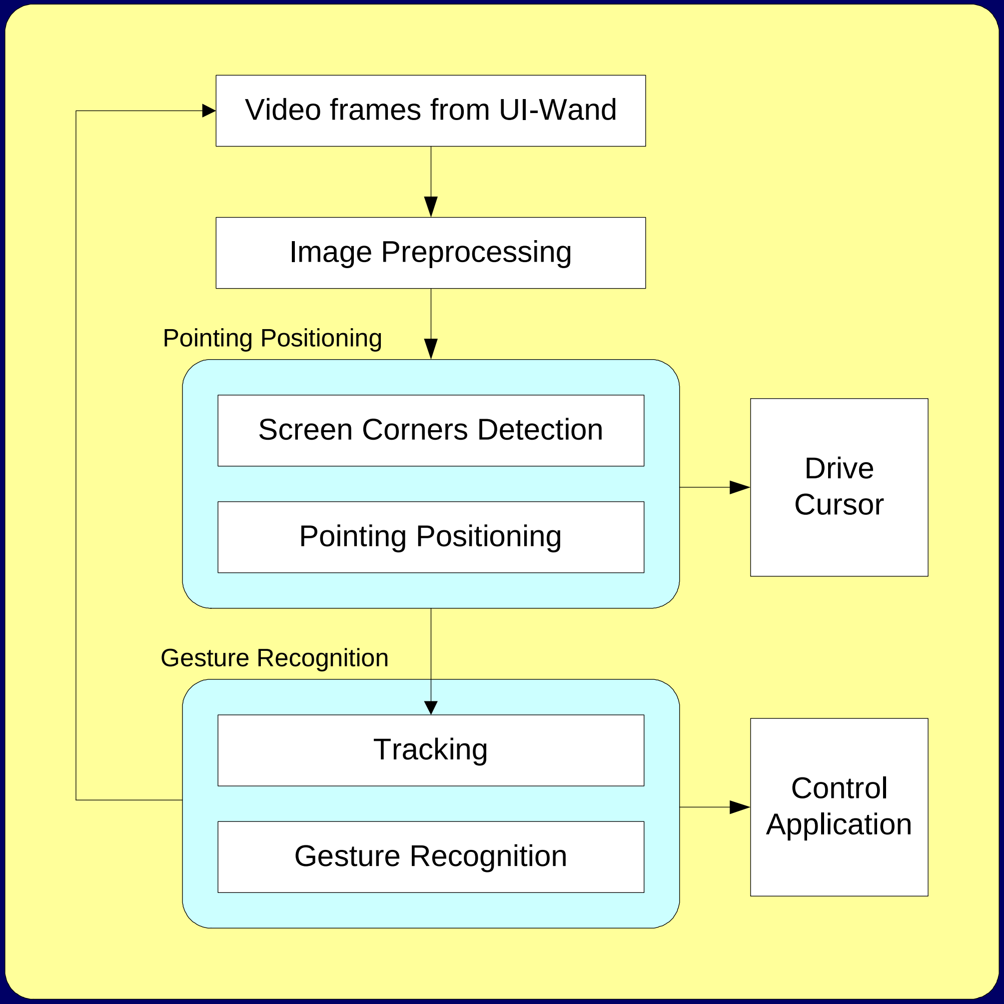
- All corners are candidates
- Four screen corners are final winners



RVM: Relevance Vector Machine

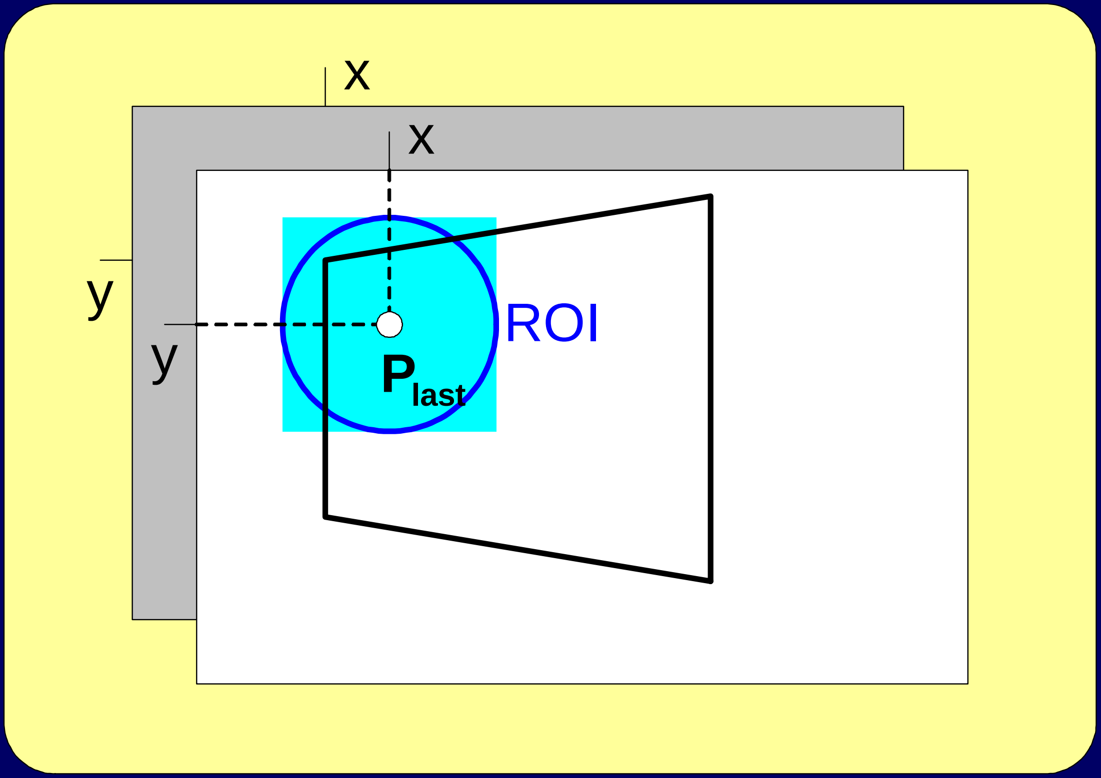
Main Modules

- Screen corner detection
- Pointing positioning
- Tracking
- Gesture recognition



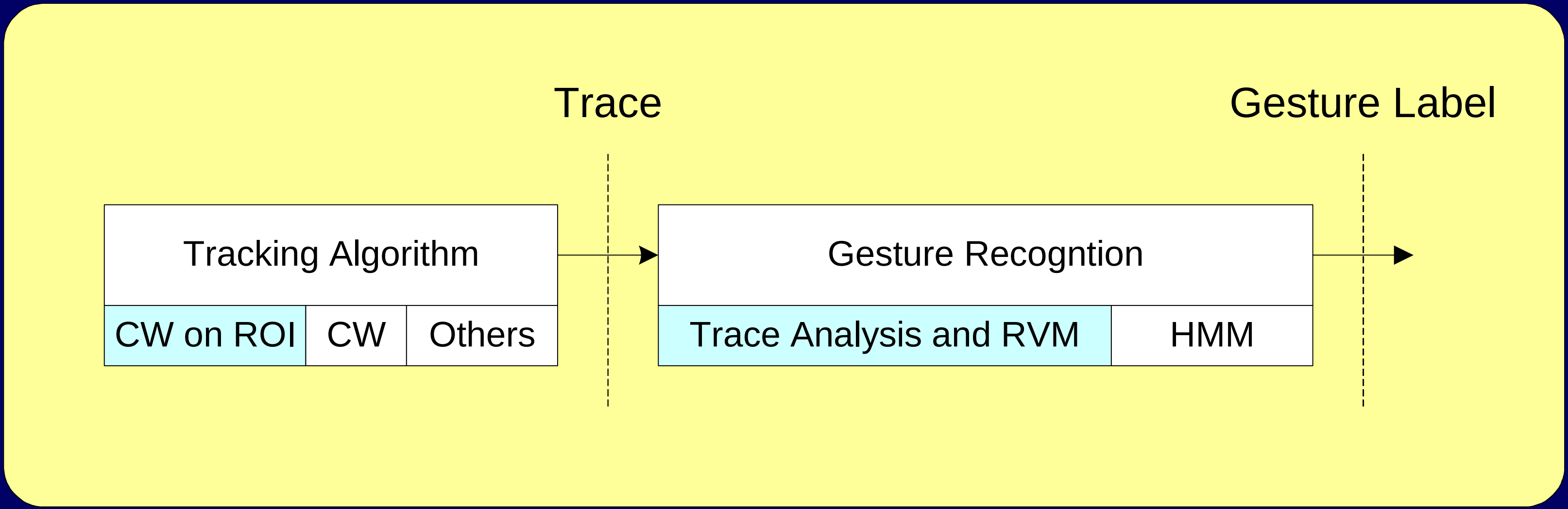
ROI Tracking

- Detect screen corners only in ROIs



Gesture Recognition

- Trace analysis
- Gesture recognition by RVM



CW: Candidates-Winners approach

Part II: Models and Algorithms

- Sojka Corner Detection Algorithm
- RVM Corner Classification Model
- Rectangle Filter
- ROI Tracking Filter
- RVM for Gesture Recognition

Sojka Corner Detection Algorithm

Corner Detection Algorithms

- Direct corner detectors
- Color distribution based corner detectors

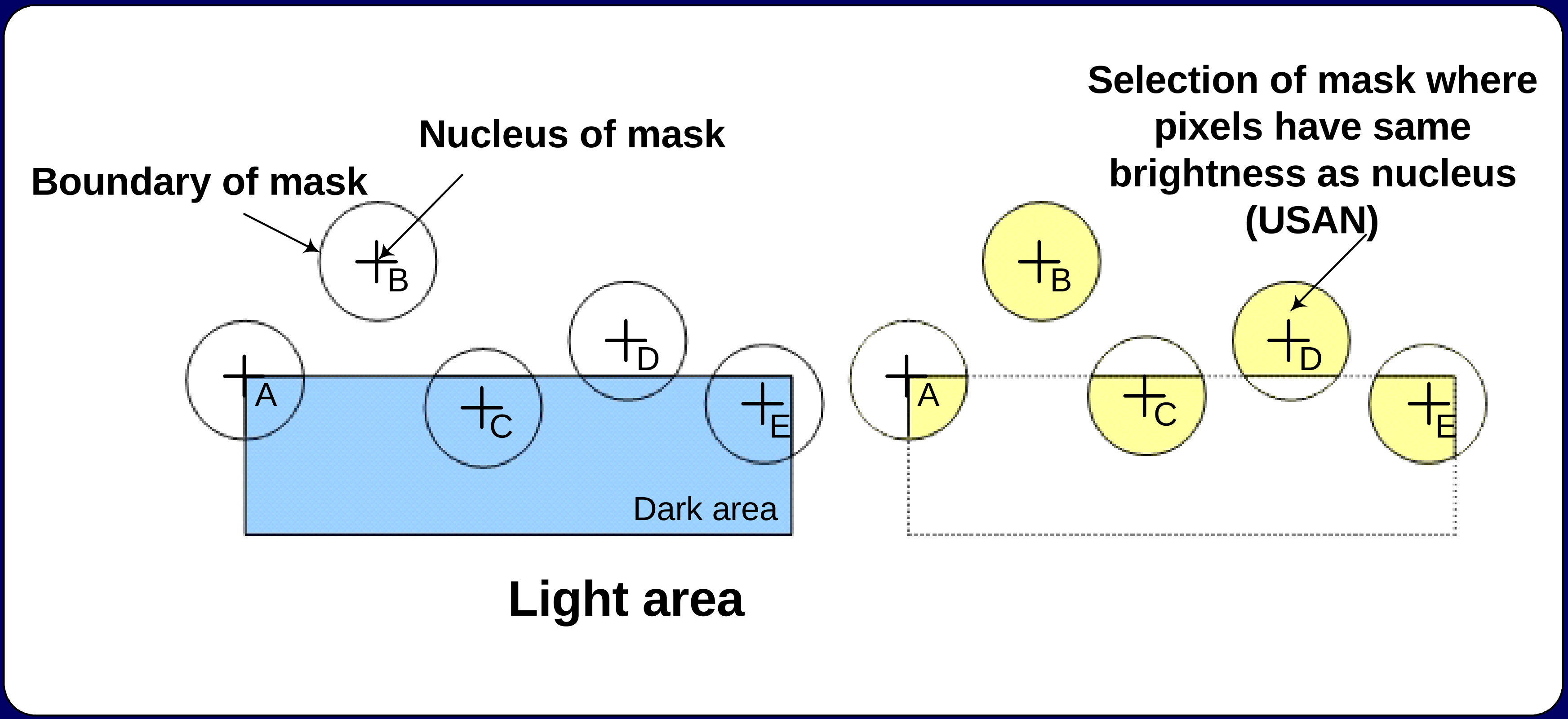
Direct Corner Detection Algorithms

- Beaudet’s detector [Bea78]
- Kitchen and Rosenfeld’s detector [Kit82]
- Harris and Stephens’ detector [Har88]
- Deriche and Giraudon’s detector [Der93]
- Sojka corner detector [Soj03]

Other Corner Detection Algorithms

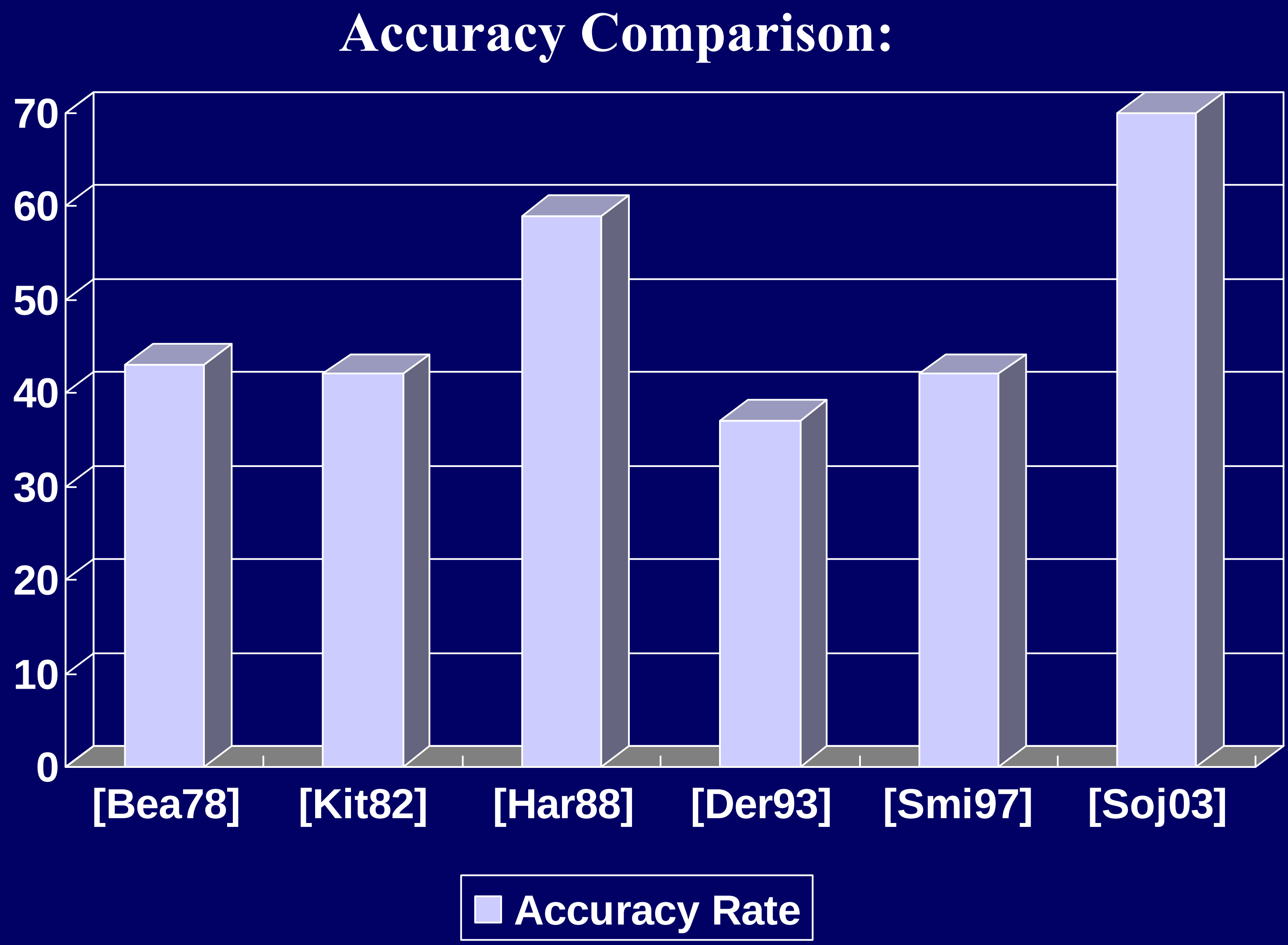
- SUSAN corner detector [Smi97]
- Compass operator [Ruz01]
- A color-distribution-based detector [Son03]

SUSAN Corner Detector



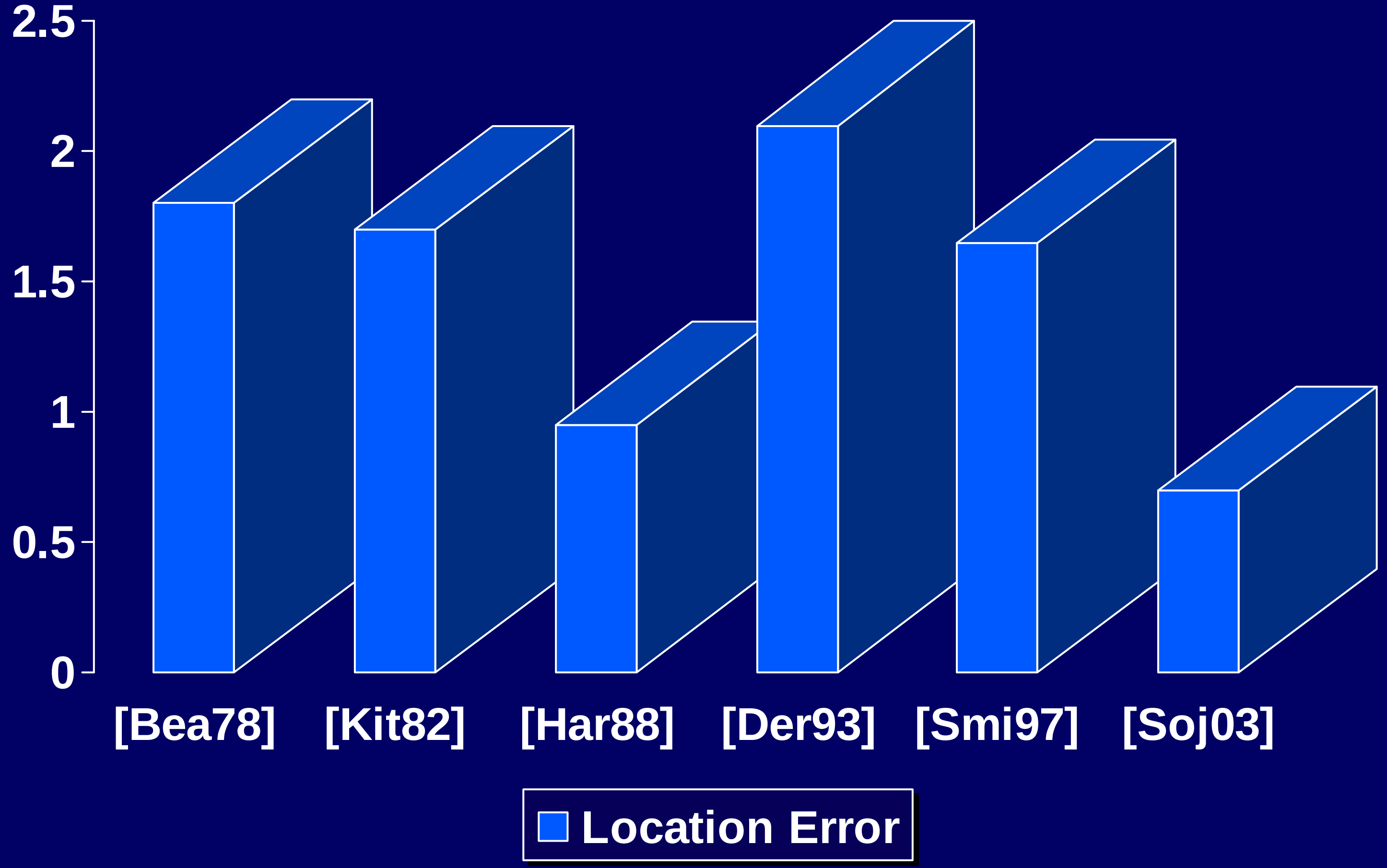
Comments: Simple, Fast, Low accuracy

Algorithms Comparison I



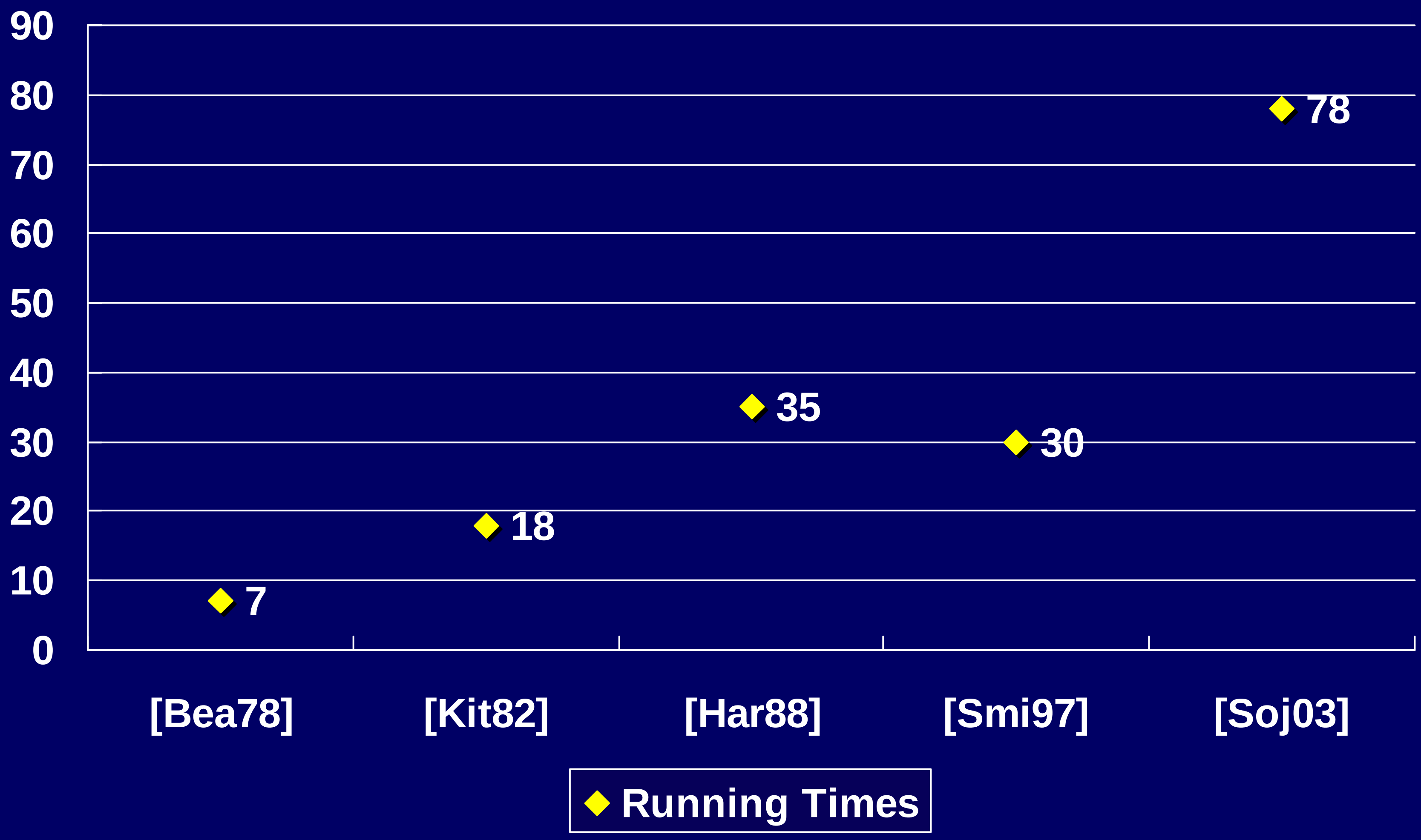
Algorithms Comparison II

Location Error Comparison:



Algorithms Comparison III

Detection Speed Comparison (in Millisecond):



Comparisons Conclusion

- The older direct corner detectors always have low detection accuracy rate
- Most of existing corner detector can be used in real time detection tasks
- The Color distribution based algorithms are only fit for still image analysis
- Sojka corner detector has obvious excellent performance compared to the other detectors

Make a Decision

Sojka corner detector!

- High accuracy
- Low location error
- Fast enough

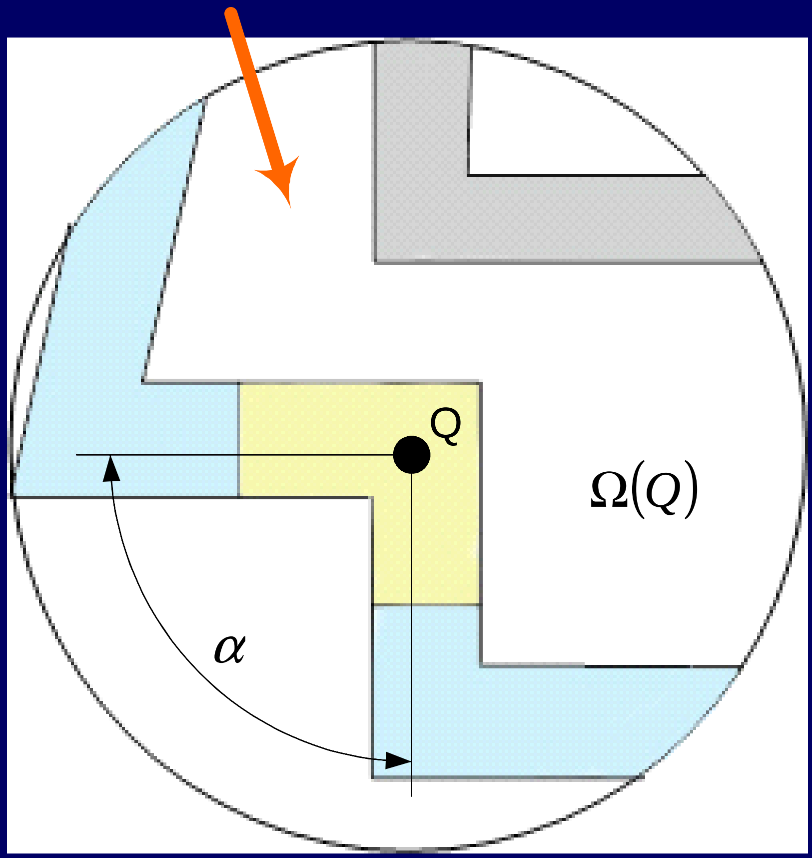
Sojka Corner Detection Algorithm

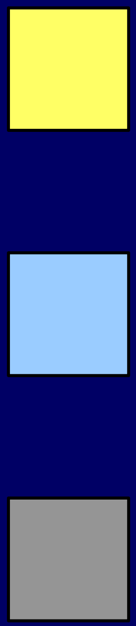
Main ideas:

- Exploiting the probability information
- Explicitly computing the corner angle
- Computing apparency of the corner

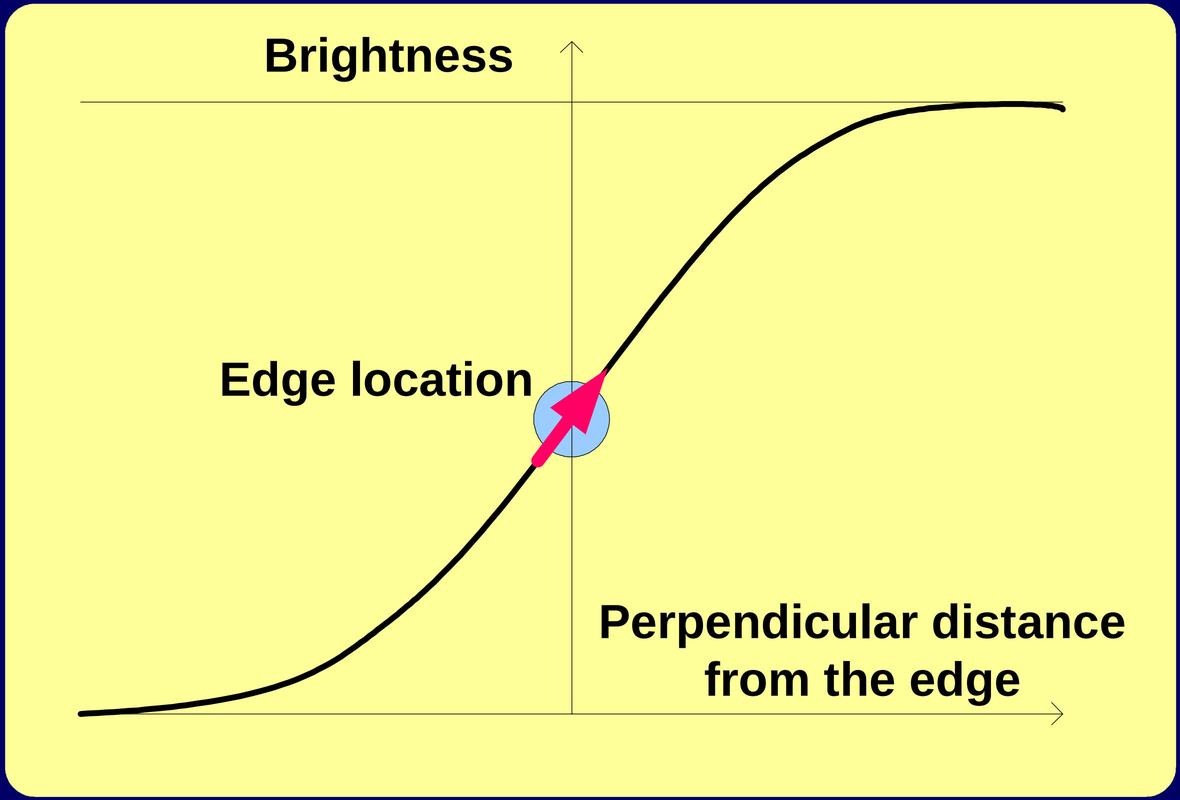
Main ideas of Sojka Corner Detection Algorithm

Neighborhood of Q





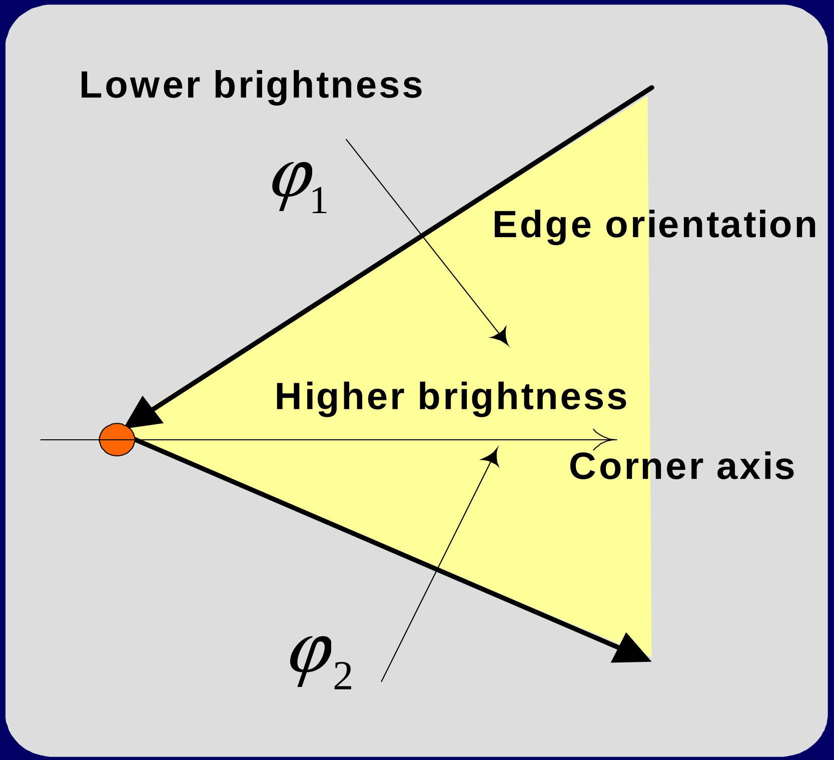
Corner Model Definition I



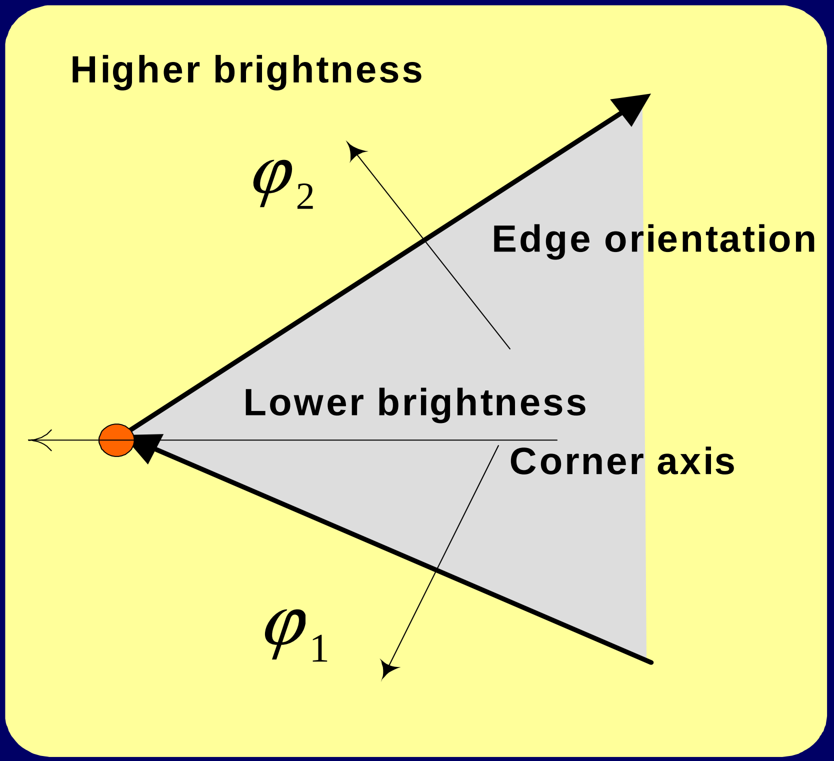
Let the derivative of brightness be positive everywhere with a single maximum at 0.

Corner Model Definition II

The edge is oriented by the rule that the higher brightness lies to the left.



A convex corner model



A concave corner model

Corner Model Definition III

Gradient direction vector: $\mathbf{n}_i = (\cos\varphi_i, \sin\varphi_i)$

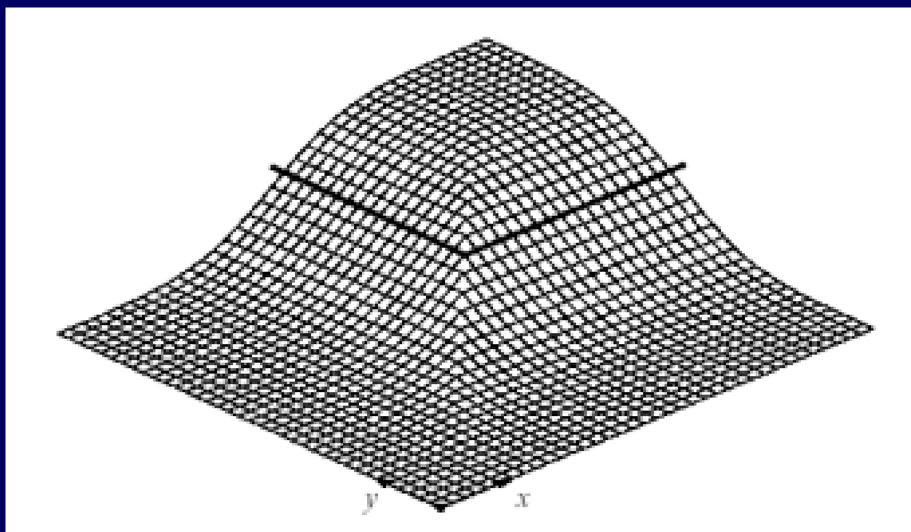
Convex corner: $\mathbf{n}_1 \times \mathbf{n}_2 > 0$

Concave corner: $\mathbf{n}_1 \times \mathbf{n}_2 < 0$

Brightness definition:

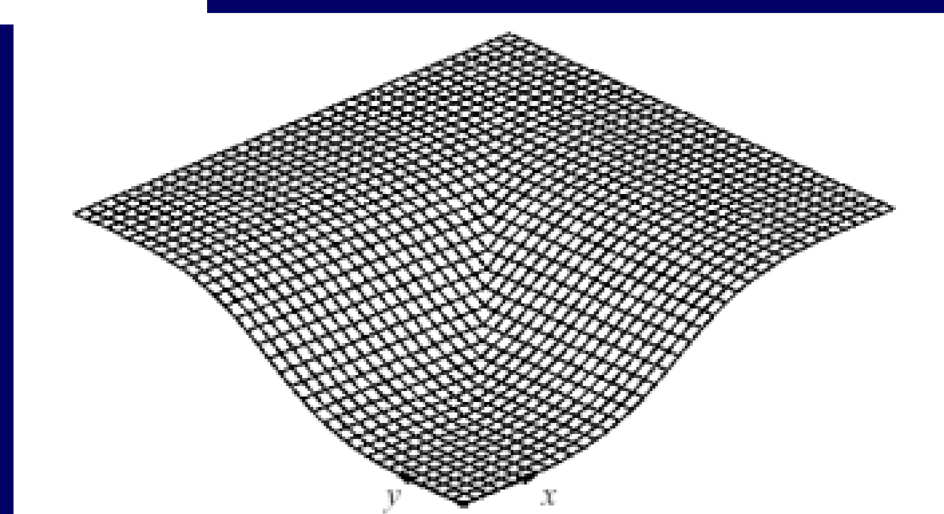
$$b(X) = \begin{cases} \min\{\psi(\mathbf{n}_1 \cdot (X - C)), \psi(\mathbf{n}_2 \cdot (X - C))\} & \text{if } \mathbf{n}_1 \times \mathbf{n}_2 \geq 0 \\ \max\{\psi(\mathbf{n}_1 \cdot (X - C)), \psi(\mathbf{n}_2 \cdot (X - C))\} & \text{otherwise} \end{cases}$$

Corner Model Examples



Convex corner

Concave corner



Three Conditional Probabilities

- Conditional probability $P\left(BrQ\middle|\Delta b\left(X\right)\right)$
- Conditional probability $P\left(DirQ\middle|h\left(X\right)\right)$
- Conditional probability $P\left(AngQ\middle|\Delta\varphi\left(X\right)\right)$

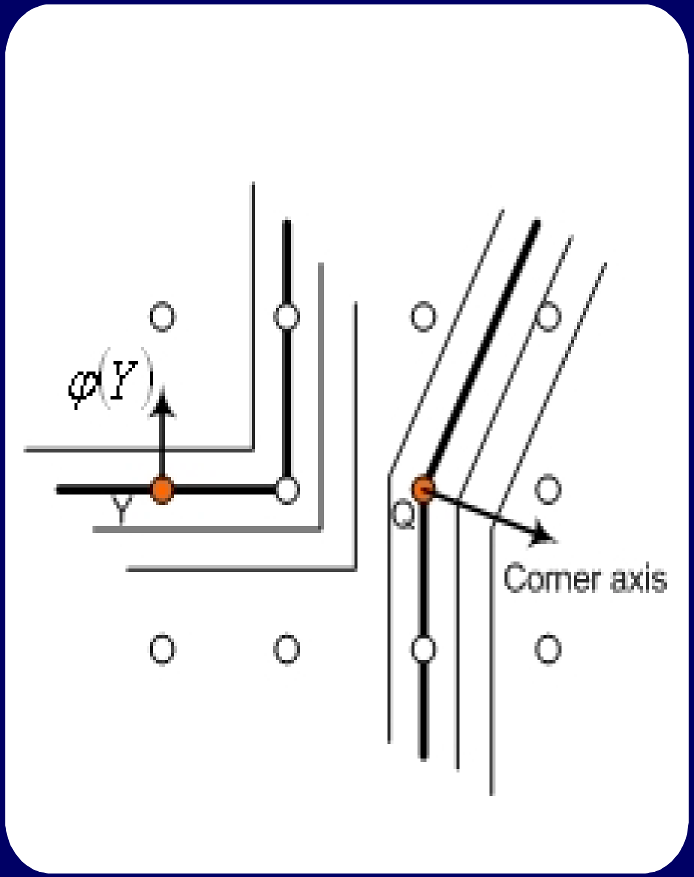
Combined Probability

$$P_{SG}(X) = \min_{Y \in QX} \left\{ P\left(BrQ \middle| \Delta b(Y)\right) P\left(DirQ \middle| h(Y)\right) P\left(AngQ \middle| \Delta\varphi(Y)\right) \right\}$$

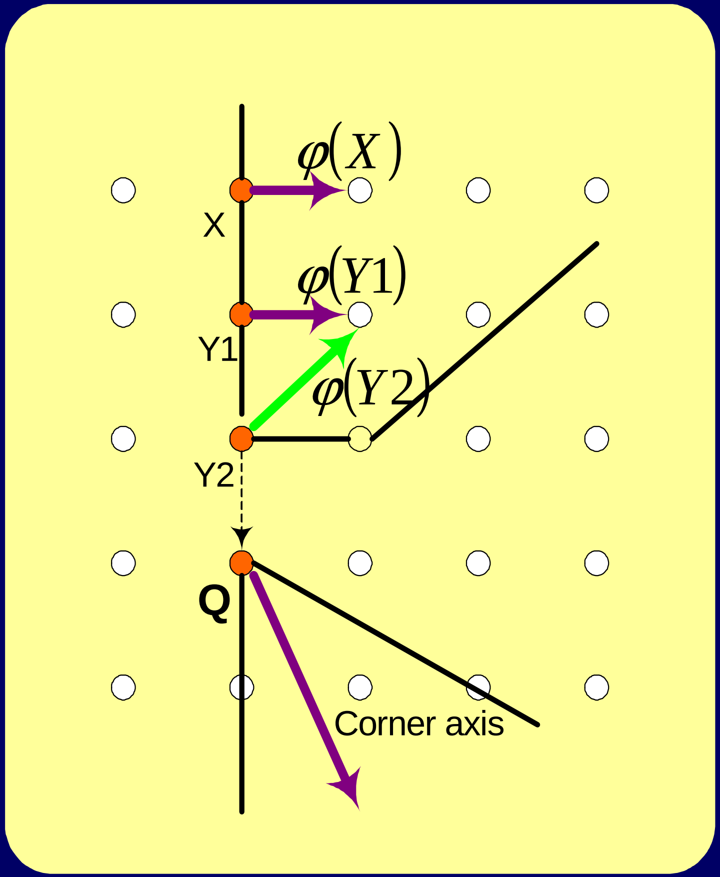
$$w(X) = P_{SG}(X) w_r(r(X)) = A_0 \min_{y \in QX} \left\{ p_d\left(\beta^{-1}(\Delta b(Y))\right) p_d(h(Y)) P\left(AngQ \,|\, \Delta\varphi(Y)\right) \right\} w_r(r(X)).$$

Why

Why? Because…



(a)



(b)

Corner Point Decision I

- $$\mu_\varphi = \frac{\sum_{X_i \in \Omega} w(X_i)\varphi(X_i)}{\sum_{X_i \in \Omega} w(X_i)\varphi}$$ $$\sigma_\varphi^2 = \frac{\sum_{X_i \in \Omega} w(X_i)\left[\varphi(X_i) - \mu_\varphi\right]^2}{\sum_{X_i \in \Omega} w(X_i)}$$

  Mean values that are computed for a corner candidate point

- --- $$Corr(Q) = g(Q)\sigma_\varphi^2(Q)$$

- --- $$Appar(Q) = \sum_{X_i \in \Omega} P_{SG}(X_i)g(X_i)\left|\varphi(X_i) - \mu_\varphi\right|$$

  Two functions for final decisions

Corner Point Decision II

- Theorem 1.

  --- The function $\sigma_{\varphi}^{2}(Q)$ has its maximum just at then points lying on the axis of the corner.

- Theorem 2.

  --- The function $Corr(Q)$ has its maximum just at the corner point.

Algorithm Realization

- Step 1: Computing the magnitude and the direction of the gradient of brightness.
- Step 2: Selecting the candidates corner points.
- Step 3: Getting the value of $\mu_\varphi(Q)$ $\sigma_\varphi^2(Q)$ $Corr(Q)$ $Appar(Q)$ by value estimation.
- Step 4: The candidates with local maximum $Corr(Q)$ and at which the value of $\sigma_\varphi^2(Q)$ $Appar(Q)$ extending some thresholds are specified as corners.

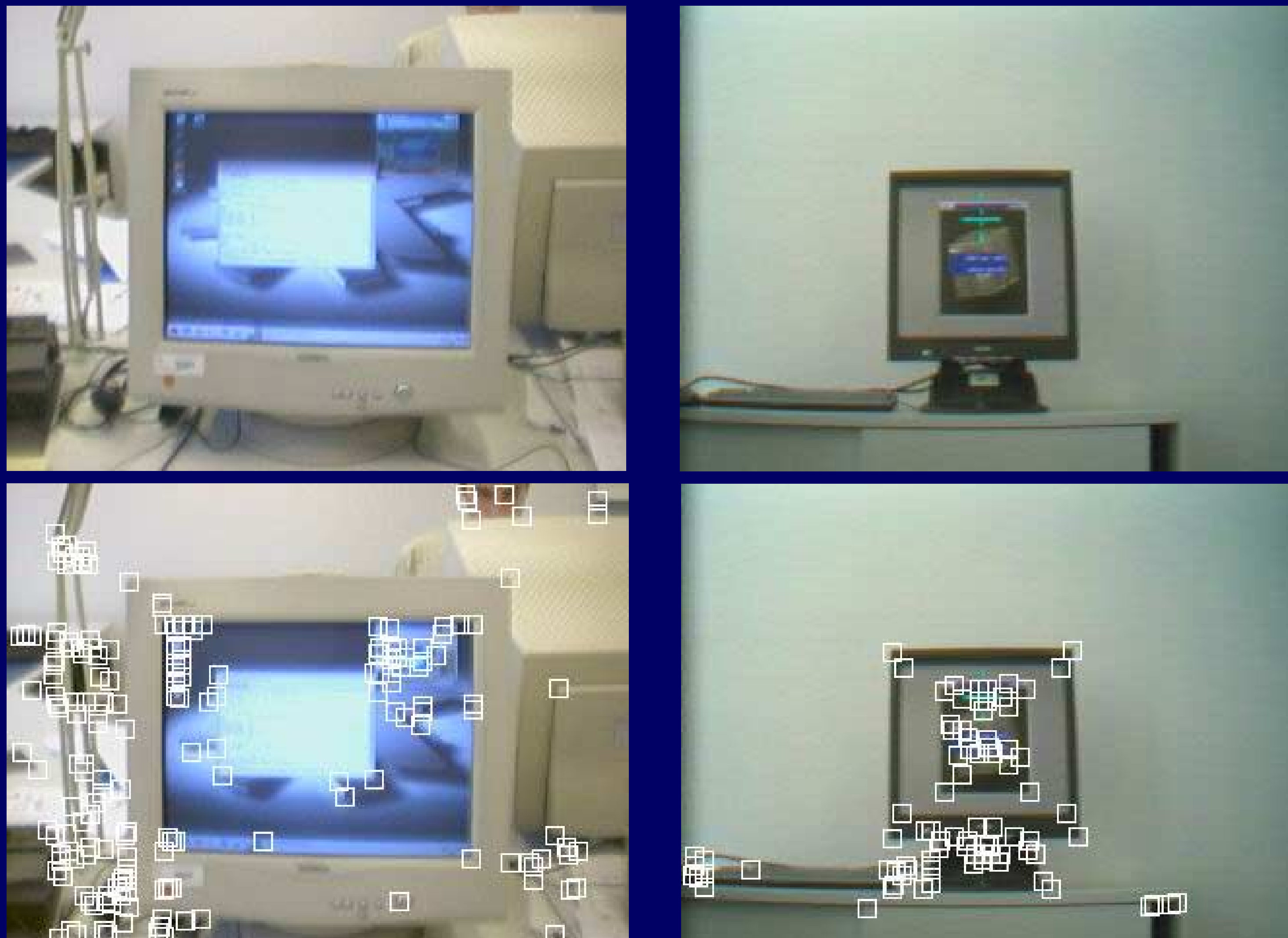
Algorithm Tests

- Accuracy tests (white screen and black screen)
- Stability tests (rotated frames)
- Sequence test (continuous frames)

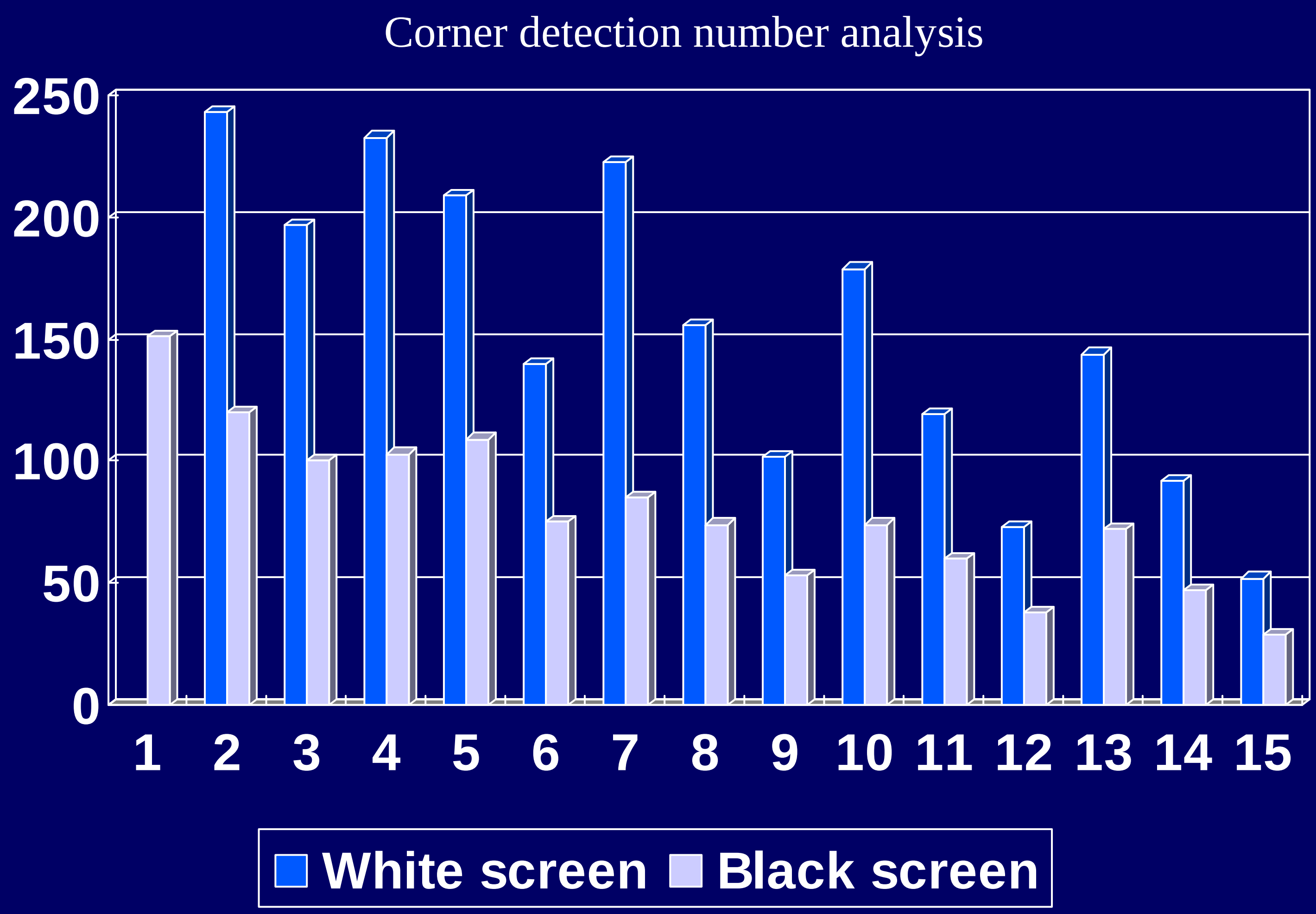
A PHIILIPS brilliance 109MP
white computer screen

A PHILIPS brilliance 180P2
black LCD computer screen

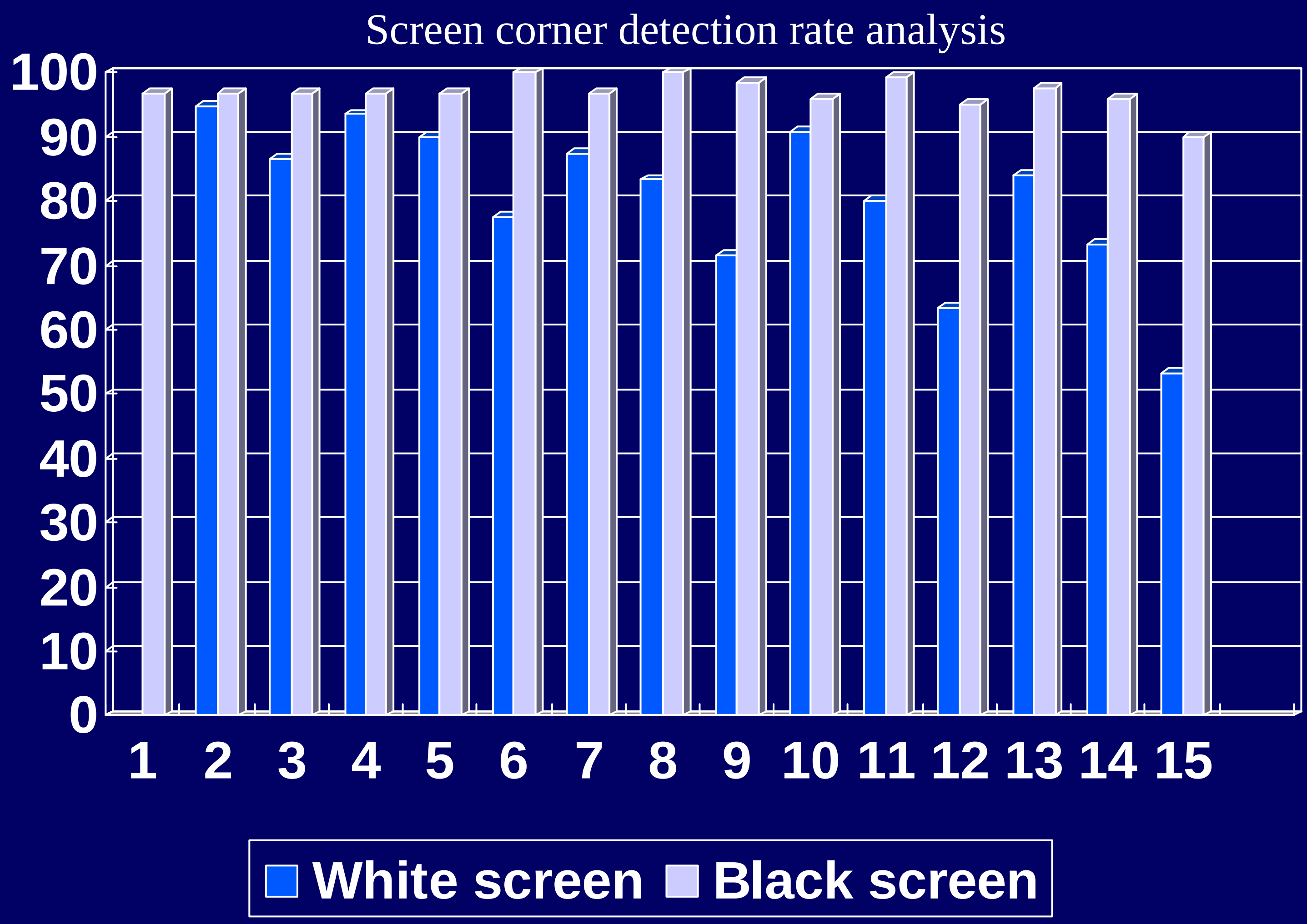
Algorithm Test Results



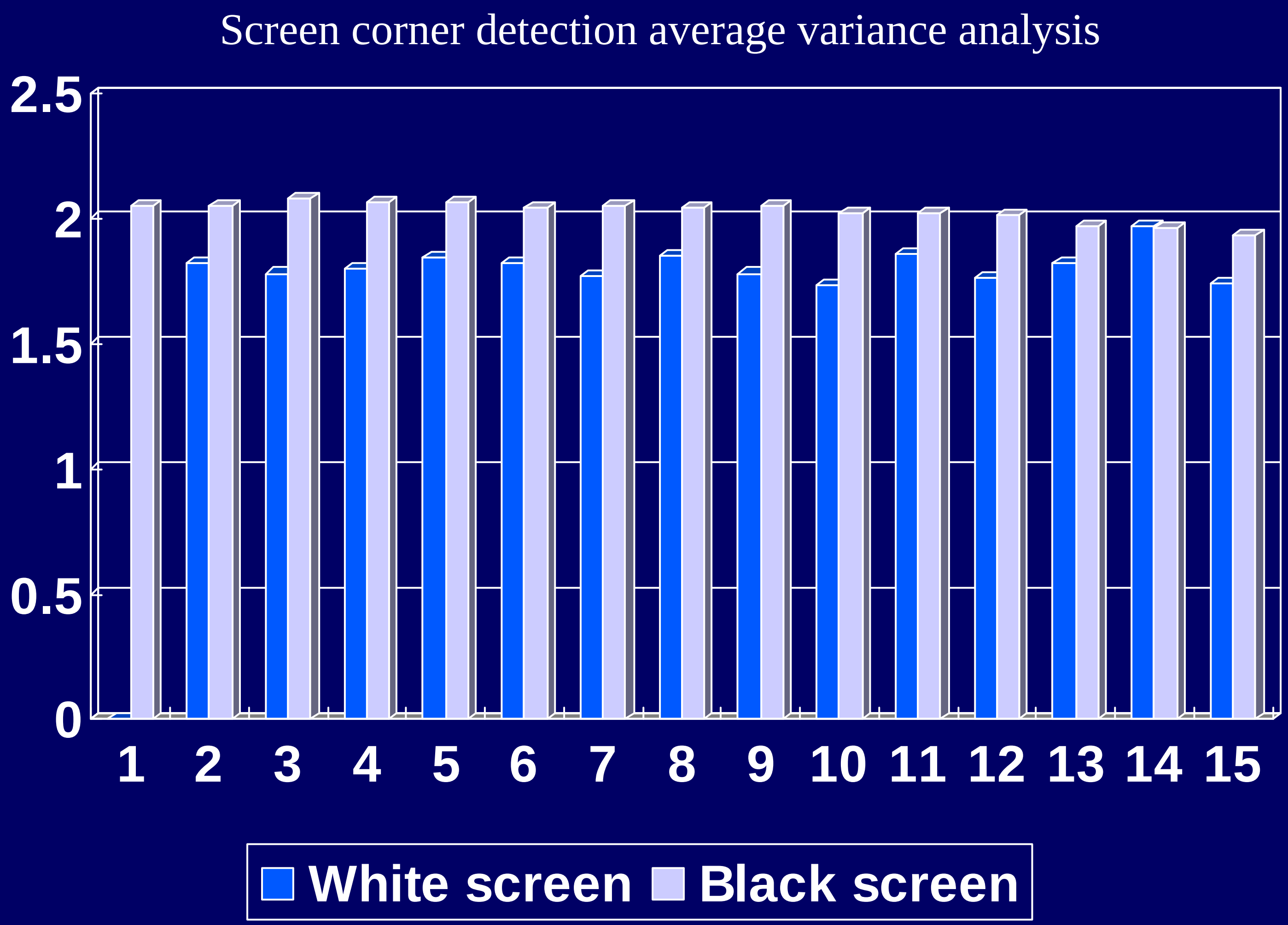
Algorithm Test Results Analysis I



Algorithm Test Results Analysis II

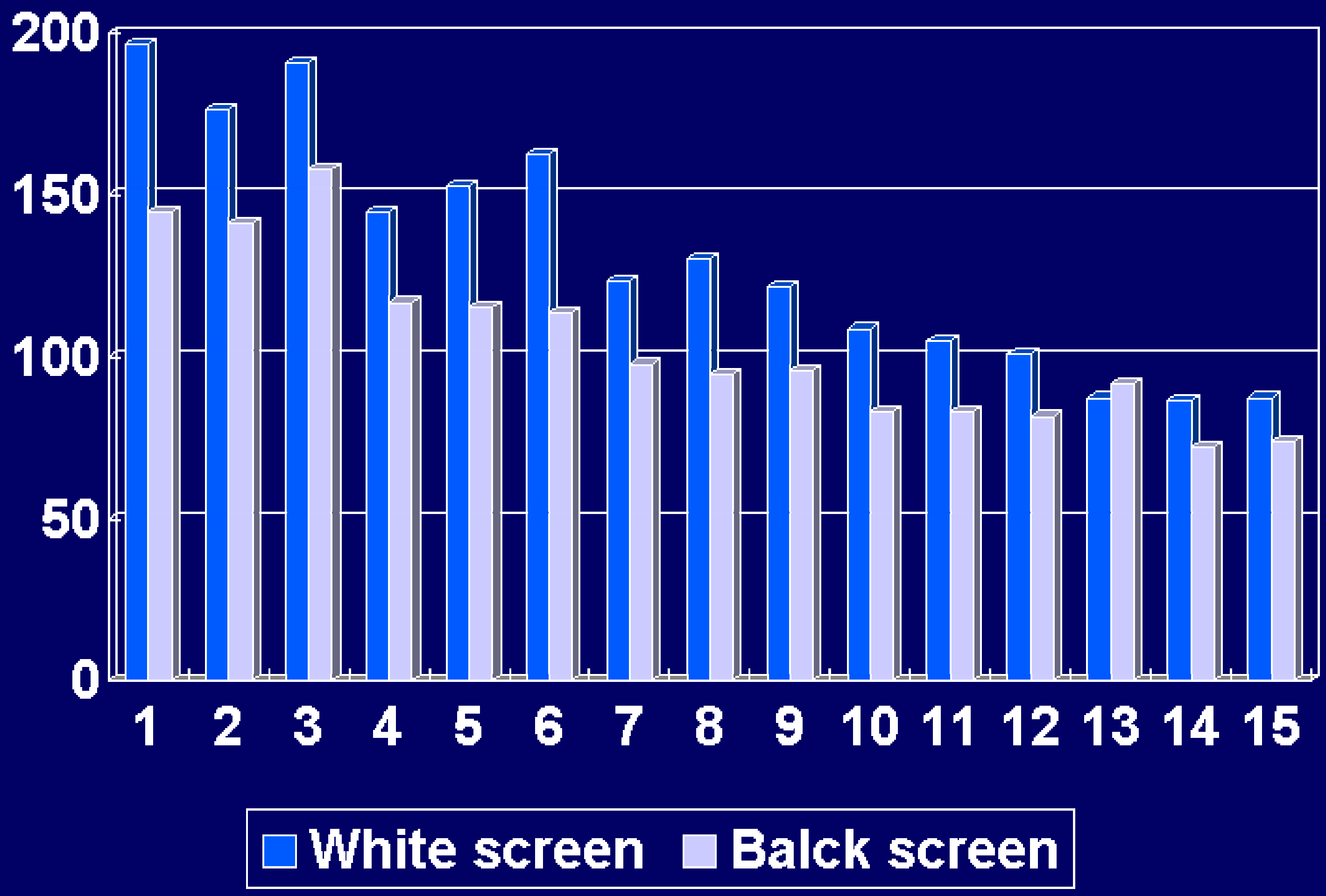


Algorithm Test Results Analysis III

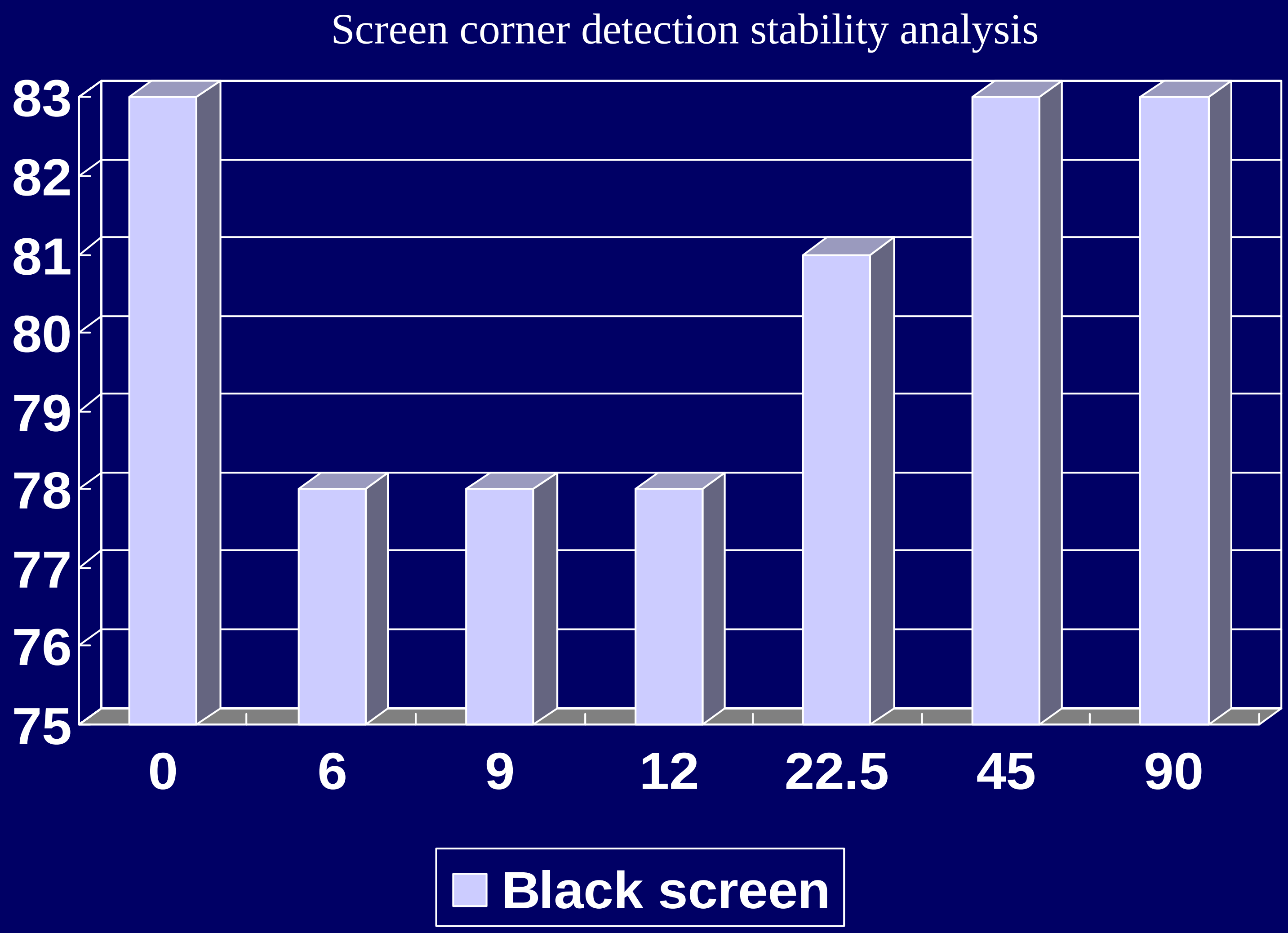


Algorithm Test Results Analysis IV

Screen corner detection speed analysis



Algorithm Test Results Analysis V



Test Results Conclusions

Attentions:

- Image quality affect the detection results
- Parameters affect the detection results

Good test result:

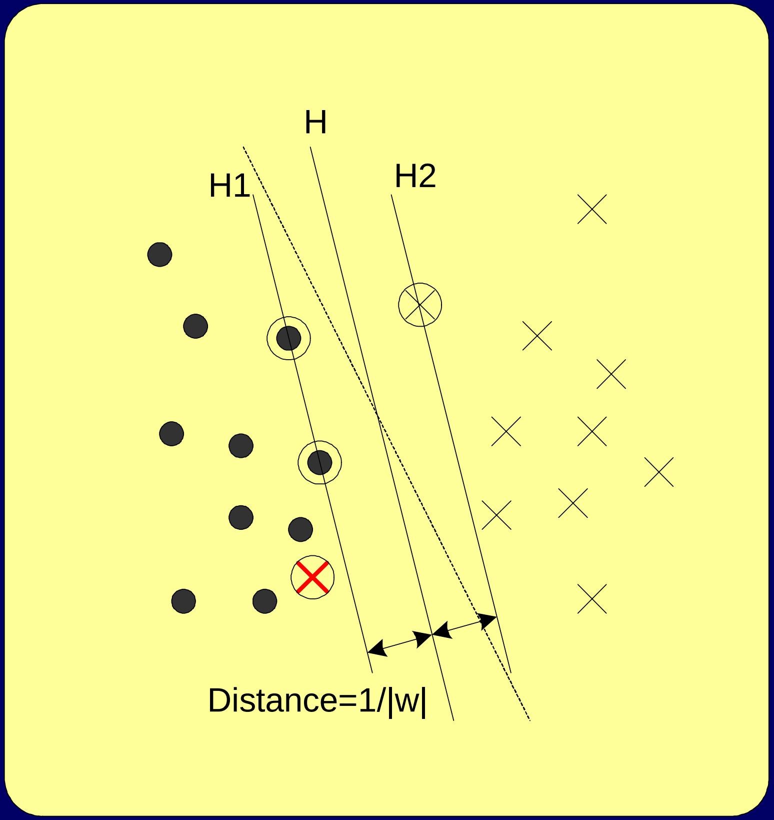
- Accurate
- Stable
- Fast

RVM Corner Classification Model

Support Vector Machine

- Finds the hyperplane leaving the largest possible fraction of points of the same class on the same side, while maximizing the distance of either class from the hyperplane.

- Disadvantages: No probability output; Support vectors grows with training set; Mercer's condition;



Relevance Vector Machine Theory

- General model in supervised learning

Samples: input vectors $\{\mathbf{x}_{\mathrm{n}}\}_{\mathrm{n}=1}^{N}$ and targets $\{t_n\}_{n=1}^{N}$

$$y(\mathbf{x};\mathbf{w}) = \sum_{i=1}^{M} w_i \phi_i(\mathbf{x}) = \mathbf{w}^{\mathrm{T}} \boldsymbol{\varphi}(\mathbf{x})$$

where,

$$\phi(\mathbf{x}) = \left(\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \ldots, \phi_M(\mathbf{x})\right)^{\mathrm{T}}$$

$$\mathbf{w} = (w_1, w_2, \ldots, w_M)^{\mathrm{T}}$$

Relevance Vector Machine Theory

- Probabilistic formulation for regression case

$t_n = y(\mathbf{x}_n;\mathbf{w}) + \varepsilon_n$, with noise: $p(t_n \mid \mathbf{x}) = N(t_n \mid y(\mathbf{x}_n), \sigma^2)$

likelihood:

$$p(\mathbf{t} \mid \mathbf{w}, \sigma^2) = (2\pi\sigma^2)^{-N/2} \exp\left\{ -\frac{1}{2\sigma^2} \left\| \mathbf{t} - \mathbf{\Phi}\mathbf{w} \right\|^2 \right\}$$

where, $\mathbf{\Phi} = [\phi(\mathbf{x}_1), \phi(\mathbf{x}_2), \ldots, \phi(\mathbf{x}_N)]^T$

$$\phi(\mathbf{x}_n) = [1, K(\mathbf{x}_n, \mathbf{x}_1), K(\mathbf{x}_n, \mathbf{x}_2), \ldots, K(\mathbf{x}_n, \mathbf{x}_N)]^T$$

Relevance Vector Machine Theory

- Hyperparameters to avoid over-fitting (no weight penalty term)

$$p(\mathbf{w} \mid \boldsymbol{\alpha}) = \prod_{i=0}^{N} N(w_i \mid 0, \alpha^{-1}),$$

$$p(\boldsymbol{\alpha}) = \prod_{i=0}^{N} \mathrm{Gamma}(\alpha_i \mid a, b), \qquad p(\sigma^{-2}) = \mathrm{Gamma}(\beta \mid c, d)$$

$$with,\ \beta \equiv \sigma^2$$

Relevance Vector Machine Theory

- Prediction in regression case

$$p(t_* \mid \mathbf{t}) = \int p(t_* \mid \mathbf{w}, \boldsymbol{\alpha}, \sigma^2)\, p(\mathbf{w}, \boldsymbol{\alpha}, \sigma^2 \mid \mathbf{t})\, d\mathbf{w}\, d\boldsymbol{\alpha}\, d\sigma^2$$

$$p(t_* \mid \mathbf{t}, \boldsymbol{\alpha}_{MP}, \sigma^2_{MP}) = \int p(t_* \mid \mathbf{w}, \sigma^2_{MP})\, p(\mathbf{w} \mid \mathbf{t}, \boldsymbol{\alpha}_{MP}, \sigma^2_{MP})\, d\mathbf{w}$$

$$p(t_* \mid \mathbf{t}, \boldsymbol{\alpha}_{MP}, \sigma^2_{MP}) = N(t_* \mid y_*, \sigma^2_*)$$

Relevance Vector Machine Theory

- Prediction in regression case

$$p(t_* \mid \mathbf{t}, \boldsymbol{\alpha}_{MP}, \sigma_{MP}^2) = N(t_* \mid y_*, \sigma_*^2)$$

with,

$$y_* = \boldsymbol{\mu}^T \phi(\mathbf{x}_*)$$

$$\sigma_*^2 = \sigma_{MP}^2 + \phi(\mathbf{x}_*)^T \boldsymbol{\Sigma} \phi(\mathbf{x}_*)$$

$$\boxed{\begin{aligned} \boldsymbol{\Sigma} &= (\sigma^{-2}\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \mathbf{A})^{-1} \\ \boldsymbol{\mu} &= \sigma^{-2}\boldsymbol{\Sigma}\boldsymbol{\Phi}^{\mathrm{t}}\mathbf{t} \\ \mathbf{A} &= \mathrm{diag}(\alpha_0, \alpha_1, \ldots, \alpha_N) \end{aligned}}$$

Relevance Vector Machine Theory

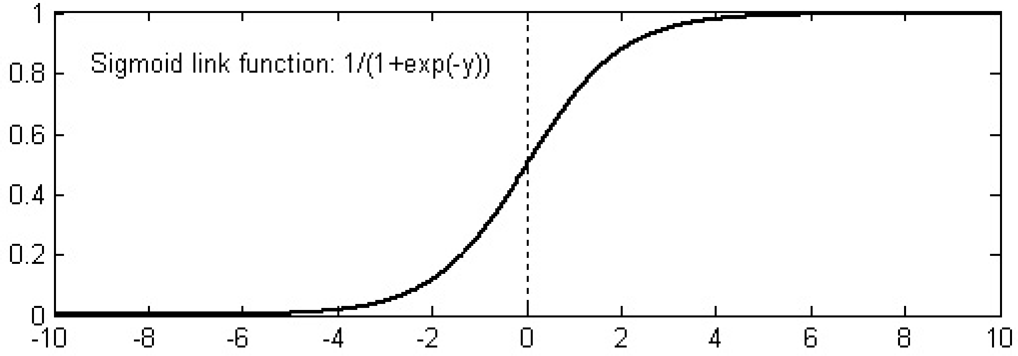
- Prediction in classification case

Likelihood changes to:

$$P(\mathbf{t} \mid \mathbf{w}) = \prod_{n=1}^{N} \sigma\{y(\mathbf{x}_n; \mathbf{w})\}^{t_n} [1 - \sigma\{y(\mathbf{x}_n; \mathbf{w})\}]^{1-t_n}$$

with,

$$\sigma(y) = 1/(1 + e^{-y})$$



Relevance Vector Machine Theory

- Prediction in classification case

$$y_* = \mathbf{w}_{\mathrm{MP}}{}^T \phi(\mathbf{x}_*)$$

$$\boldsymbol{\Sigma} = (\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{B}\boldsymbol{\Phi} + \mathbf{A})^{-1}$$

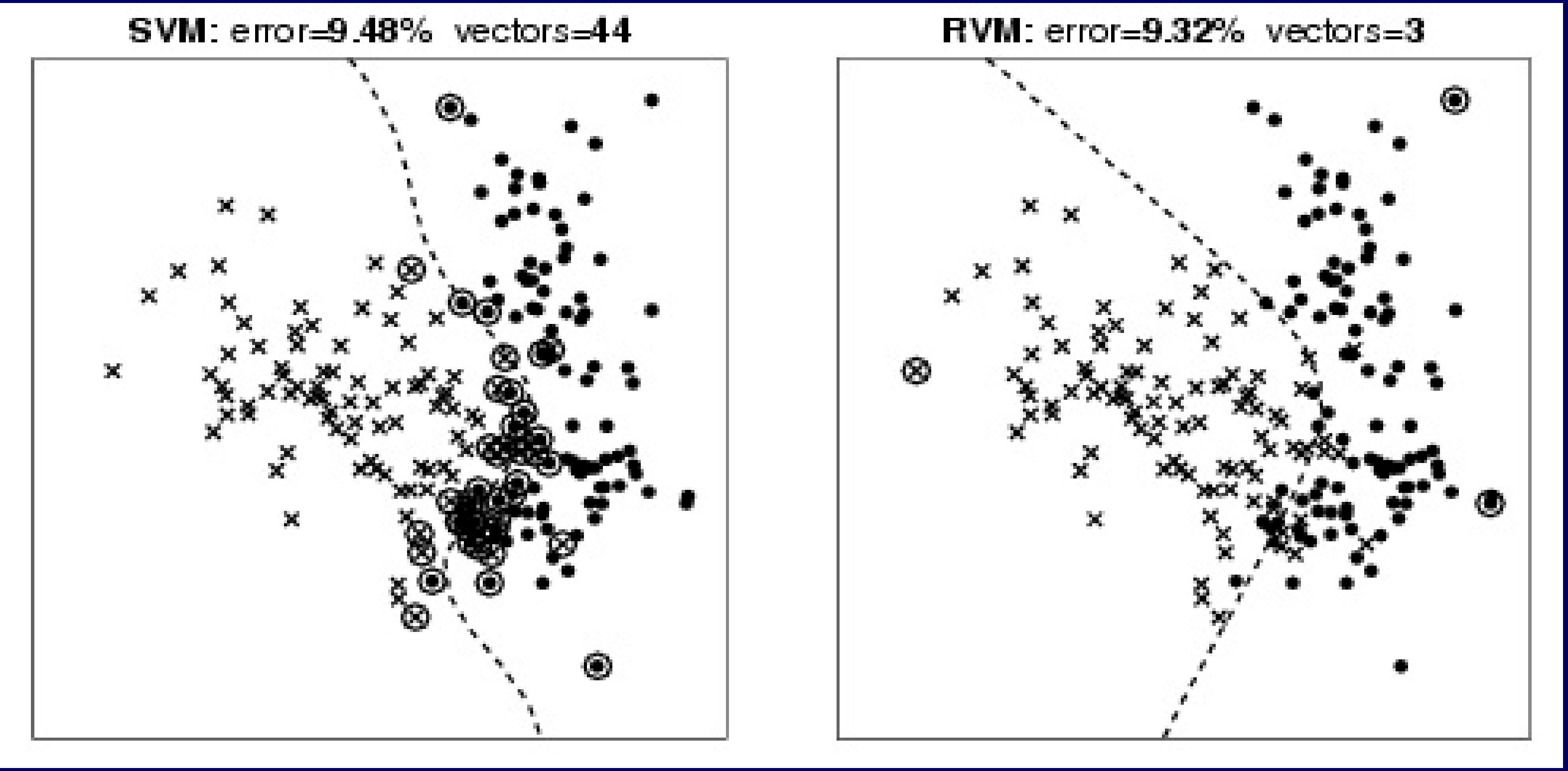
$$\mathbf{w}_{\mathrm{MP}} = \boldsymbol{\Sigma}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{B}\mathbf{t}$$

$$\mathbf{B} = diag(\beta_1, \beta_2, \ldots, \beta_N)$$

$$\beta_n = \sigma\{y(\mathbf{x}_n)\}[1 - \sigma\{y(\mathbf{x}_n)\}]$$

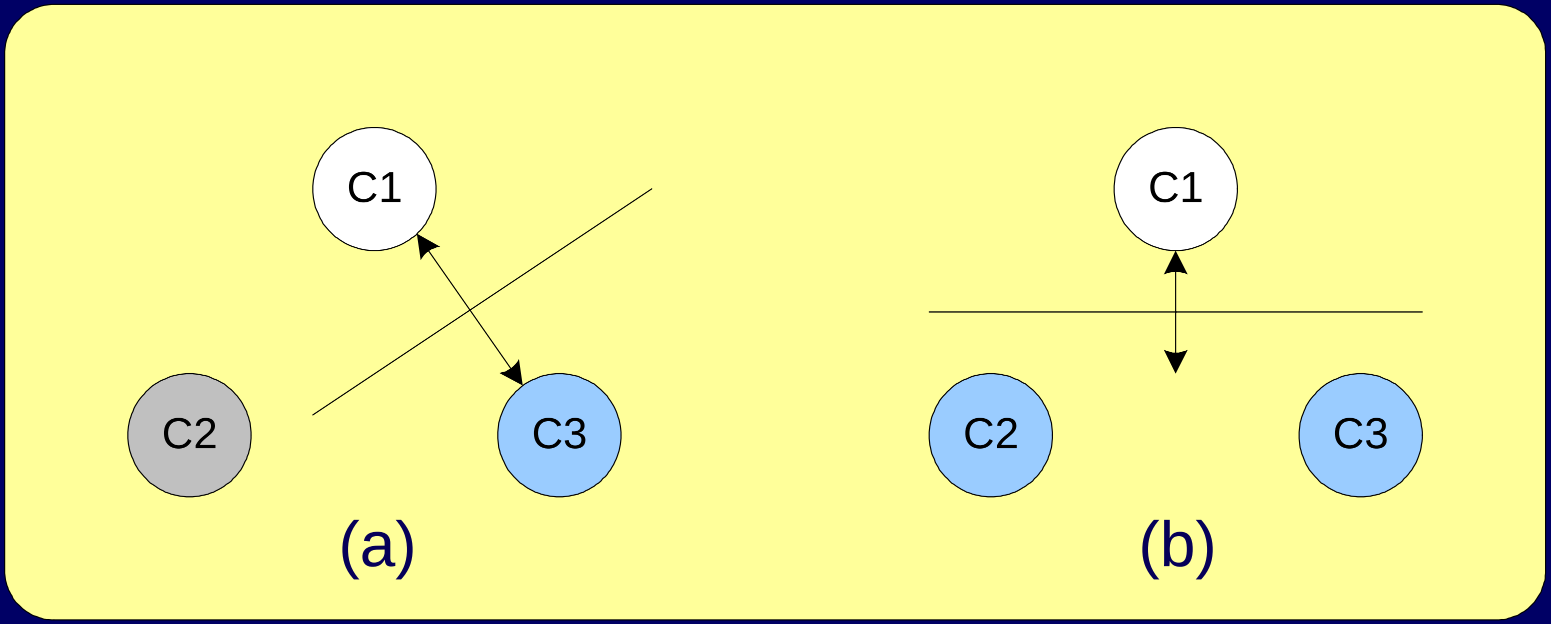
Relevance Vector Machine Theory

- Comparison between SVM and RVM



Multi-class Classification (K classes)

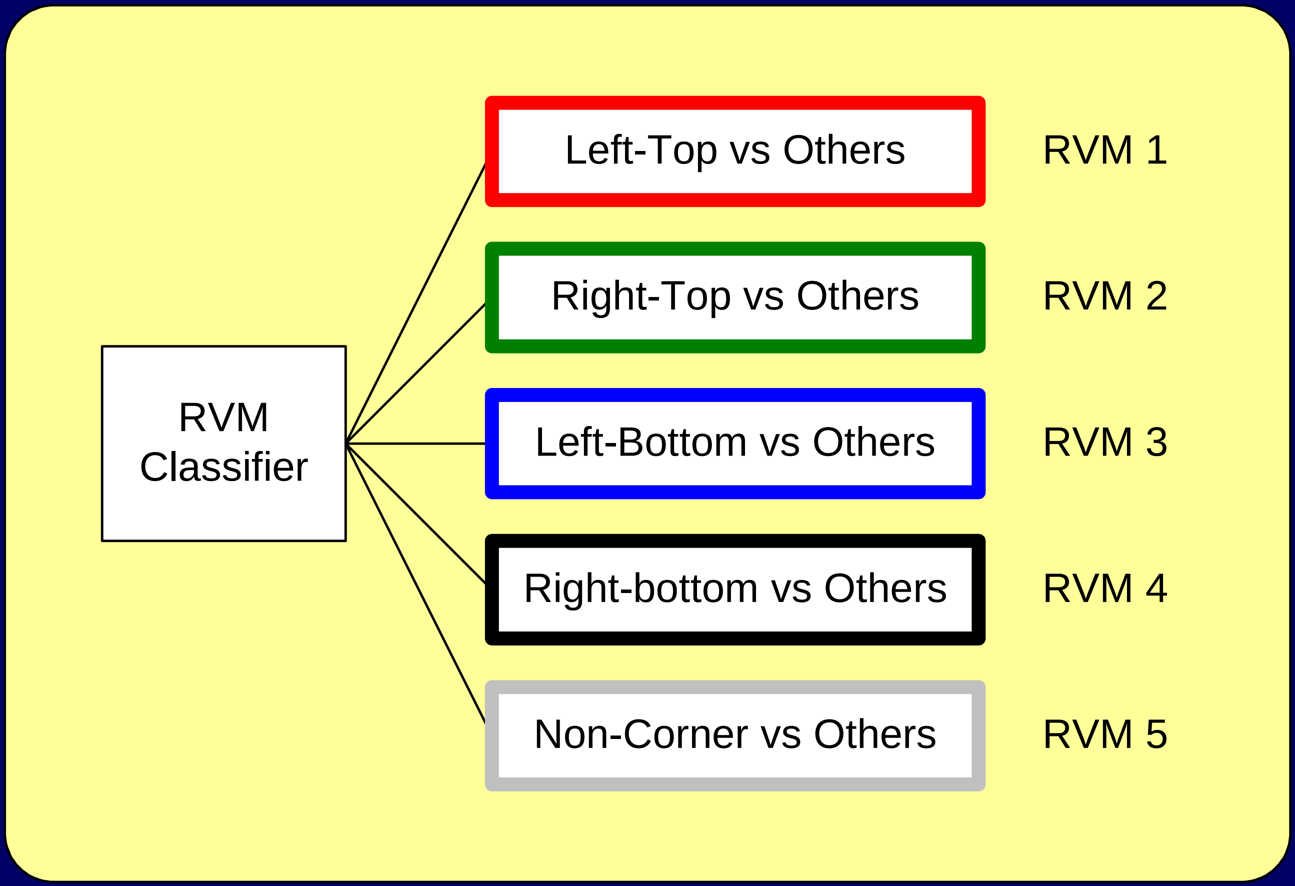
- Pair-wise, K*(K-1)/2 RVMs
- One-versus-others, K RVMs



Pair-wise is not more accurate than One-versus-others

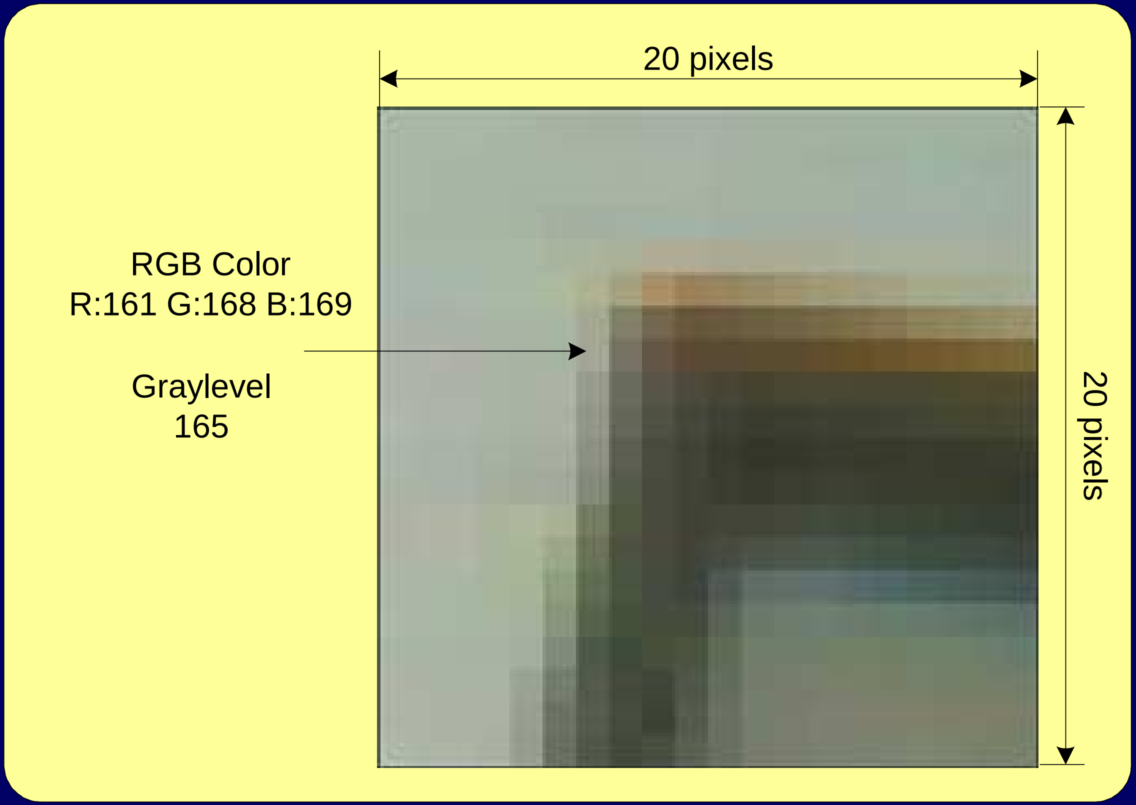
Multi-class Classification (K classes)

- Five RVM models used in corner classification



Feature Extraction

- Intensity values as features



Dimension is too big and might be changed

DCT Coefficients as Feature Vectors

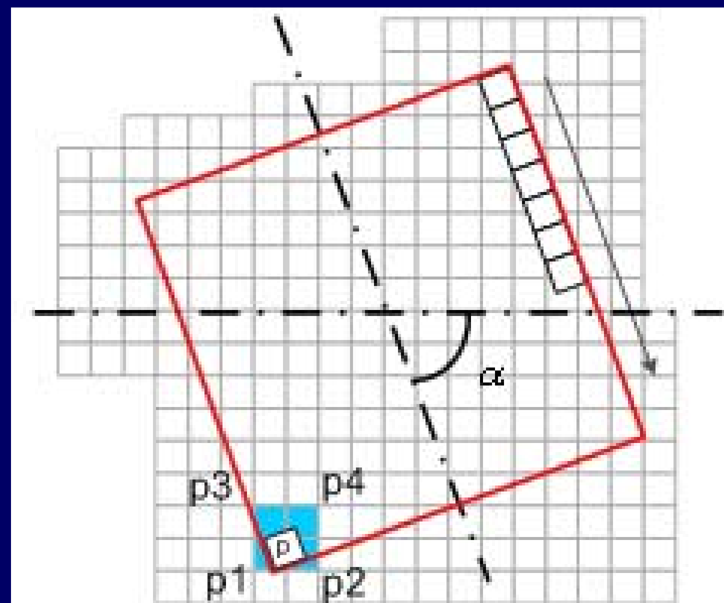
- Independence with sample size
- Dimension become much smaller

v \ u	0	1	2	3	4	...	N
0	4.96	1.36	-0.48	-0.14	...		
1	1.17	0.72	-0.19	...			
2	-0.47	-0.29	...				
3	-0.03	...					
4	...						
⋮							
M							

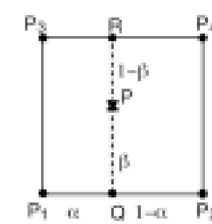
Feature Vector:
[4.96, 1.36, -0.48, -0.14, 1.17, 0.72, -0.19, -0.47, -0.29, -0.03]

Feature Extraction on Rotated Images

- Rotate target window



$$d_P(\alpha,\beta) = d_1(1-\alpha)(1-\beta) + d_2\alpha(1-\beta) + d_3(1-\alpha)\beta + d_4\alpha\beta$$



Sample Dataset Selection

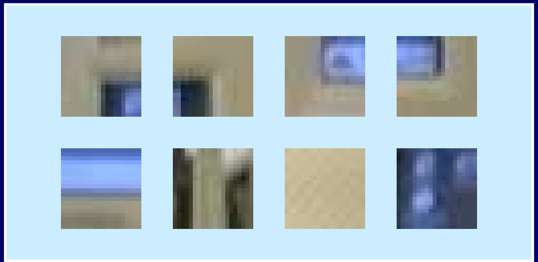
- Manually selected sample dataset
- Synthetic sample datasets
- Adaptive sample datasets

Sample Dataset Selection

- Sample dataset specification

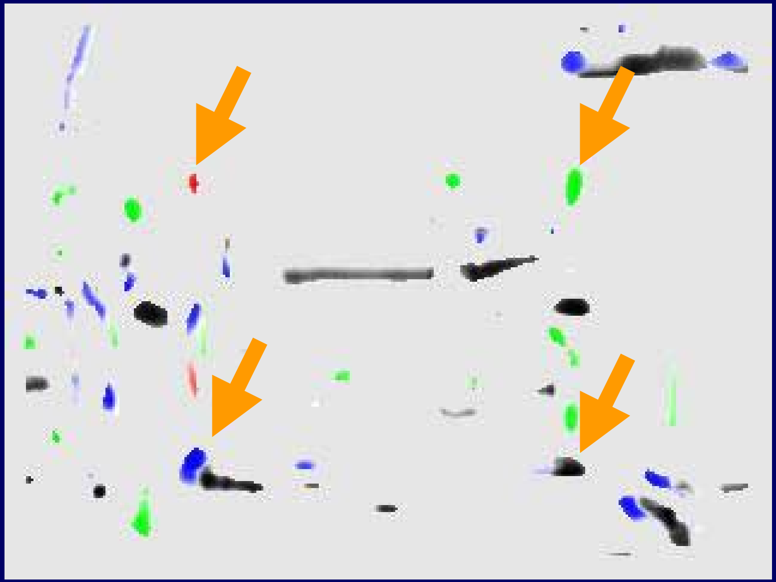
Sample Size	20 x 20 pixels
Number of Sample Class	5
Number of Samples	80
Samples Distribution	10/10/10/10/40

- Corner samples:



Sample Dataset Selection

- Classification results

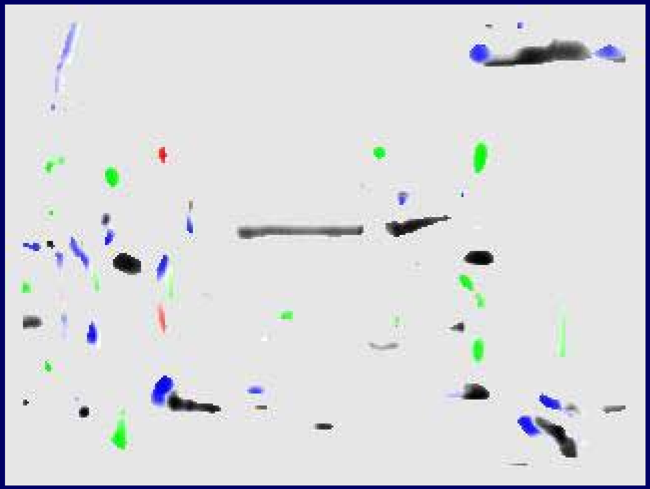


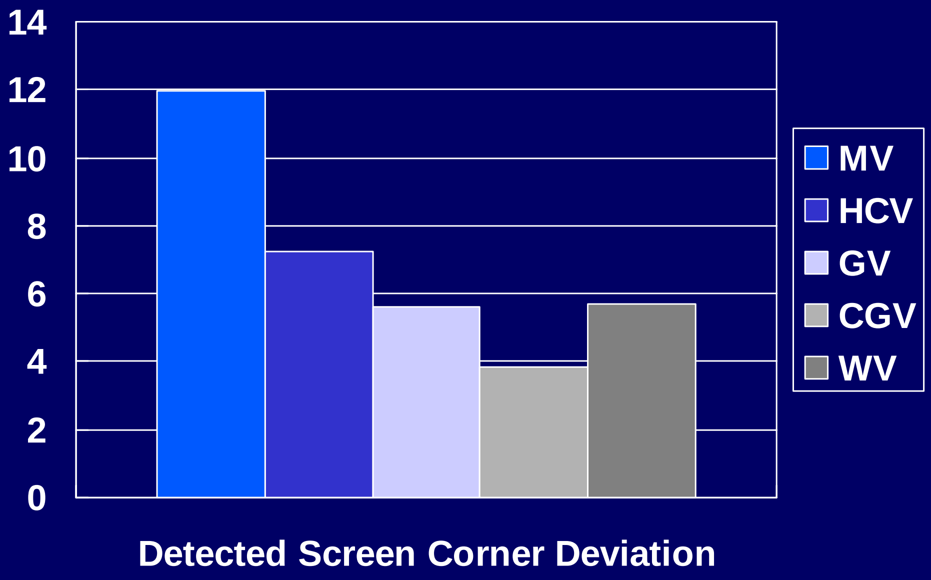
RVM		Error Analysis Utility Outputs						
KF	RVN	HCV	MV	GV	CGV	MUR	CD	WV
L-2.0	18.4	7.26	11.99	5.62	3.85	93%	98%	5.71

Sample Dataset Selection

- Classification results

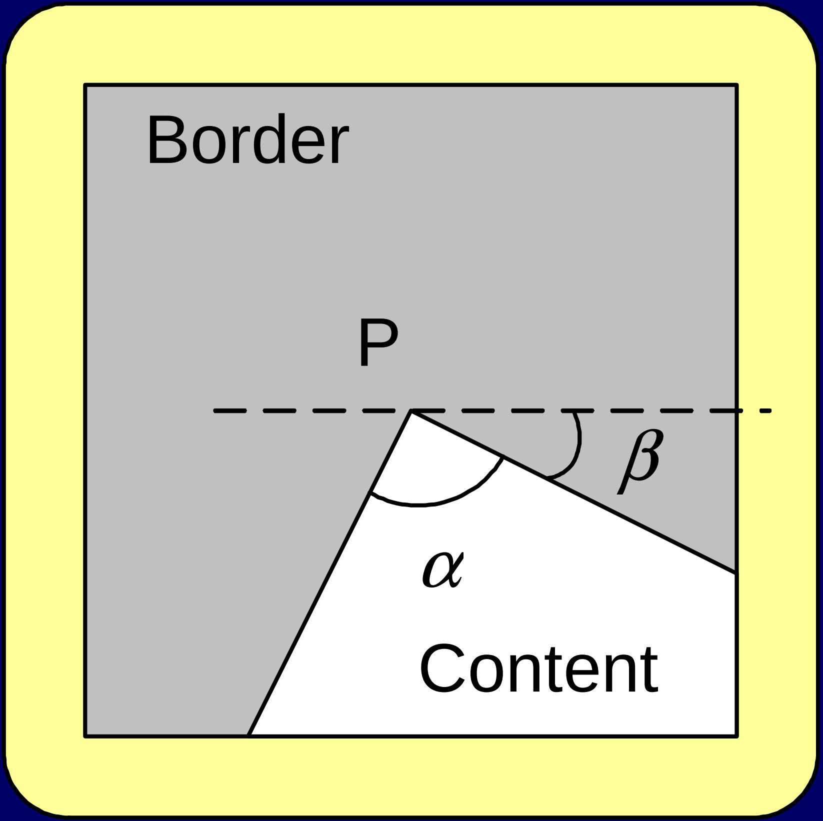
1. Probability output is effective
2. Misunderstanding Rate is high





Synthetic Sample Dataset

- Generation

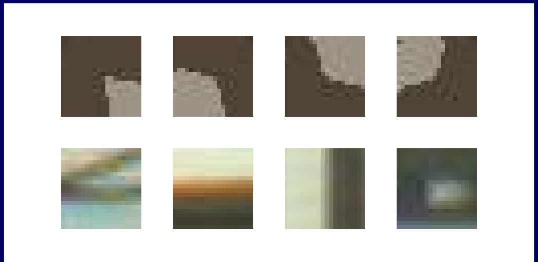


Synthetic Sample Dataset

- Sample dataset specification

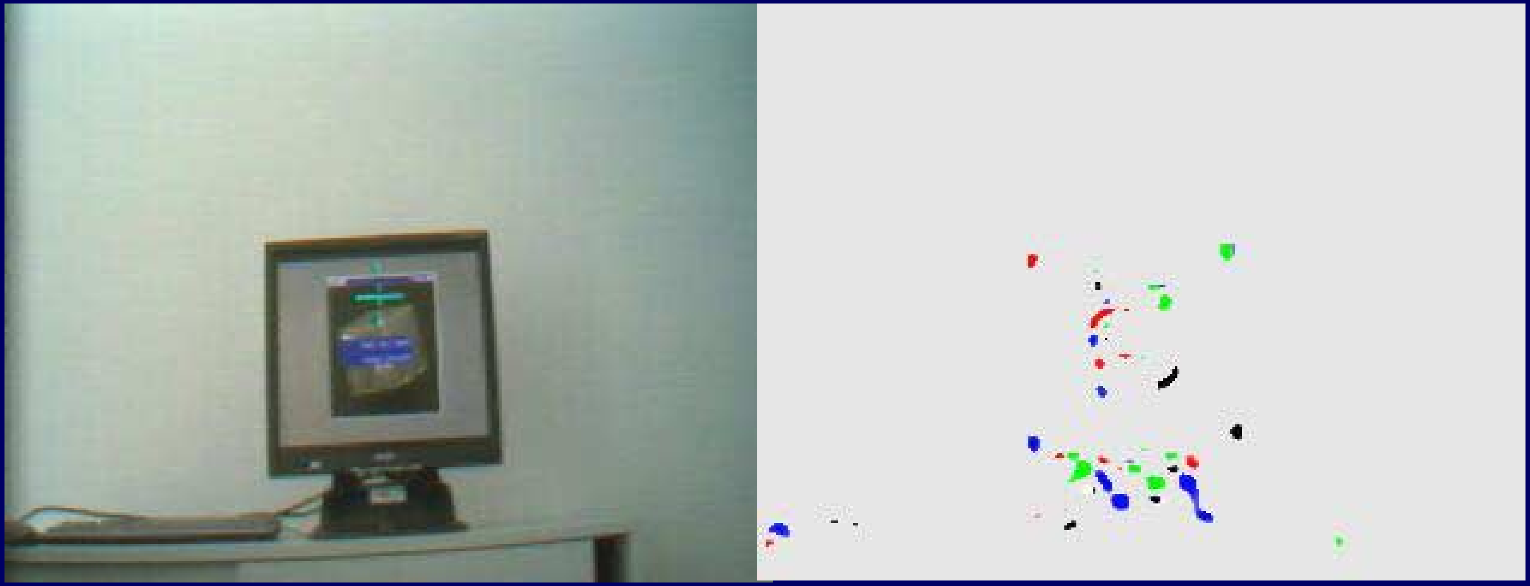
Sample Size	10 x 10 pixels
Number of Sample Class	5
Number of Samples	100
Samples Distribution	10/10/10/10/60

- Corner samples:



Synthetic Sample Dataset

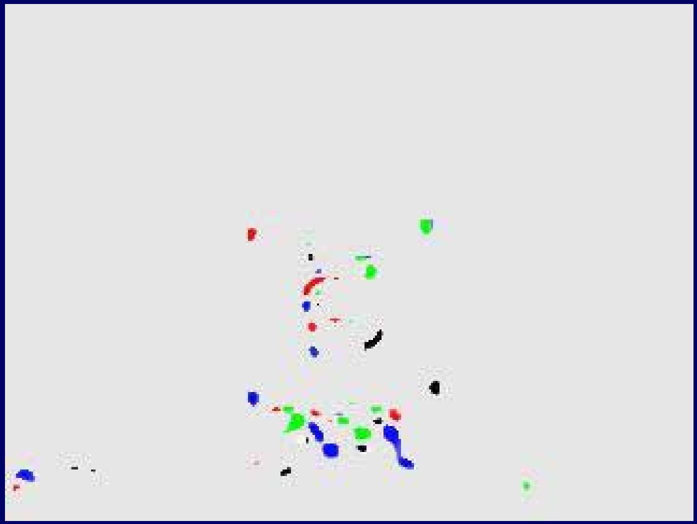
- Classification results

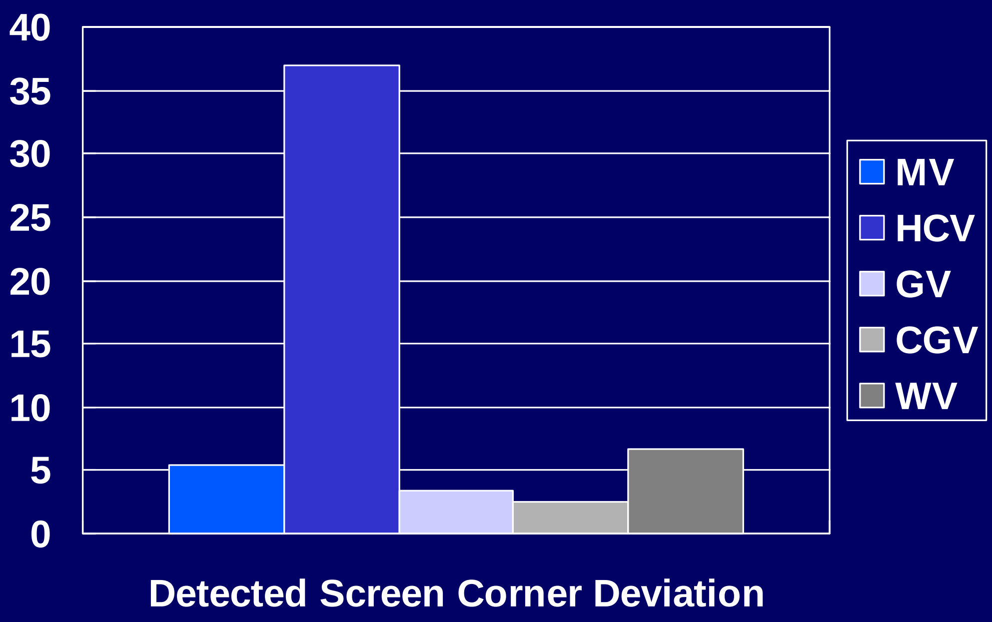


Synthetic Sample Dataset

- Classification results

RVM		Error Analysis Utility Outputs						
KF	RVN	HCV	MV	GV	CGV	MUR	CD	WV
G-0.5	**4.4**	36.94	5.38	3.47	2.57	84.%	100%	6.65





Adaptive Sample Dataset

- Sample dataset specification

Sample Size	40 x 40 pixels
Number of Sample Class	5
Number of Samples	108
Samples Distribution	10/10/10/10/68

Adaptive Sample Dataset

- Non-corner samples selection (three categories)

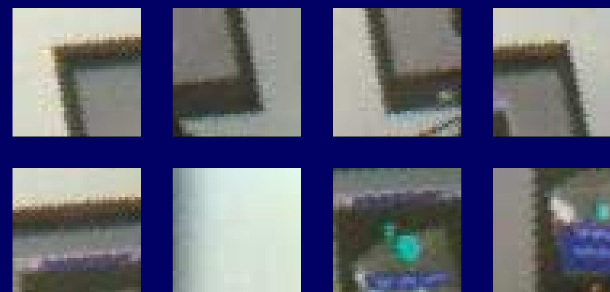
Base: The samples that can be miss-classified

Background: The samples far from corners

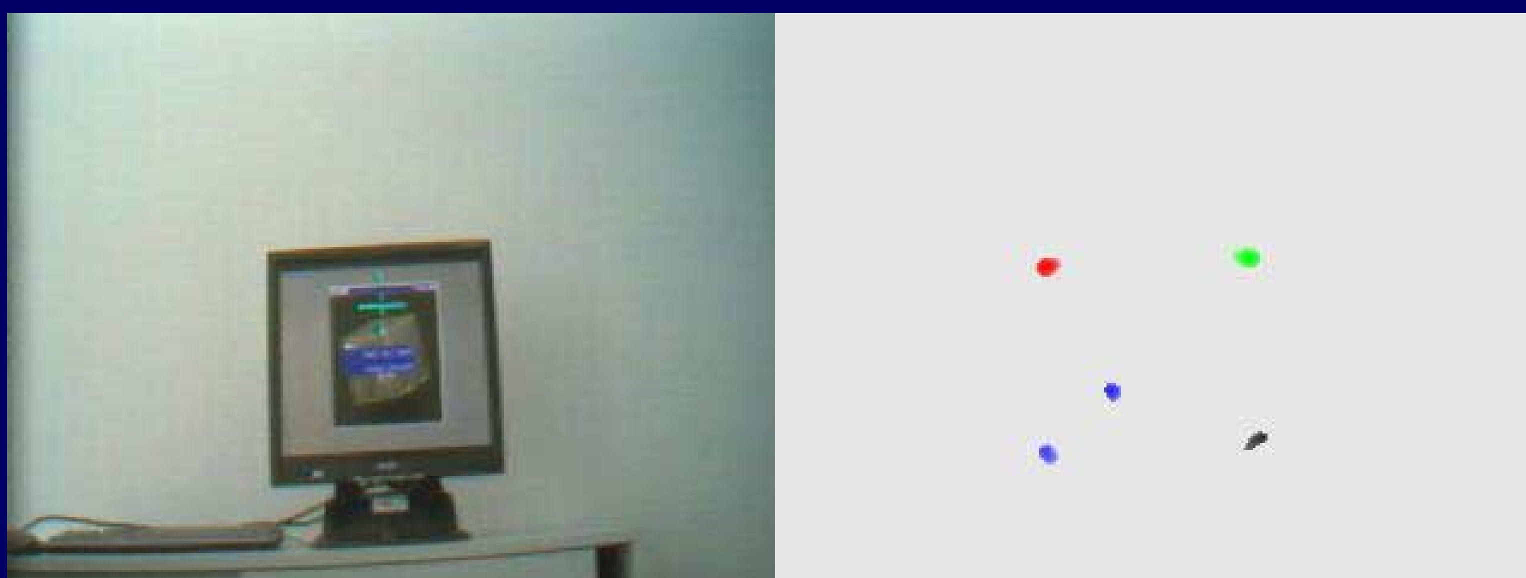
Adaptive: The samples miss-classified before

Adaptive Sample Dataset

- Samples:



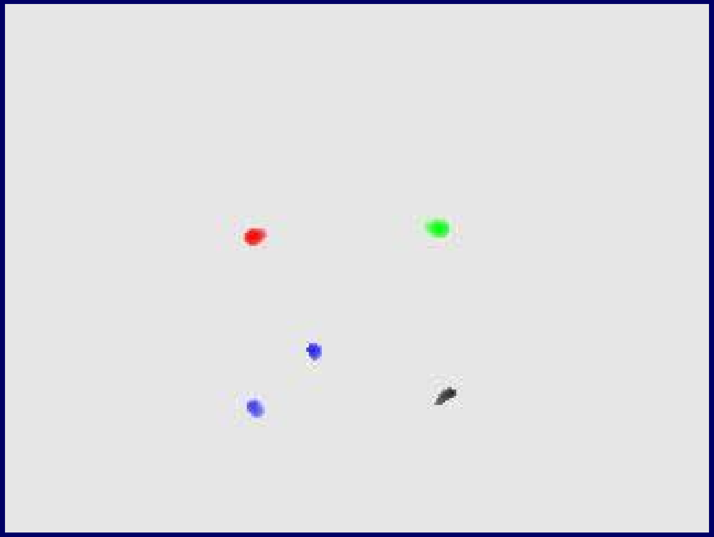
- Classification results

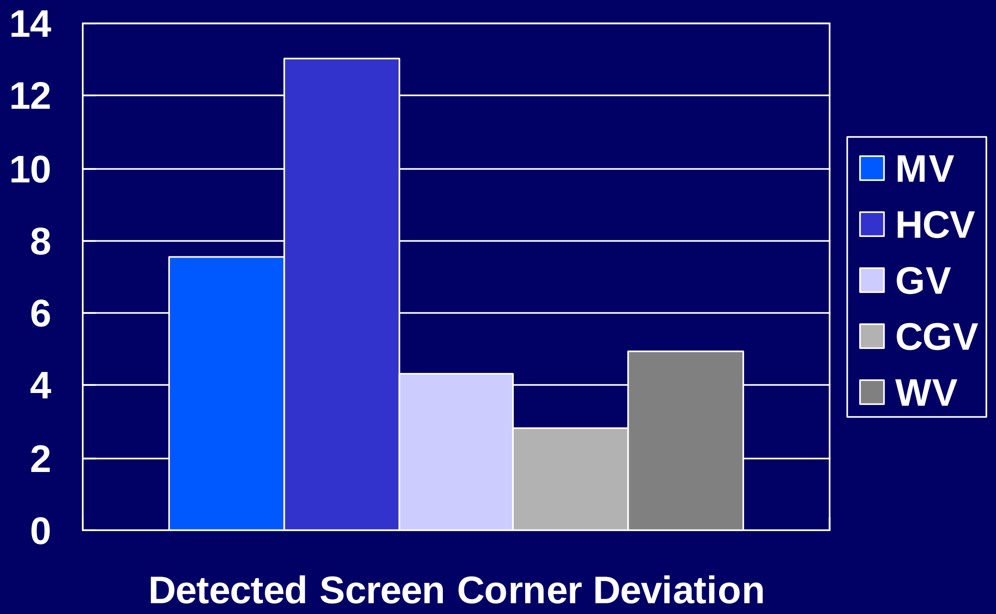


Adaptive Sample Dataset

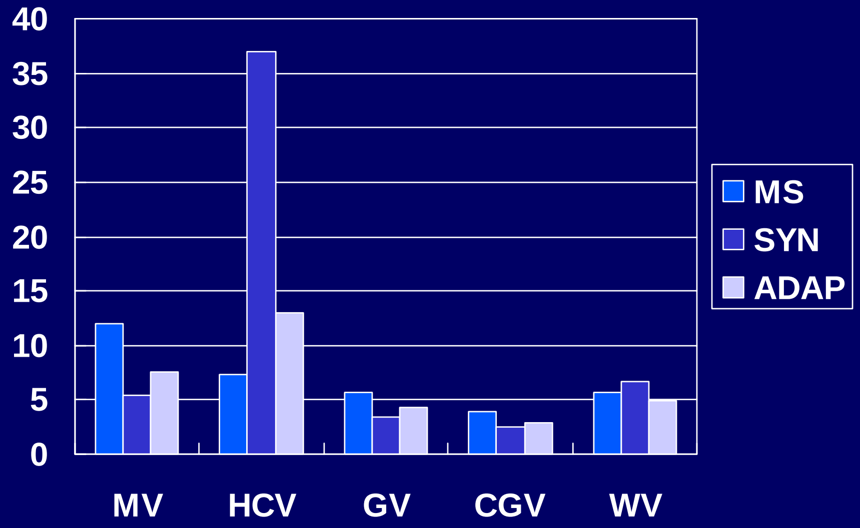
- Classification results

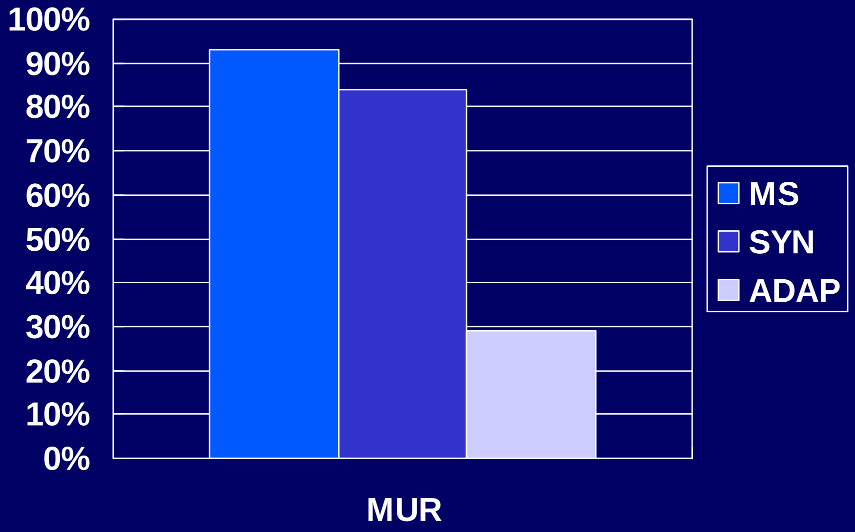
RVM		Error Analysis Utility Outputs						
KF	RVN	HCV	MV	GV	CGV	MUR	CD	WV
L-2.0	**15.6**	13.01	7.56	4.33	2.84	**29%**	100%	4.93

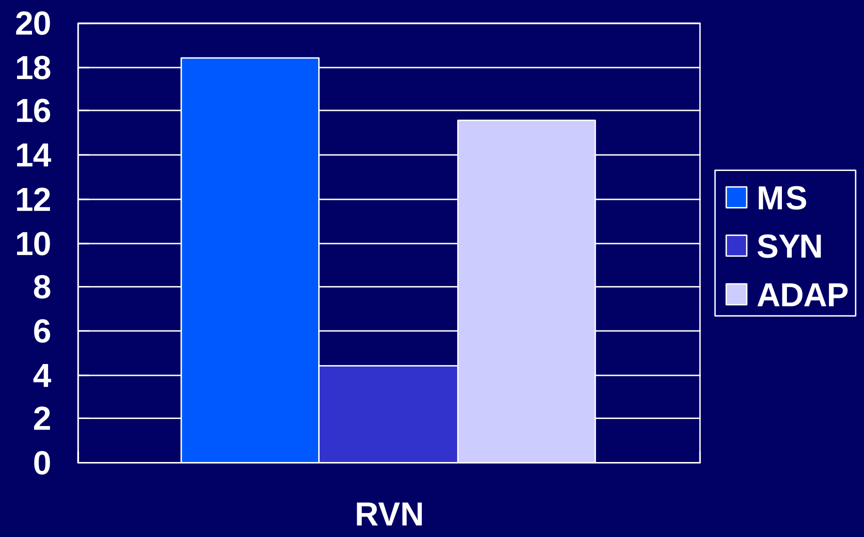




Sample datasets comparison







Sample datasets comparison

1. Manually selected dataset is not convenient for users
2. Adaptive dataset makes classification slow
3. Synthetic dataset is suitable for current use

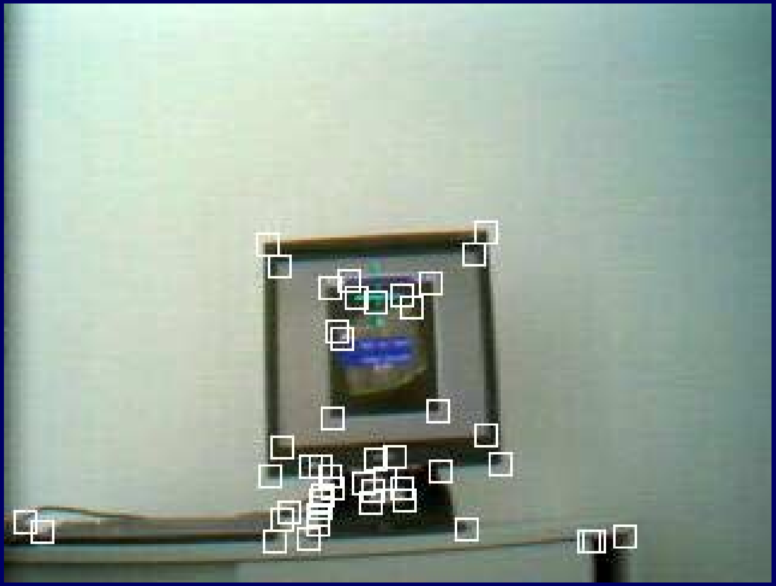
Feature extraction speed

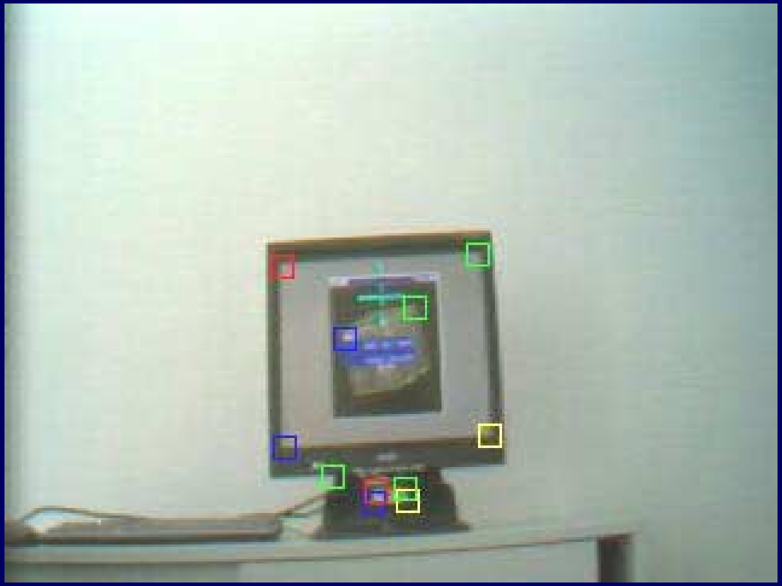
Target Size	10x10	20x20	40x40
Speed (s/sec)	718	199	49

So we used synthetic dataset in final system.

Rectangle Filter

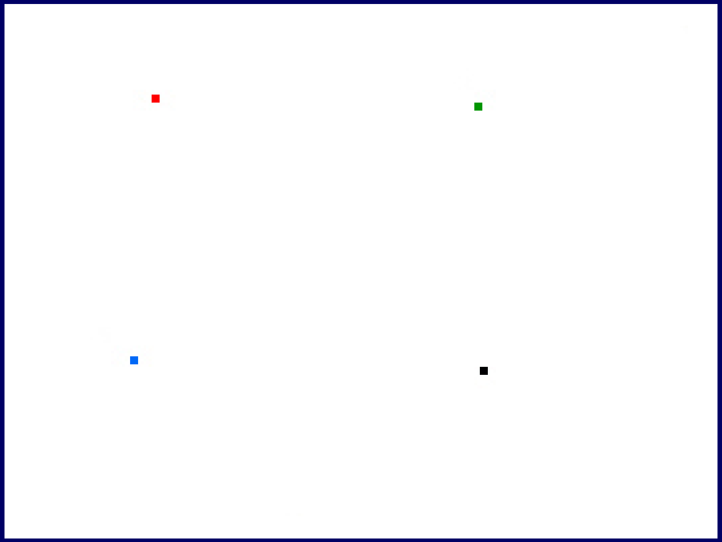
Candidates-Winners Result By far





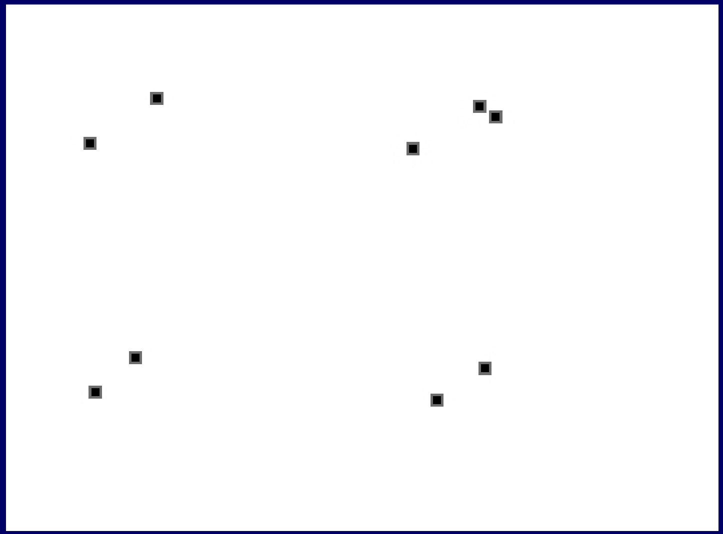
Rectangle Filter with Corner Type Information

- Select out the final screen corners



Rectangle Filter Before RVM

- Reduce candidates number



15 candidates decreases to 9

Results

Less than 1 millionsecond per sample

ROI Tracking

Purposes

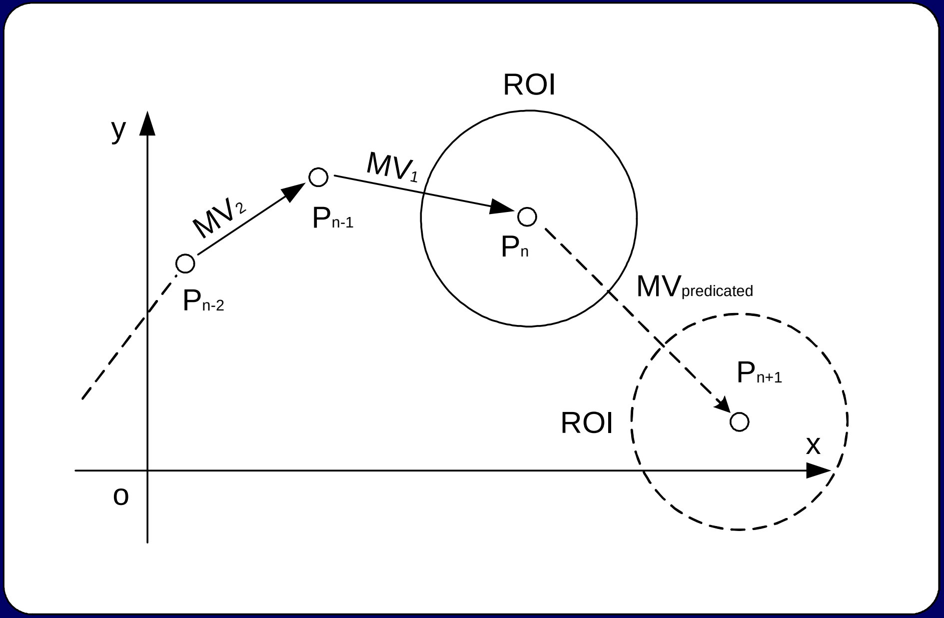
1. Accelerate detection speed
2. Store the trace information

ROI Tracking Filter

Cannot work well if move fast

ROI Prediction

- Simple motion predication

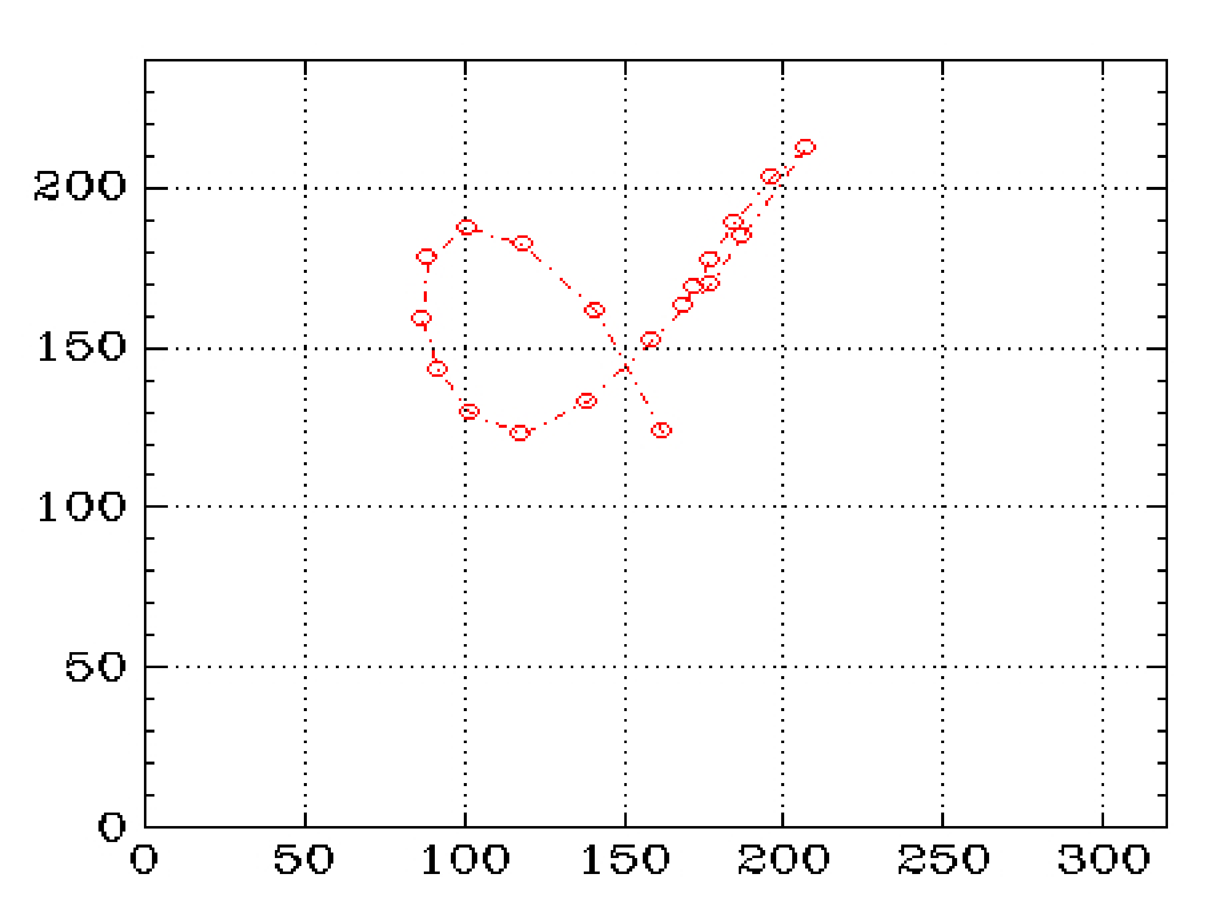


$$\|\mathbf{mv}_{predicted}\| = \|\mathbf{mv}_1\| * \|\mathbf{mv}_1\| / \|\mathbf{mv}_2\|$$

$$\alpha_{predicted} = \alpha_1 + w * (\alpha_1 - \alpha_2)$$

$$w = \begin{cases} \|\mathbf{mv}_2\| / \|\mathbf{mv}_1\|, & if \|\mathbf{mv}_2\| \le \|\mathbf{mv}_1\| \\ \|\mathbf{mv}_1\| / \|\mathbf{mv}_2\|, & if \|\mathbf{mv}_1\| \le \|\mathbf{mv}_2\| \end{cases}$$

ROI Prediction Results



Missing Corners Prediction and Alignment

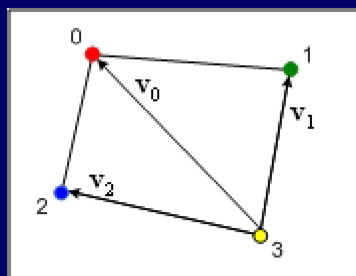
- Type I missing
  Without screen information

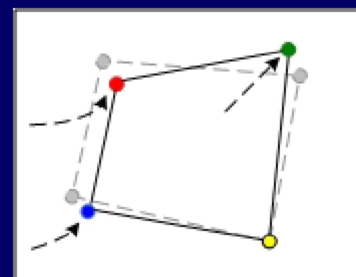
- Type II missing
  With screen detected in previous frames

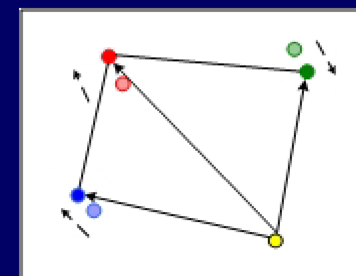
Type I missing seldom happens;
Need to do more work on Type II missing

Type II Missing

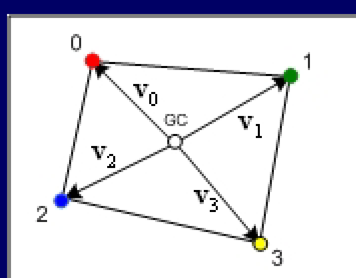
- Miss less than 4 corners

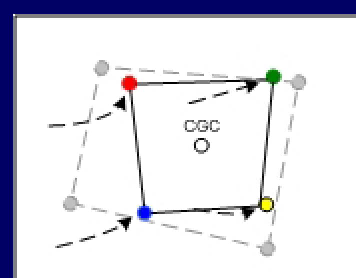


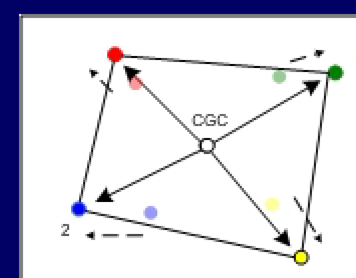




- Miss 4 corners



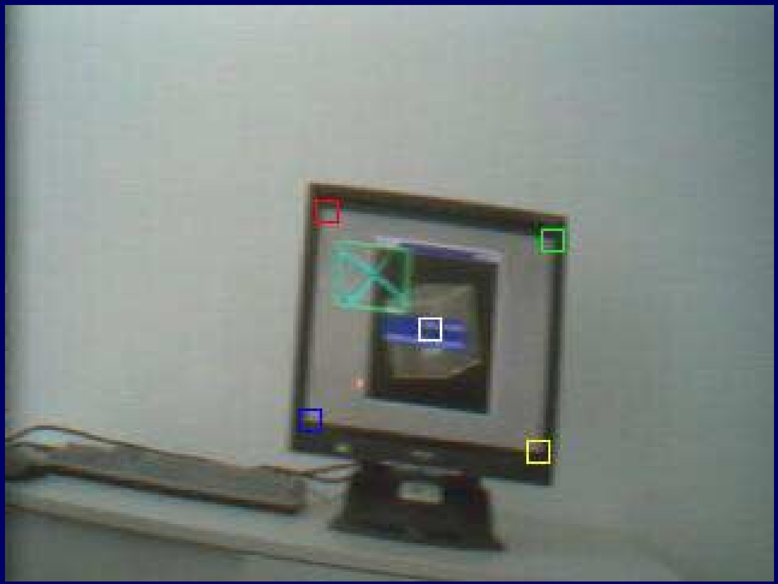




Improvement Results

Prediction without alignment

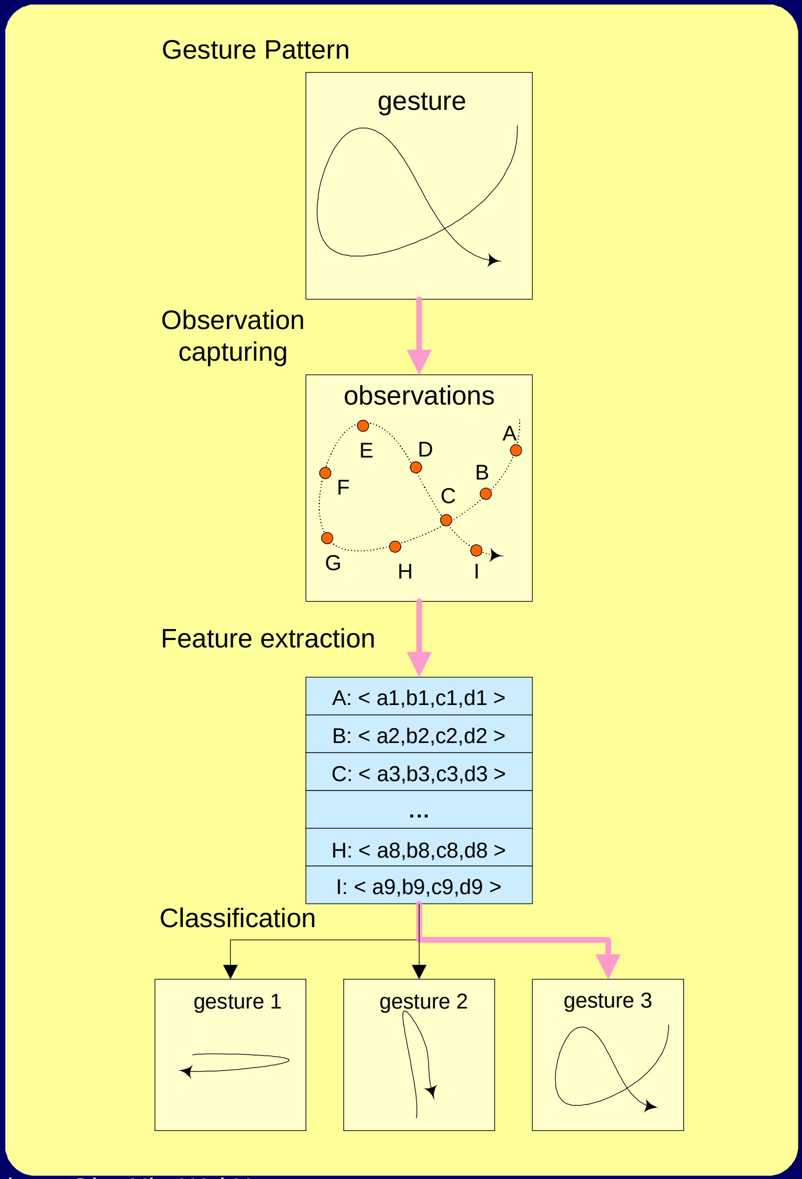
Prediction with alignment



Gesture Recognition

Gesture Recognition Procedure

- Observations capturing
- Feature extraction
- Classification



Gesture Recognition Technologies

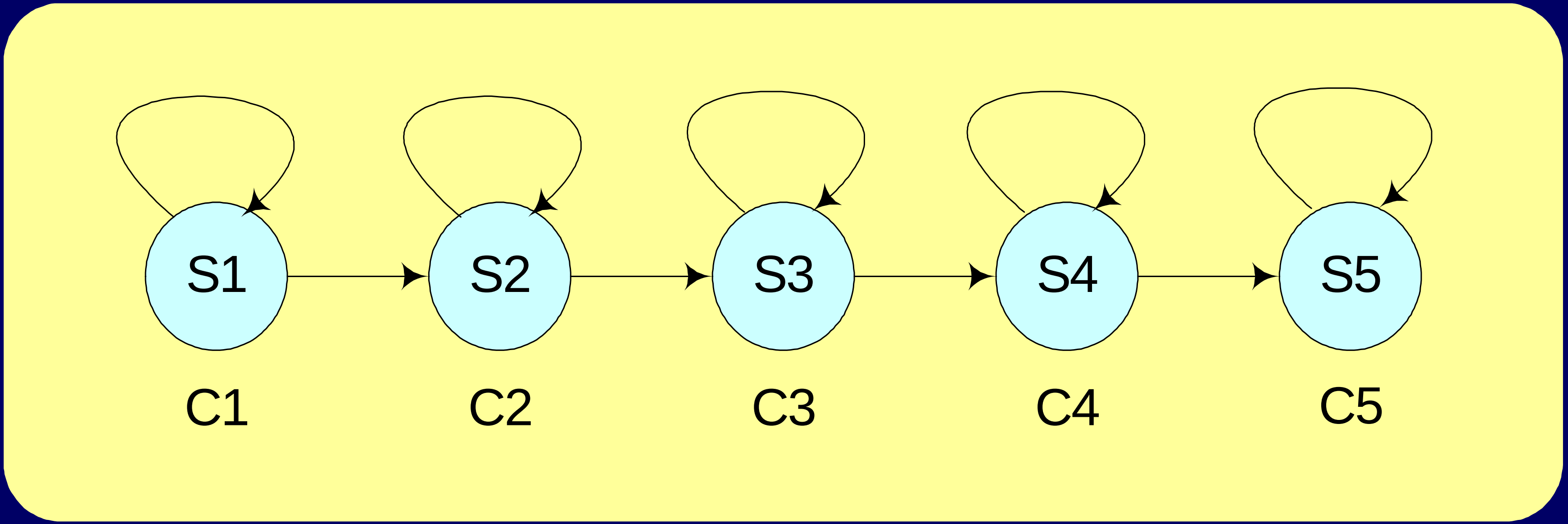
- HMM model for classification

  --- States transformation, probability

- RVM model for classification

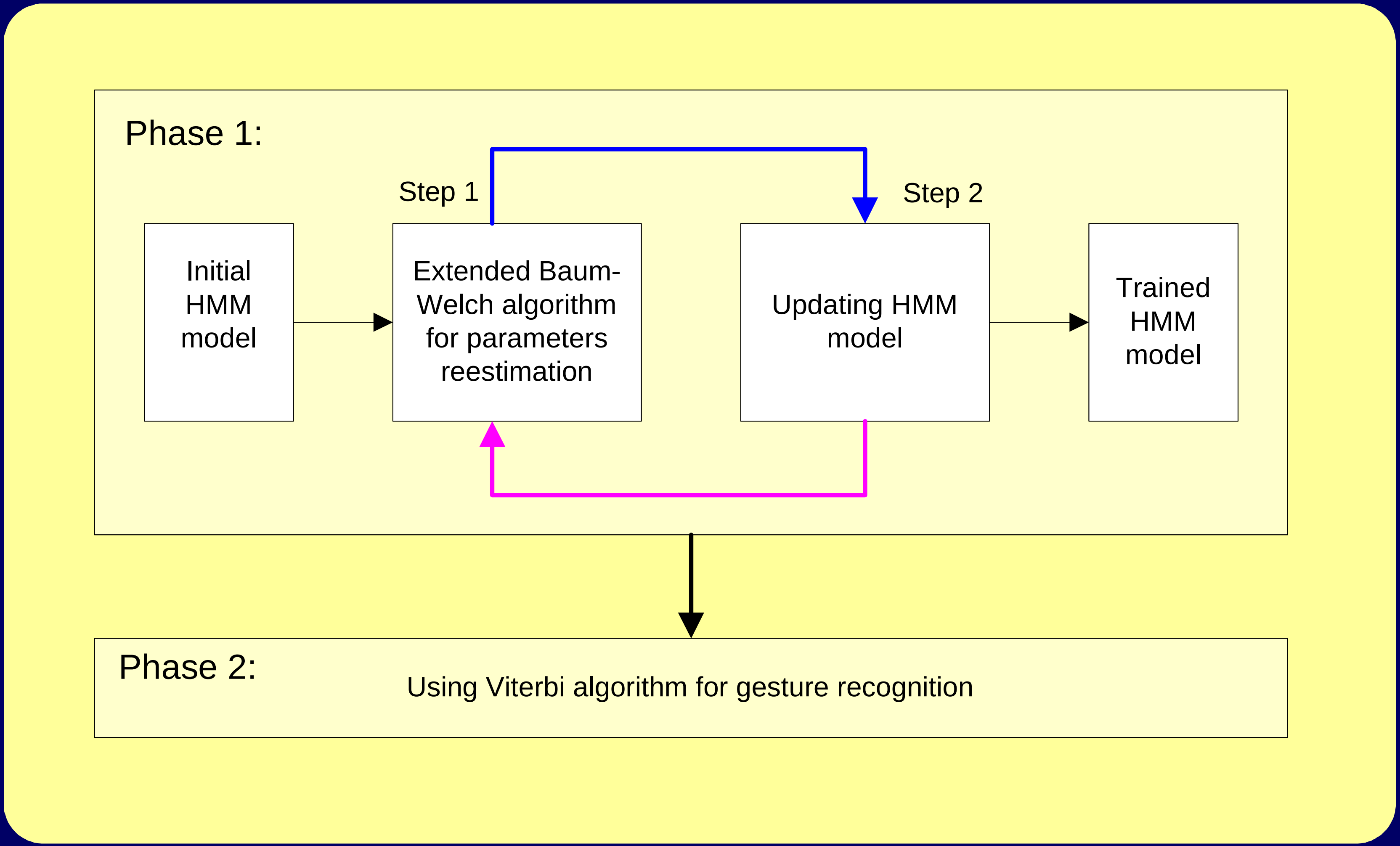
  --- Disjoint regions divided ,feature vectors

HMM for Gesture Recognition I



- Training phase
- Classification phase

HMM for Gesture Recognition II



HMM for Gesture Recognition III

- One HMM for each gesture (initial work)
- Need well-trained HMM (cost time)
- Speed up technology
- More fitful for continuous gesture recognition

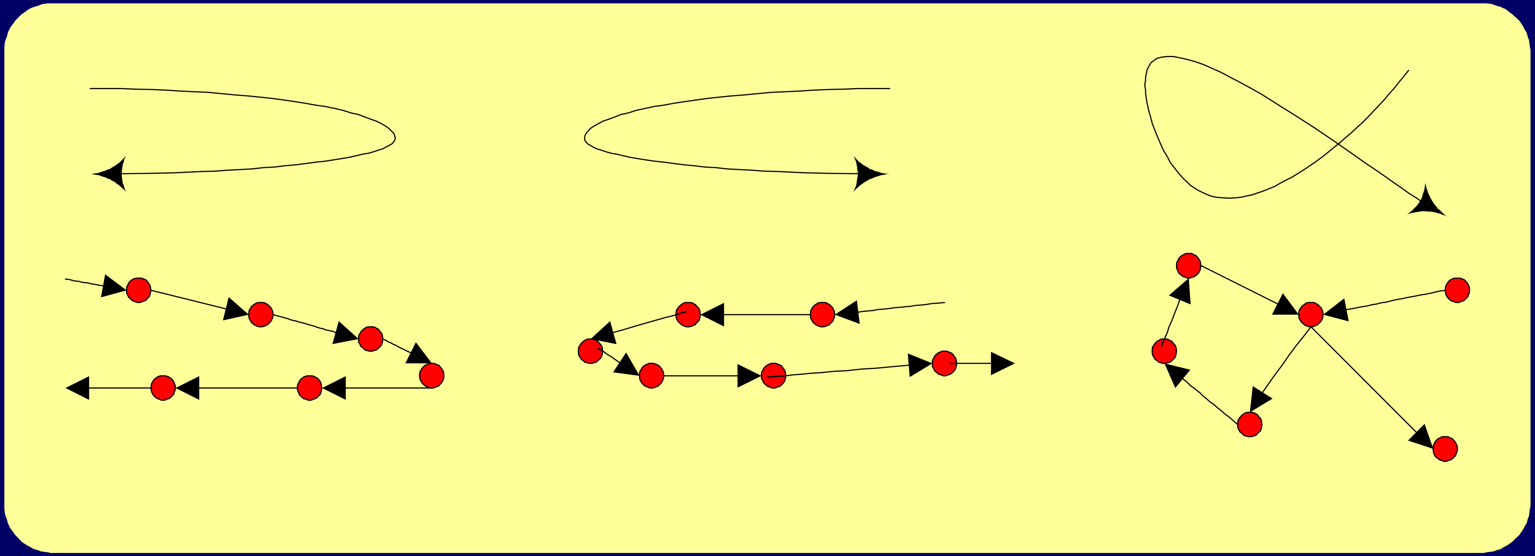
New Idea for Gesture Recognition

- We just apply simple gestures
- Gesture is a pattern
- We already have an existing classifier

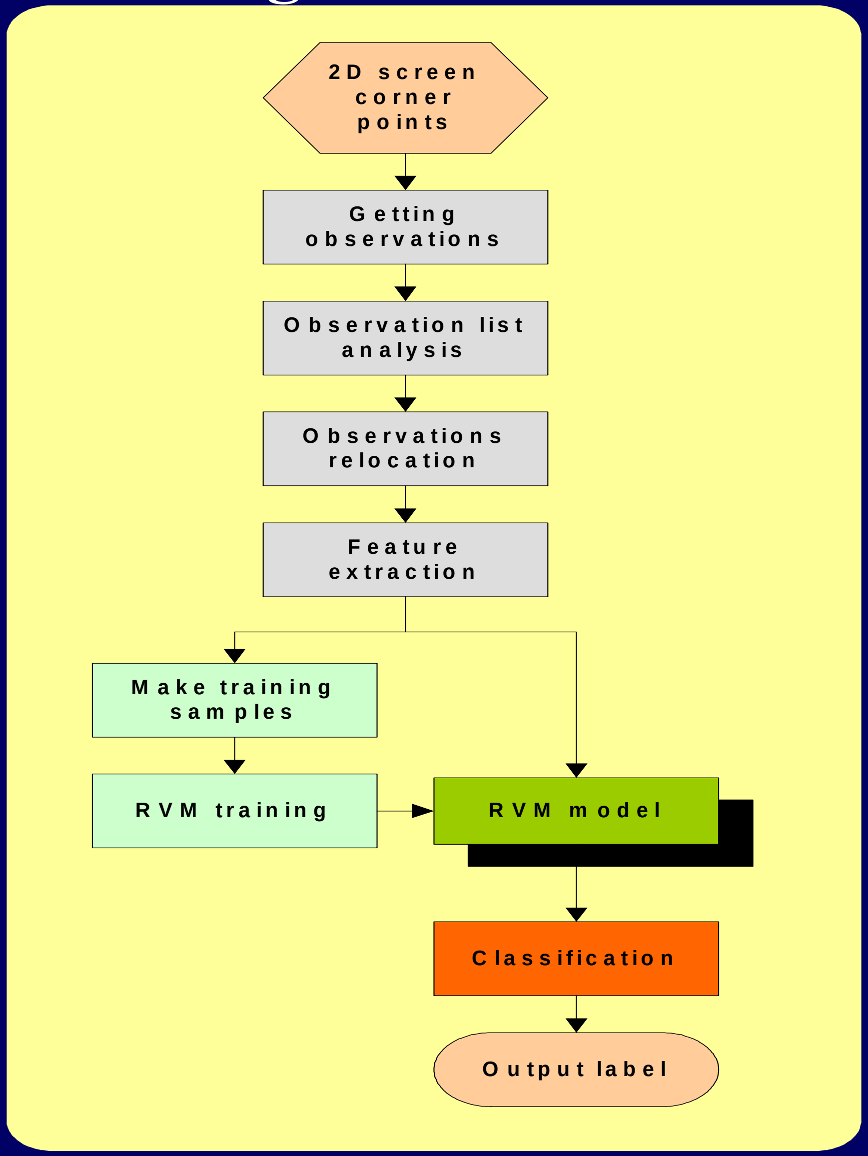
----RVM

RVM for Gesture Recognition

- Training phase
- Classification phase



RVM for Gesture Recognition

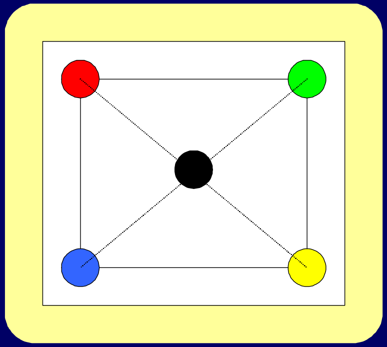


RVM Training Phase

- Getting observations
- Selecting candidate gesture traces
- Interpolating candidates traces (relocation observations)
- Extracting feature vectors
- Training RVM (training samples generation)

RVM Training Phase I

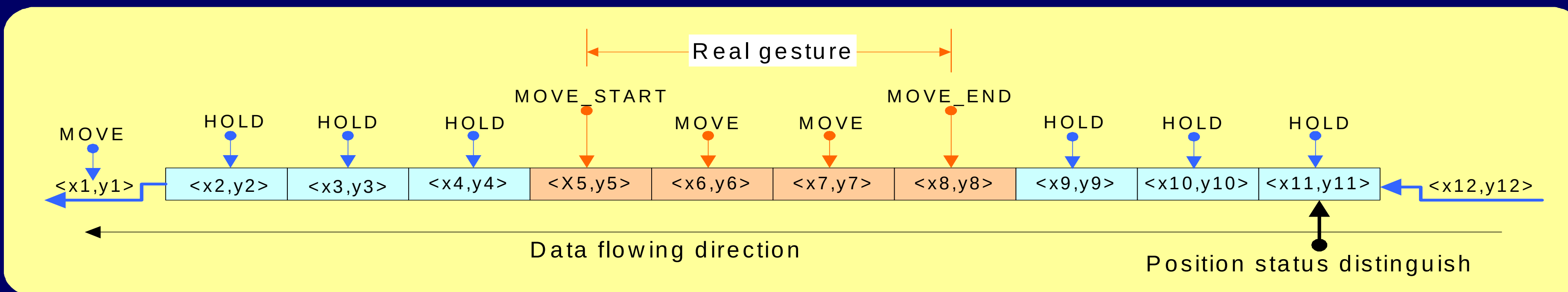
--- Getting observations



- ROI for tracking screen corners
- Four screen corners positioning
- Geometric center of the screen corners
- A sequence of positions as observations

RVM Training Phase II

---Selecting candidate gesture traces



- Observation list analysis

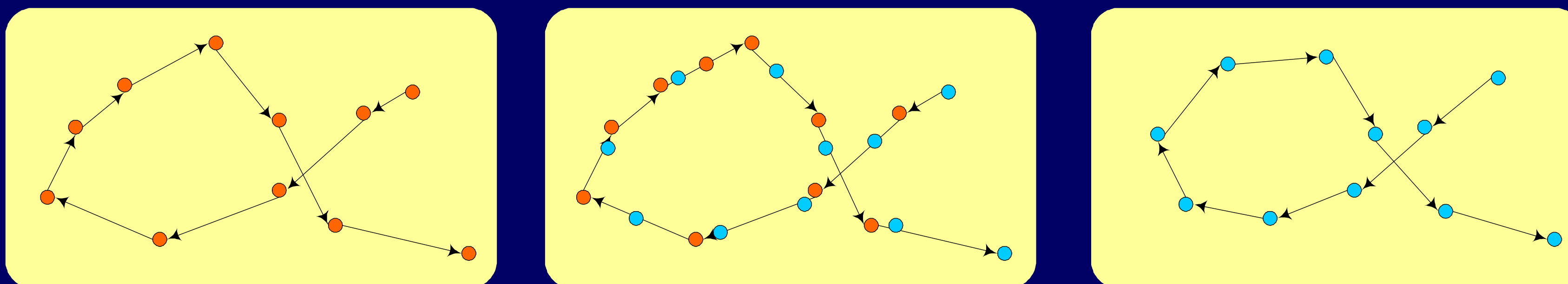
--- List length **(>=50 elements)**

--- Element states **(MOVE, HOLD, MOVE_START, MOVE_END, INITIAL)**

-- Thresholds **(MOVEMENT_SENSITIVITY, HOLDING_NUM_UNITTIME, MISSING_TOLERANCE)**

RVM Training Phase III

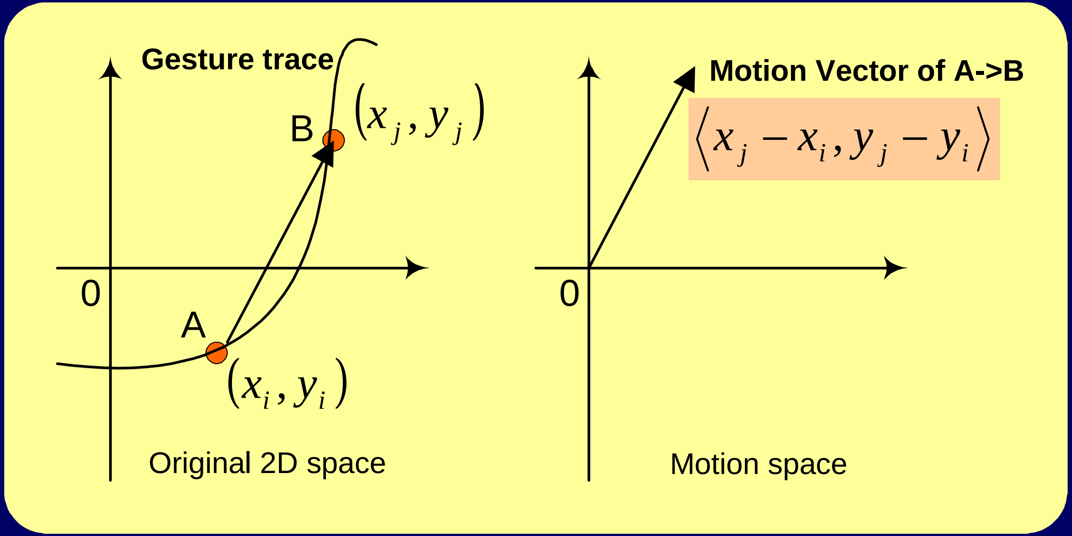
--- Interpolating candidate traces



- Balanced observations distribution
- Well-shaped gesture pattern
- Keep the original trace shapes

RVM Training Phase IV

--- Extracting feature vectors



- Motion vectors (direction and distance)
- Without normalizing (keep velocity information)
- 2*N-dimensional vectors (each for one gesture)

RVM Training Phase V

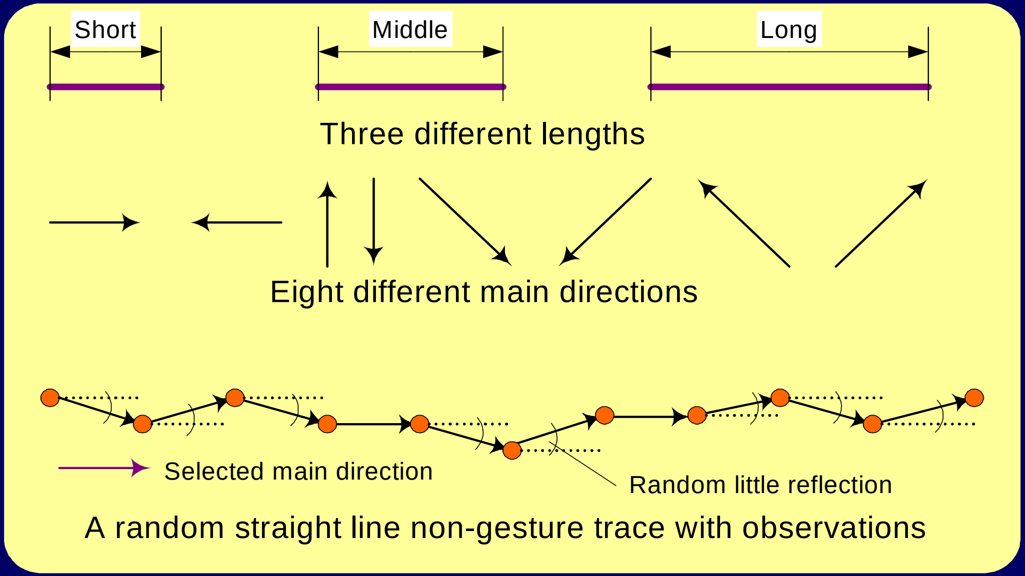
---Training RVM model

- Gesture traces generation (follows I,II,III,IV)
- Non-gesture traces generation (see examples)
- Training RVM (changes kernel function)

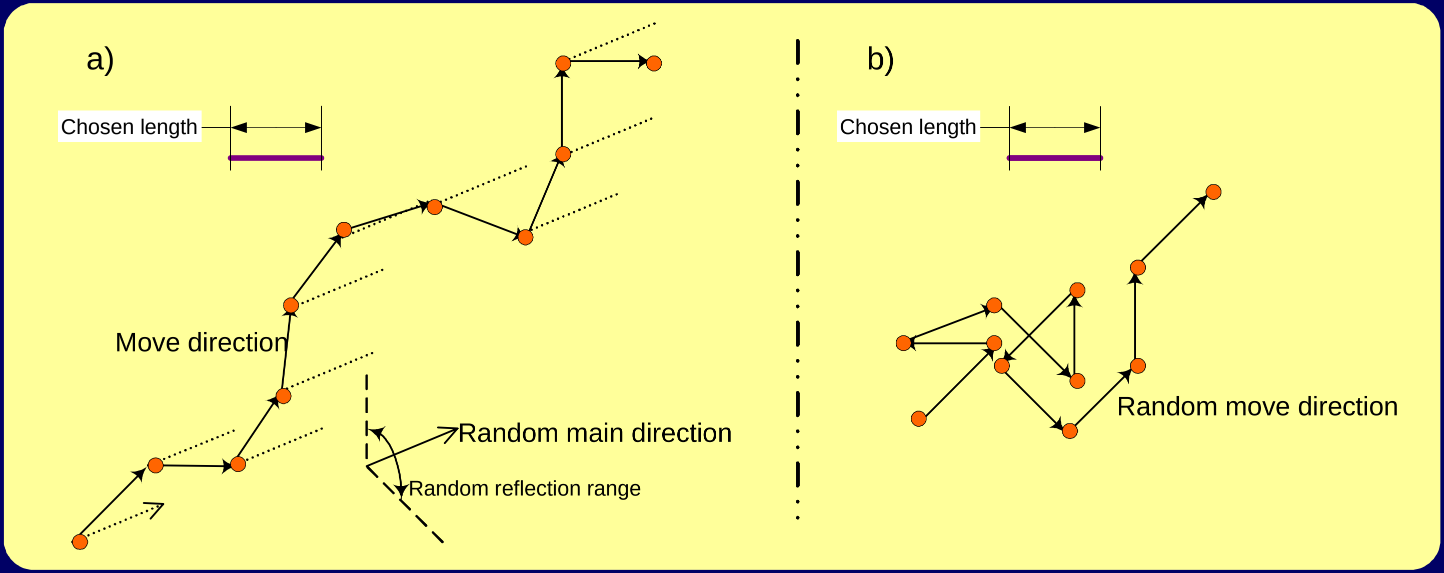
$$K(\mathbf{x}_{\mathbf{m}}, \mathbf{x}_{\mathbf{n}}) = (\mathbf{x}_{\mathbf{m}}\mathbf{x}_{\mathbf{n}} + 1)^{r}$$

Examples

Random straight lines



Random lines



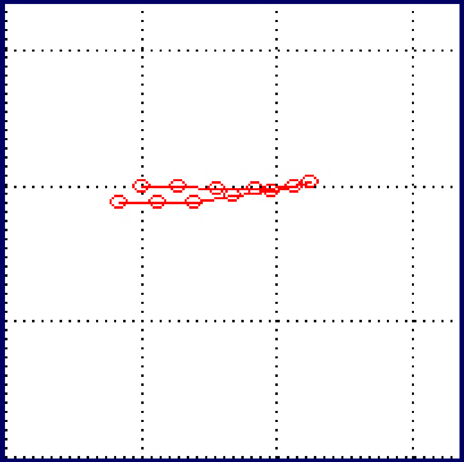
RVM Classification Phase

- Getting observations
- Selecting candidate gesture traces
- Interpolating candidate gesture traces
- Feature extraction
- Putting into RVM for classification

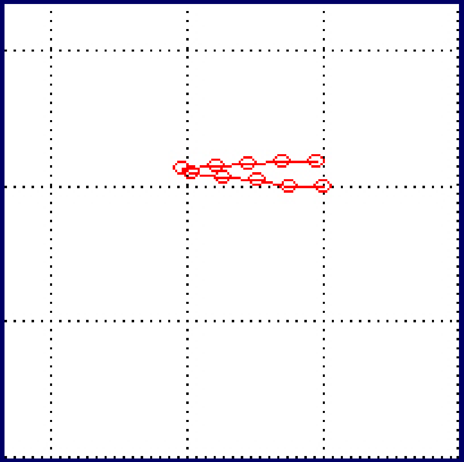
--- Gesture type can be specified.

RVM Gesture Recognition Tests I

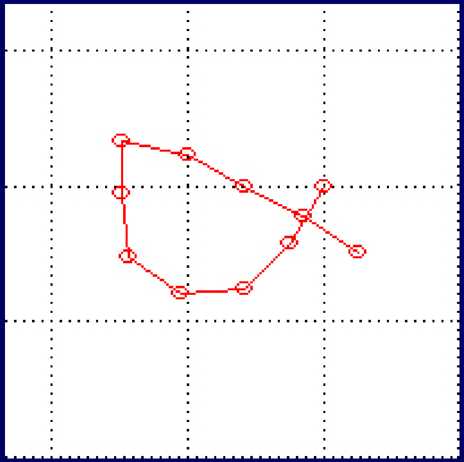
Left-Right



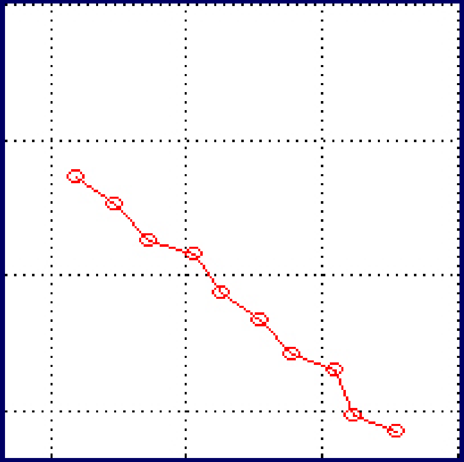
Right-Left



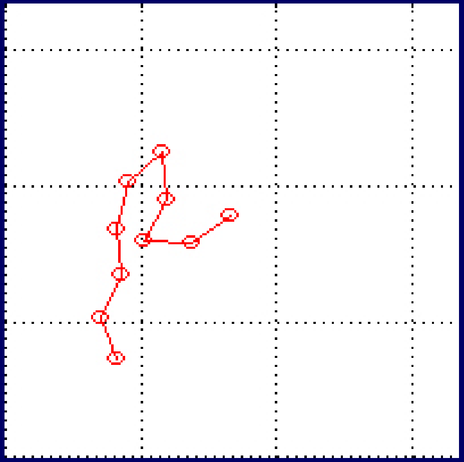
Cross



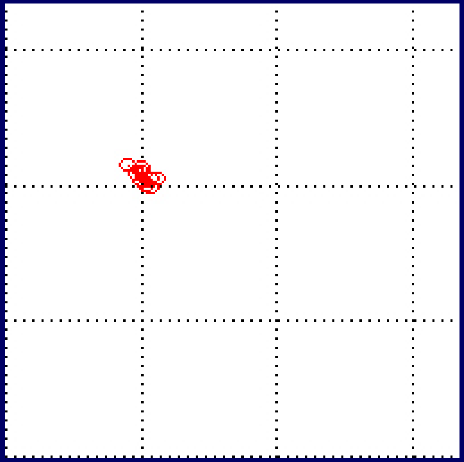
Non-gesture



Non-gesture

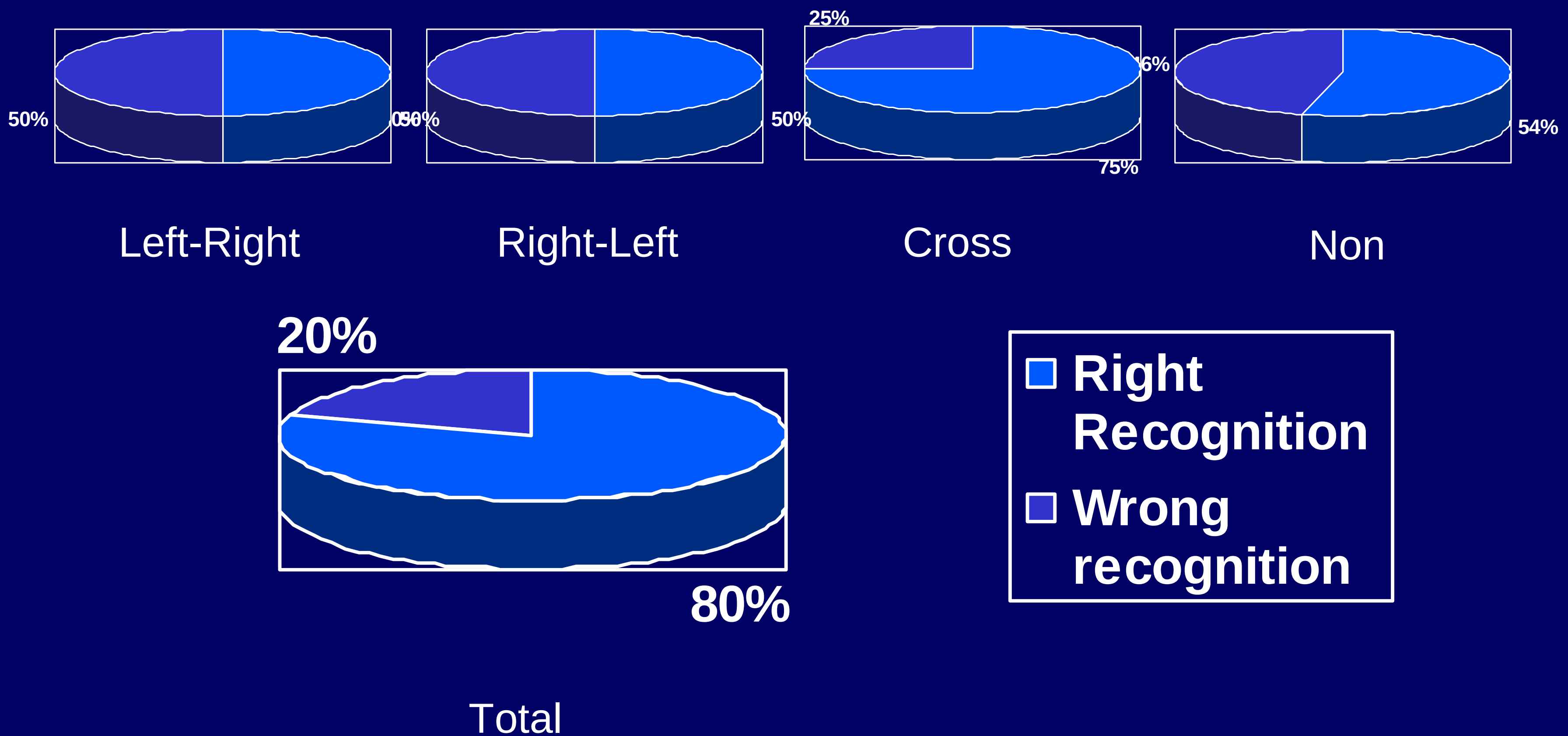


Non-gesture



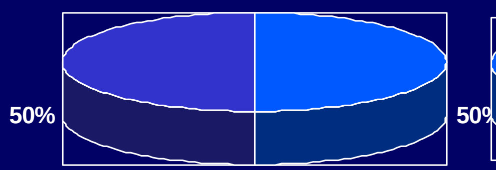
RVM Gesture Recognition Tests II

---Sequence 1

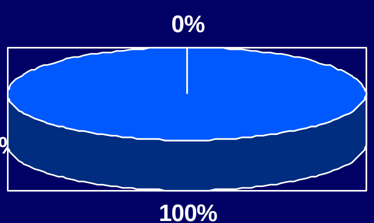


RVM Gesture Recognition Tests II

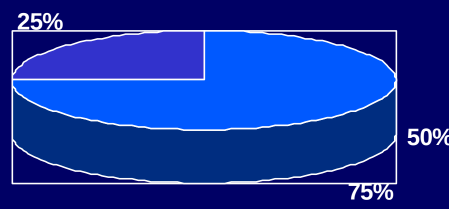
---Sequence 2



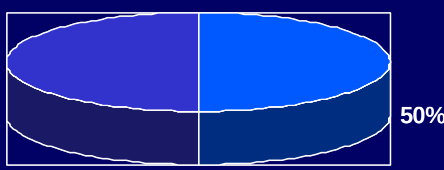
Left-Right



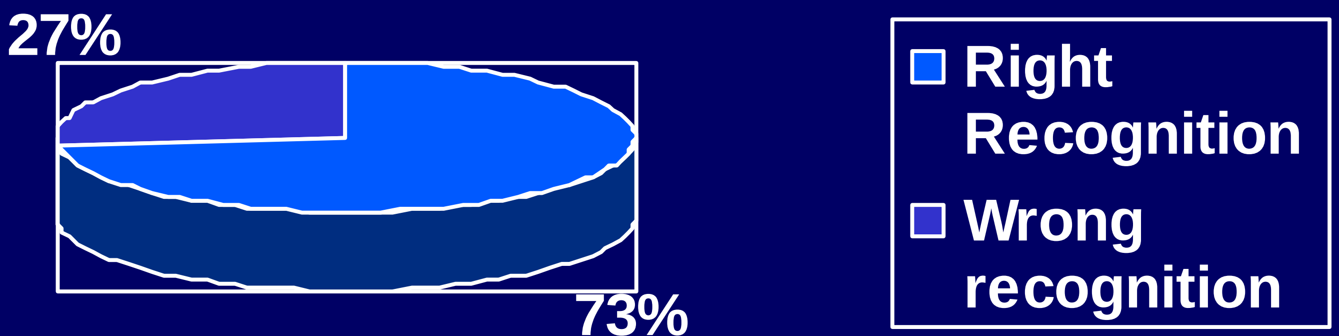
Right-Left



Cross



Non



Total

RVM Gesture Recognition Conclusions

Problems:

- Testing results are not so satisfied as our imagination

  --- Training sample qualities and quantities

Future works:

- Thresholding every gesture for recognition
- Continuous real-time processing

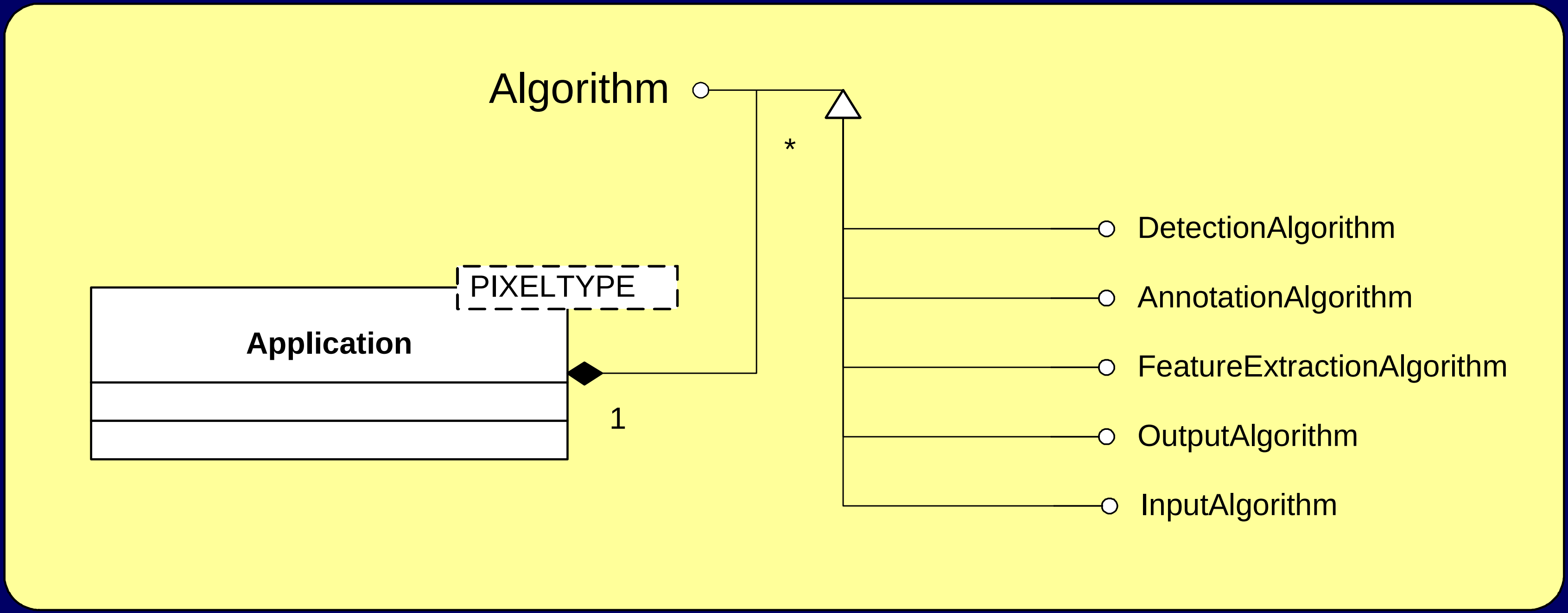
Part III: System Implementation Design

- UI-Wand System Design
- UI-Wand Utilities Design

UI-Wand System Design

Visipirin Framework

- Application class and Algorithm Interface



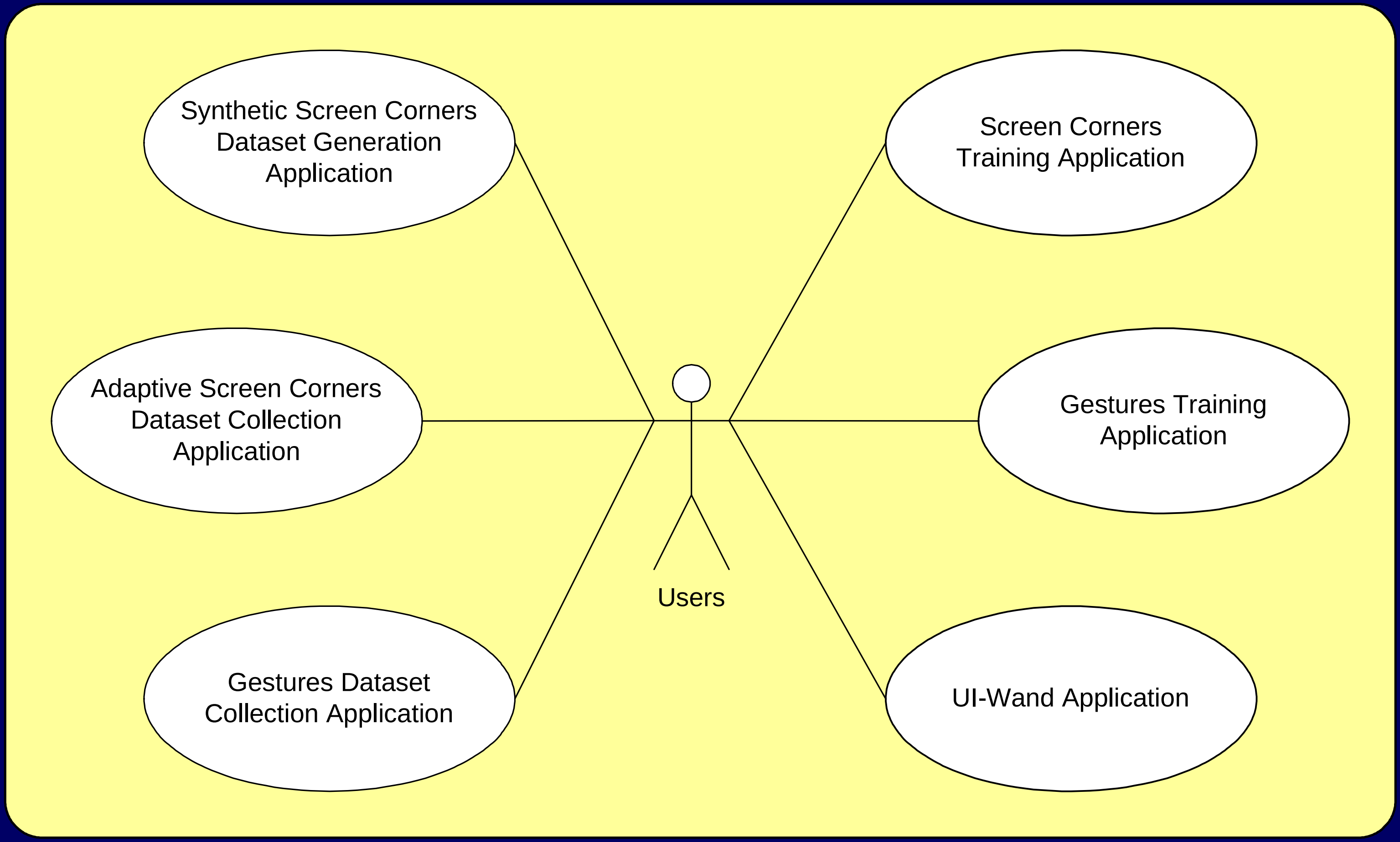
Visipirin Framework

- Application execution

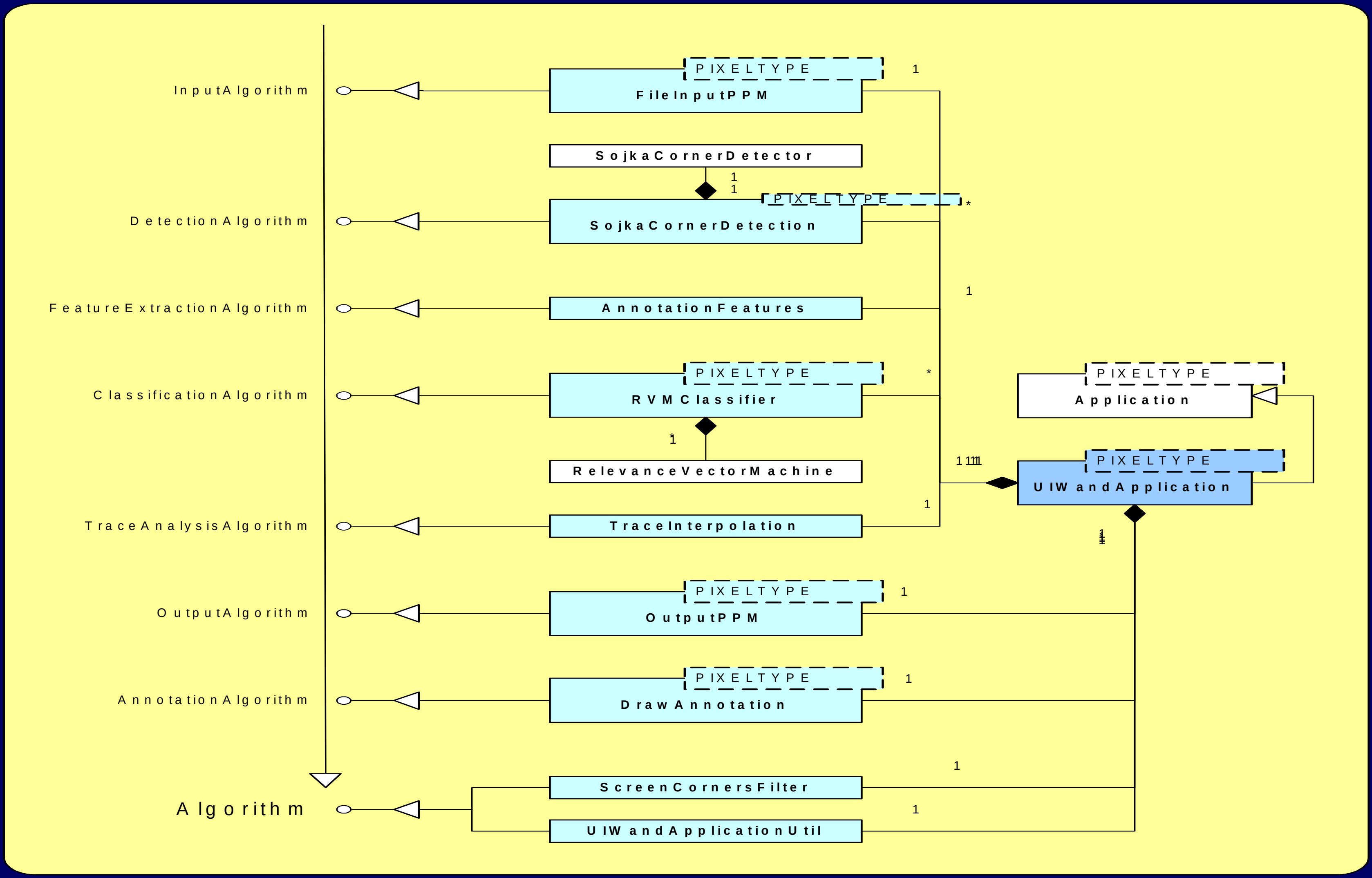
The command line format:
(a) *applicationname parameters ./theframesyouwanttorun/**
(b) *applicationname parameters.par ./theframesyouwanttorun/**

The parameters have the following format:
(a) *Scopename=AlgorithmName*
(b) *Scopename::ParameterName=ParameterValue*

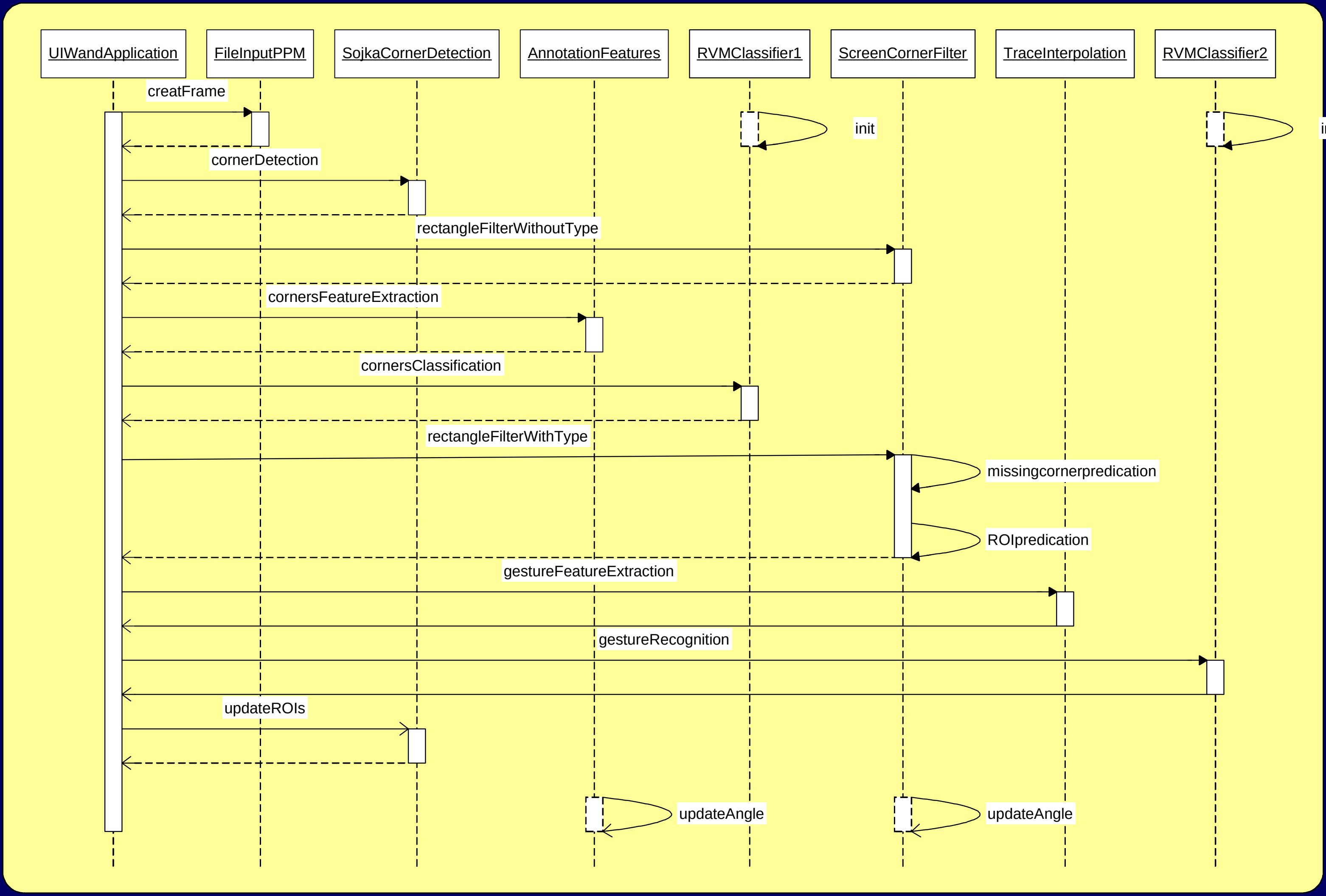
UI-Wand System Use Cases



UI-Wand Application Class Diagram

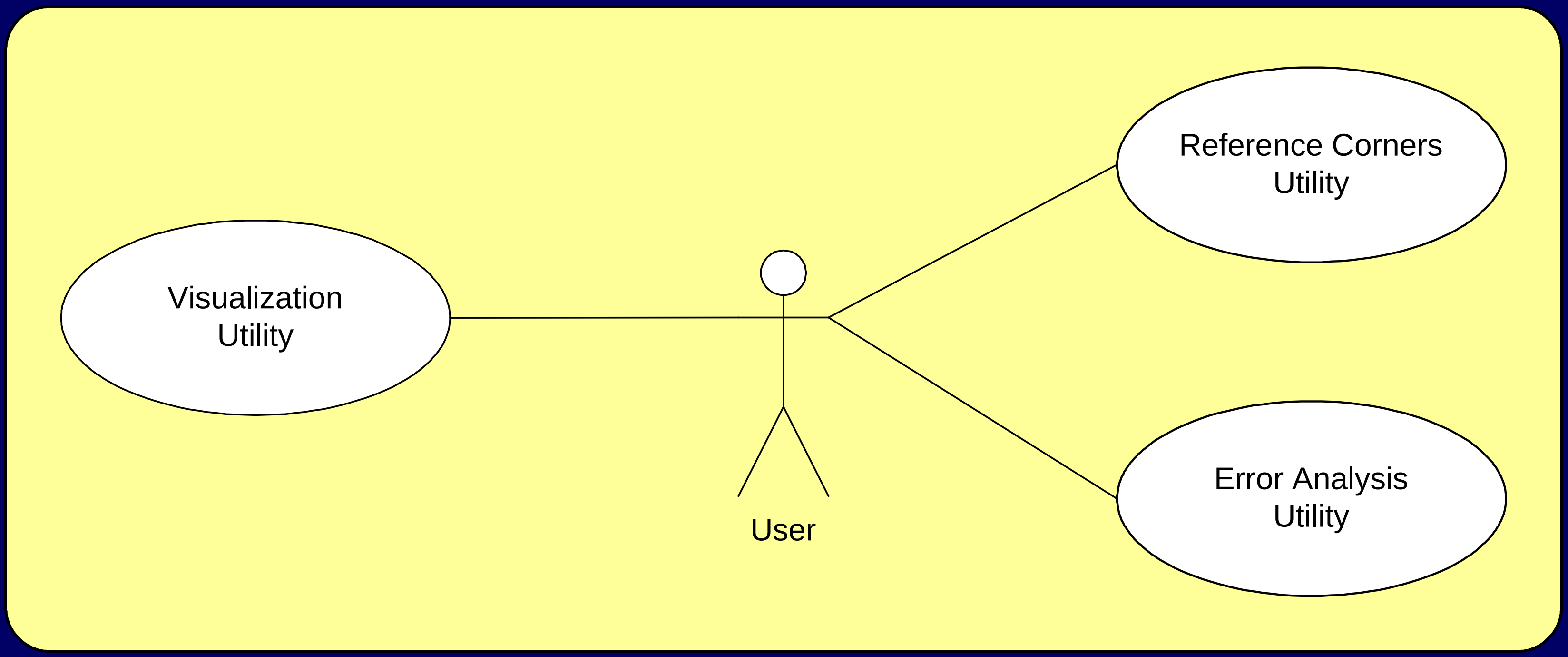


UI-Wand Application Sequence Diagram

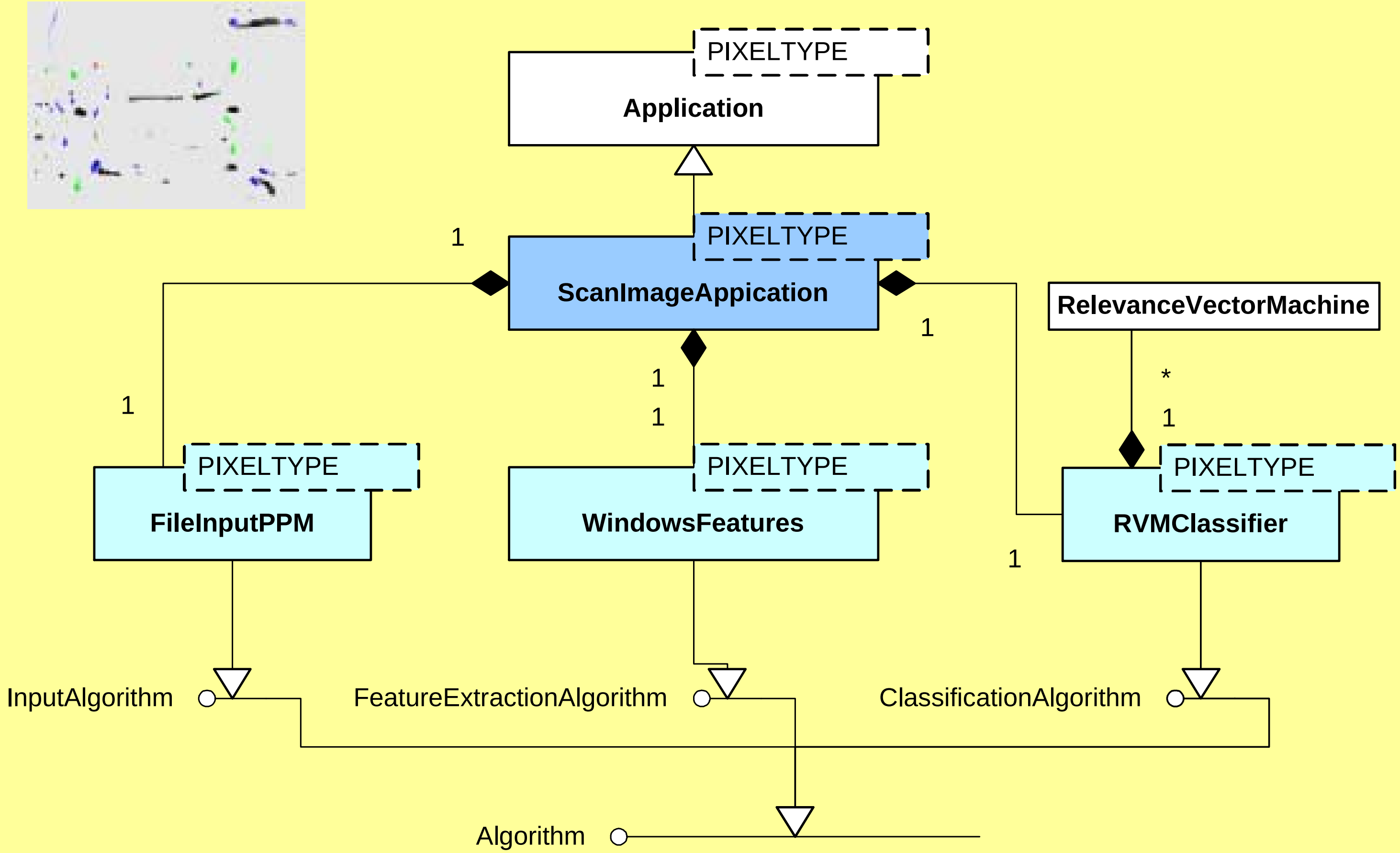


UI-Wand Utilities Design

UI-Wand Utilities Use Cases



Visualization Utility Class Diagram



Error Analysis GUI Utility

Error Analysis Results Table

- Corner detection analysis results

Corner Detection Analysis

		DCN	lt_D	lt_V	rt_D	rt_V	lb_D	lb_V	rb_D	rb_V	wh_RR
1	nages/testbed_27_05/frame0.ppm	30	y	1.41	n	----	y	2.24	y	3.16	3
2	nages/testbed_27_05/frame1.ppm	46	y	1.00	y	2.00	y	3.16	y	1.41	4
3	nages/testbed_27_05/frame2.ppm	32	y	2.00	y	2.24	y	2.24	y	1.41	4
4	nages/testbed_27_05/frame3.ppm	21	y	1.41	n	----	y	0.00	y	2.00	3
5	nages/testbed_27_05/frame4.ppm	28	y	1.00	y	2.24	y	3.61	y	3.00	4
6	nages/testbed_27_05/frame5.ppm	26	y	0.00	y	2.24	y	1.00	y	2.24	4
7	nages/testbed_27_05/frame6.ppm	25	y	1.00	y	2.24	y	1.00	y	2.24	4
8	nages/testbed_27_05/frame7.ppm	16	n	----	n	----	n	----	n	----	0
9	nages/testbed_27_05/frame8.ppm	30	y	2.00	y	1.00	y	0.00	y	1.00	4
10	nages/testbed_27_05/frame9.ppm	24	y	1.00	y	0.00	y	2.24	y	2.83	4
TOT		28	90%	1.20	70%	1.71	90%	1.72	90%	2.14	85.00%

Error Analysis Results Table

- RVM classification analysis results

RVM Classification Analysis

		lt_C	lt_HCV	lt_MV	lt_GV	lt_CGV	rt_C	rt_HCV	rt_MV	rt_GV	rt_CGV
1	s/frame0.ppm	y	101.04	3.61	2.22	1.72	y	64.66	18.38	6.93	5.75
2	s/frame1.ppm	y	1.00	3.61	1.99	1.68	y	98.08	5.00	2.98	2.55
3	s/frame2.ppm	y	3.16	4.47	2.71	2.24	y	105.69	5.00	2.85	2.52
4	s/frame3.ppm	y	41.62	3.61	2.07	1.52	y	3.61	7.81	3.77	3.40
5	s/frame4.ppm	y	40.00	4.47	2.69	2.07	y	115.75	6.40	3.33	2.93
6	s/frame5.ppm	y	40.61	4.12	2.33	1.66	y	115.62	7.07	3.44	3.20
7	s/frame6.ppm	y	91.02	3.61	2.04	1.55	y	38.01	43.93	15.27	11.60
8	s/frame7.ppm	y	118.98	3.61	2.22	1.66	y	119.02	5.00	2.94	2.59
9	s/frame8.ppm	y	42.01	3.16	1.89	1.63	y	117.61	5.00	2.44	2.17
10	s/frame9.ppm	y	89.01	4.24	2.56	2.08	y	3.16	36.40	13.68	10.58
TOT		100%	56.85	3.85	2.27	1.78	100%	78.12	14.00	5.76	4.73

lb_C	lb_HCV	lb_MV	lb_GV	lb_CGV	rb_C	rb_HCV	rb_MV	rb_GV	rb_CGV	MUR	wh_RR	wh_V
y	49.04	5.39	2.84	2.57	y	2.24	5.00	3.03	2.45	90%	100%	8.98
y	41.88	5.39	3.01	2.64	y	1.41	4.47	2.64	2.44	85%	100%	6.01
y	41.18	5.39	3.03	2.70	y	2.24	4.47	2.80	2.62	87%	100%	6.44
y	47.63	4.47	2.43	2.15	y	2.00	5.10	2.91	2.63	87%	100%	4.98
y	42.80	6.40	3.46	3.22	y	3.00	4.12	2.61	2.30	86%	100%	7.84
y	46.23	3.61	2.26	2.01	y	2.24	4.47	2.74	2.47	86%	100%	7.61
y	46.10	4.47	2.45	2.19	y	2.24	4.47	2.74	1.97	87%	100%	9.83
y	46.40	4.47	2.26	2.04	y	48.02	5.00	3.08	2.88	84%	100%	10.73
y	47.20	3.61	2.28	1.97	y	1.00	4.47	2.64	2.39	86%	100%	7.34
y	45.34	5.00	2.79	2.57	y	2.83	5.39	3.30	2.83	86%	100%	8.82
100%	45.38	4.82	2.68	2.41	100%	6.72	4.70	2.85	2.50	87%	100%	7.86

Error Analysis Results Table

- Complete application detection analysis results

Whole Test Analysis

		lt_E	lt_V	rt_E	rt_V	lb_E	lb_V	rb_E	rb_V	whole_V	HCF	LCF	WCF	WF
1	7_05/frame0.ppm	y	2.83	y	6.32	y	5.83	y	5.83	5.20	y	n	n	n
2	7_05/frame1.ppm	y	4.24	y	4.47	y	4.12	y	2.00	3.71	y	n	n	n
3	7_05/frame2.ppm	y	5.83	y	4.47	y	3.16	y	5.00	4.62	y	n	n	n
4	7_05/frame3.ppm	y	3.16	y	5.00	y	4.12	y	4.47	4.19	y	n	n	n
5	7_05/frame4.ppm	y	4.24	y	5.66	y	9.43	y	6.08	6.35	y	n	n	n
6	7_05/frame5.ppm	y	4.24	y	5.00	y	4.24	y	5.10	4.65	y	n	n	n
7	7_05/frame6.ppm	y	3.61	y	3.16	y	5.00	y	5.39	4.29	y	n	n	n
8	7_05/frame7.ppm	y	3.61	y	4.47	y	5.00	y	5.83	4.73	y	n	n	n
9	7_05/frame8.ppm	y	3.16	y	2.83	y	5.00	y	3.61	3.65	y	n	n	n
10	7_05/frame9.ppm	y	5.83	y	4.24	y	5.39	y	5.00	5.11	y	n	n	n
TOT		100%	4.08	100%	4.56	100%	5.13	100%	4.83	4.65	100%	0%	0%	0%

Part IV: System Tests and Future Works

- System Tests
- Conclusions and Future works

System Tests

System Test Conditions

- Hardware conditions:

  --- Test machine (PIV-2.4G, 256M)

  --- Test monitor (PHILIPS brilliance 180P2 black LCD computer screen)

  --- Devices (UI-Wand components)

- Software conditions:

  --- Operation systems (Red Hat Linux, Windows XP)

  --- Error analysis tool (UI-Wand Error Analysis Utility)

System Tests I

- Off-line tests I

Seq.	Corner detection	Corner detection +RVM classification	Corner detection + RVM classification + Filter	
	Corner number	Corner number	Variance	Rates
SS1	82	16	2.09	87%
SS2	78	22	1.60	100%
SS3	78	23	1.52	100%
SS4	79	17	1.81	100%
SS5	50	10	1.83	75%
SS6	69	13	2.12	93%
SS7	61	11	4.90	70%
SS8	60	10	9.57	90%
SS9	78	20	1.72	100%
SS10	66	18	1.46	100%

System Tests II

- Off-line tests II

Seq.	Corner detection	Corner detection + RVM classification		Corner detection + RVM classification + Filters			
	DT	FE	RVM	DT	FE	RVM	WHOLE
LS1	138.80	127.27	60.93	27.93	51.77	9.07	88.77
LS2	127.00	132.82	61.65	29.60	54.98	7.92	92.5
LS3	113.05	123.70	63.03	42.87	52.95	9.83	105.65
AVR	126.28	127.93	61.87	33.47	53.23	8.94	95.64

System Tests III

- On-line tests

No.	Number of frames	Processing speed (fps)
1	813	10.11
2	875	10.92
3	1381	9.3
4	1490	11.35
5	1524	10.98
6	2186	10.93
7	2328	11.27
8	2420	11.79
9	2758	12.63
10	3309	10.32
AVR	1908	10.96

System Tests Conclusions

- High accuracy
- Fast speed

Conclusions and Future Works

Conclusions

- Detect screen corners by Candidates-Winners approach
- Invent a model for gestures recognition.
- The System is robust with some lighting changes.
- It can run different applications.
- It can work in a range of working space.
- Its speed is faster than 10 frames/sec.
- Developed evaluation utilities
- The development is based on Vispirin, the existing framework

Future Works

- Current system improvement

  1. Parameters optimization
  2. Try Kalman filter to do the motion predication
  3. Do some research on color images
  4. Improve gesture recognition model

- Invent a new structure

  Accelerate RVM Classification speed by using adaptive dataset, which should be a very promising way to go

Questions?

Thank you!

Soon….

PHILIPS

TUDelft

Delft University of Technology