Robust Real Time Pattern Matching using Bayesian Sequential Hypothesis Testing

Ofir Pele Michael Werman

The Hebrew University
of Jerusalem

Goal of Work

- Find patterns in images
- In real time
- Under different transformations:
  - Light changes
  - Occlusion
  - Non rigid deformations
  - (small) geometrical transforms

Pattern

Occurrence of pattern in image

Pattern

Occurrence of pattern in image

Performance

- Image Size: 640x480 pixels
- Online Time: **0.037 sec**

Related Work – Fast Pattern Matching

- Scale Invariant Features
  - Lowe 04
- Normalized Cross Correlation
  - Lewis 95
- Fast filters for norm distances
  - Cha 00, Hel-Or and Hel-Or 05
- Sequential sampling
  - Barnea and Silverman 72, Pereira and Mascarenhas 84
  - **Our method**

Contributions

- A simple and robust family of distance measures, the "*Image Hamming Distance Family*"

+

- Bayesian framework for sequential hypothesis testing on finite populations.

=

- **Robust Real Time Pattern Matching**

Sliding Window Approach

Sliding Window Approach Questions

- Which similarity measure ?
- Can we apply it fast ?
- Can we use information from one window to the next ?

Sliding Window Approach Questions

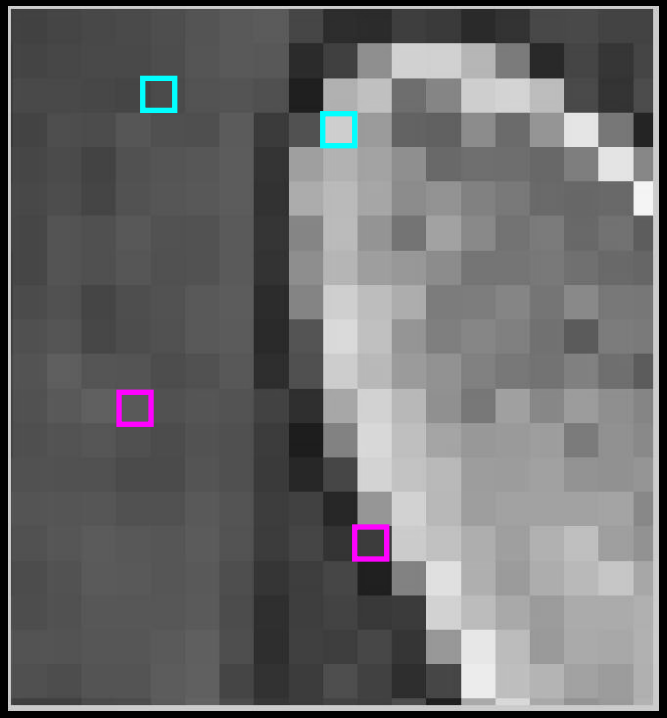
- **Which similarity measure ?**
- Can we apply it fast ?
- Can we use information from one window to the next ?

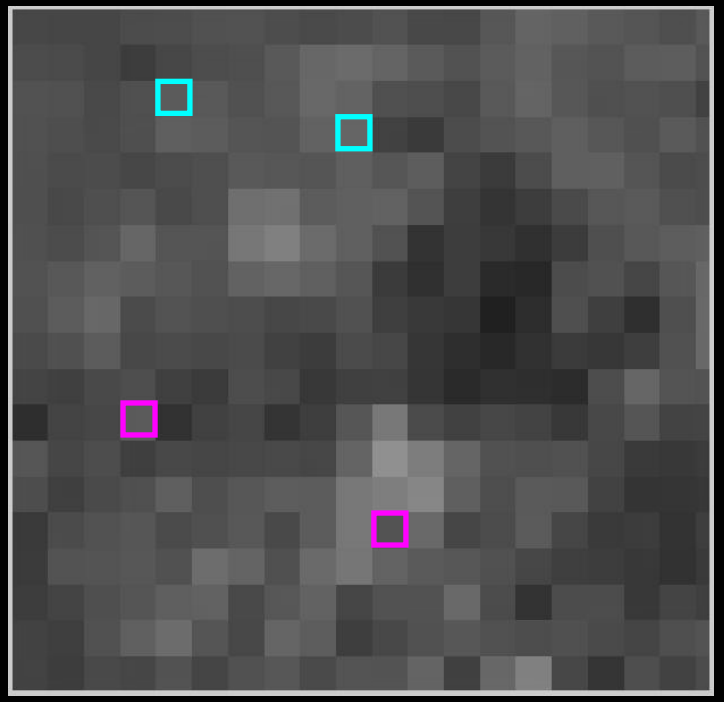
Sliding Window Approach Questions

- **Which similarity measure - Hamming**
- Can we apply it fast ?
- Can we use information from one window to the next ?

Hamming Distance on Pixels

Hamming Distance on Pairs of Pixels





Advantages of Hamming Distance

- Pluggable invariance:
  - Noise that preserve monotonic relations
  - Small non-rigid deformations
  - …
- Inherent robustness to noise that change the value of pixels completely:
  - Occlusions
  - Out of plane rotations
  - Shading
  - Spectral reflactance

Sliding Window Approach Questions

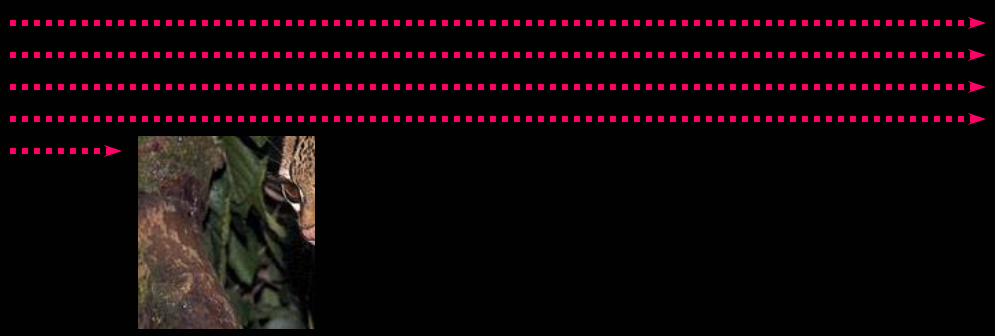
- Which similarity measure ?
- **Can we apply it fast ?**
- Can we use information from one window to the next ?

Sliding Window Approach Questions

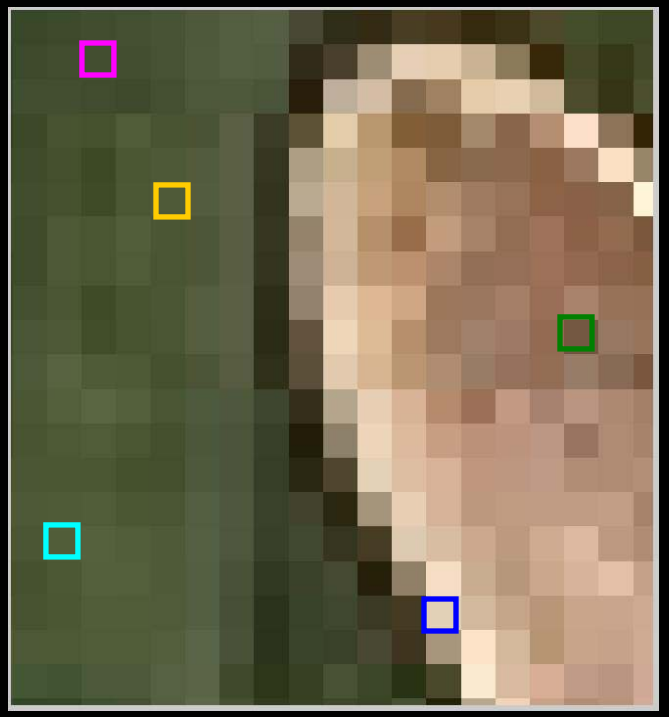
- Which similarity measure ?
- **Can we apply it fast - Sample**
- Can we use information from one window to the next ?

Related Work – SSDA (Barnea and Silverman 72)

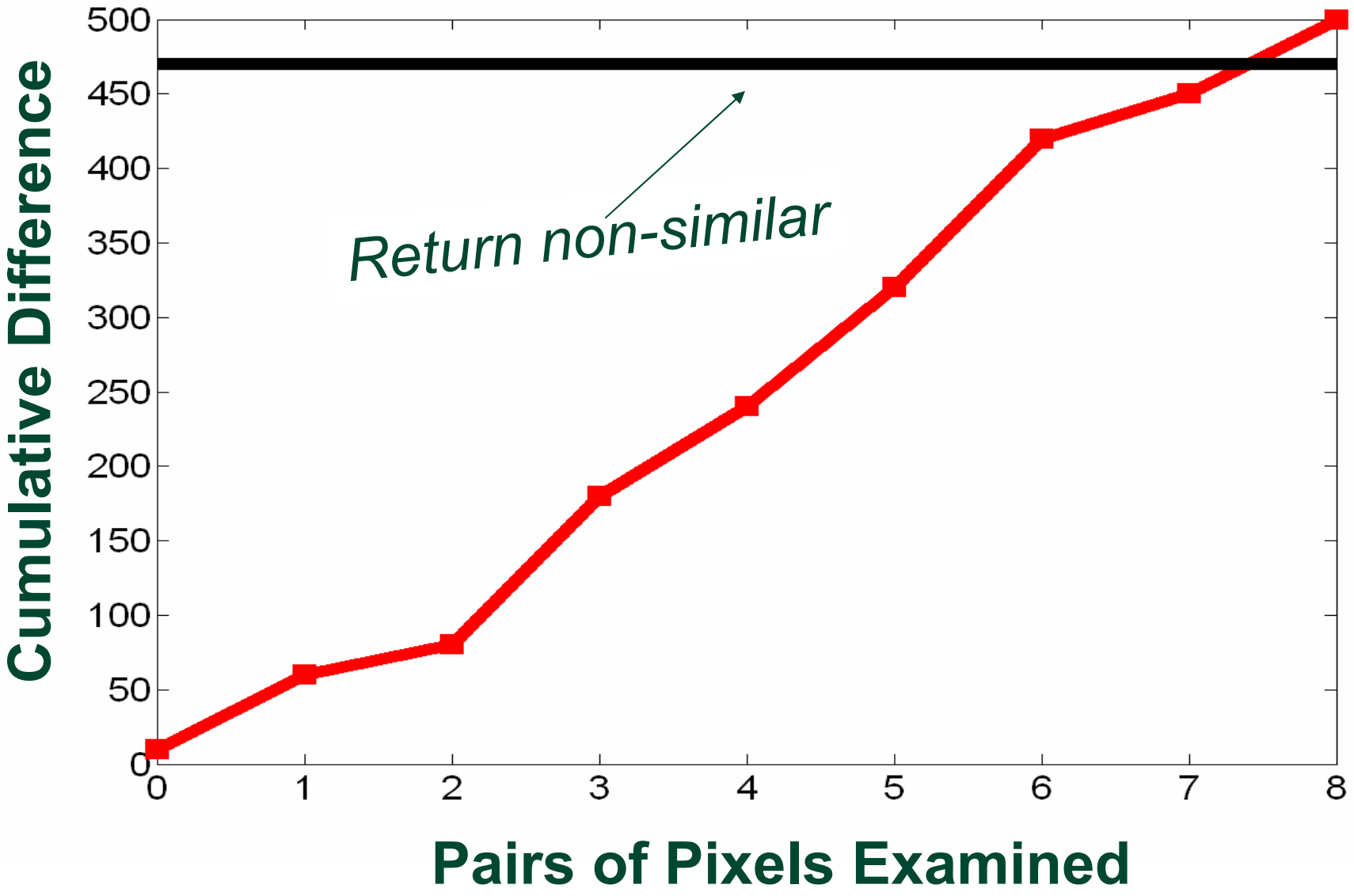
Related Work – SSDA (Barnea and Silverman 72)



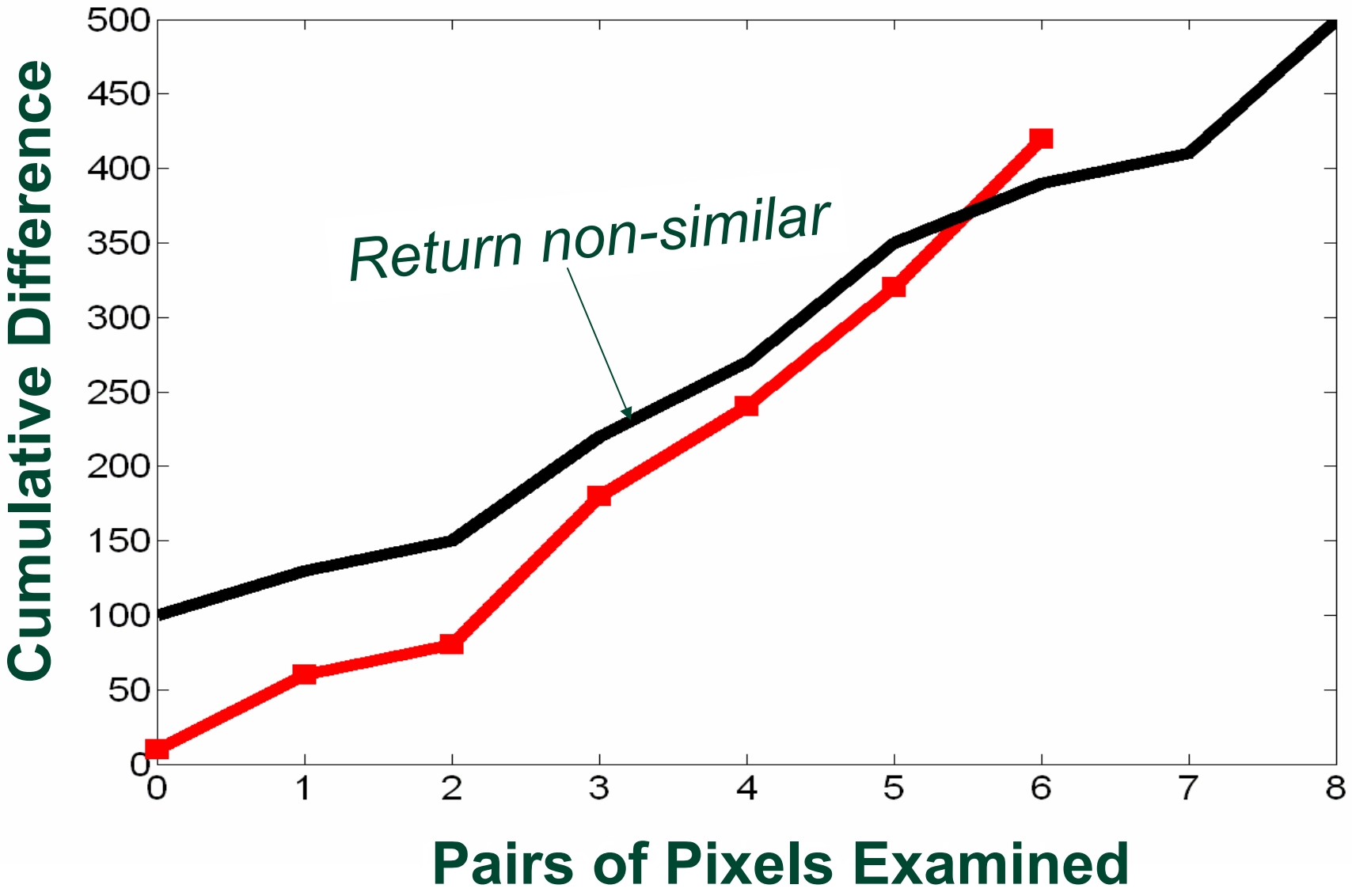
Related Work – SSDA (Barnea and Silverman 72)



Related Work – SSDA (Barnea and Silverman 72)



Related Work – SSDA (Barnea and Silverman 72)



Related Work – SSDA (Barnea and Silverman 72)

- Which distance measure to use ?
- How to design the sampling algorithm ?

Related Work – Wald's SPRT (Pereira and Mascarenhas 84)

- Sample while:

$$\mathcal{A} < \frac{P(\text{difference}|\text{images are non-similar})}{P(\text{difference}|\text{images are similar})} < \mathcal{B}$$

Related Work – Wald's SPRT (Pereira and Mascarenhas 84)

- Wald's approximation:

$$\mathcal{A} = \frac{\beta}{1-\alpha} \qquad \mathcal{B} = \frac{1-\beta}{\alpha}$$

$$\alpha = \text{Maximum false negative}$$

$$\beta = \text{Maximum false positive}$$

Related Work – Wald's SPRT (Pereira and Mascarenhas 84)

- How to model $P(\text{difference})$?

1. Transform to binary images ➔

$$P(\text{difference}) \sim \text{Binomial}$$

2. Assume everything Gaussian ➔

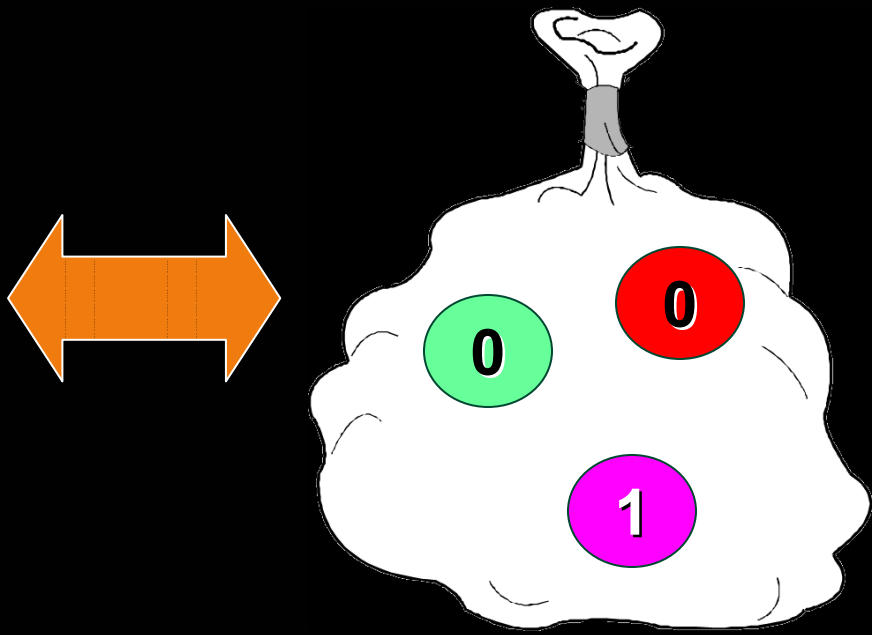
$$P(\text{difference}) \sim \text{Gaussian}$$

Designing an Optimal Sampling Algorithm

Reduction To Hypothesis Testing

Template:

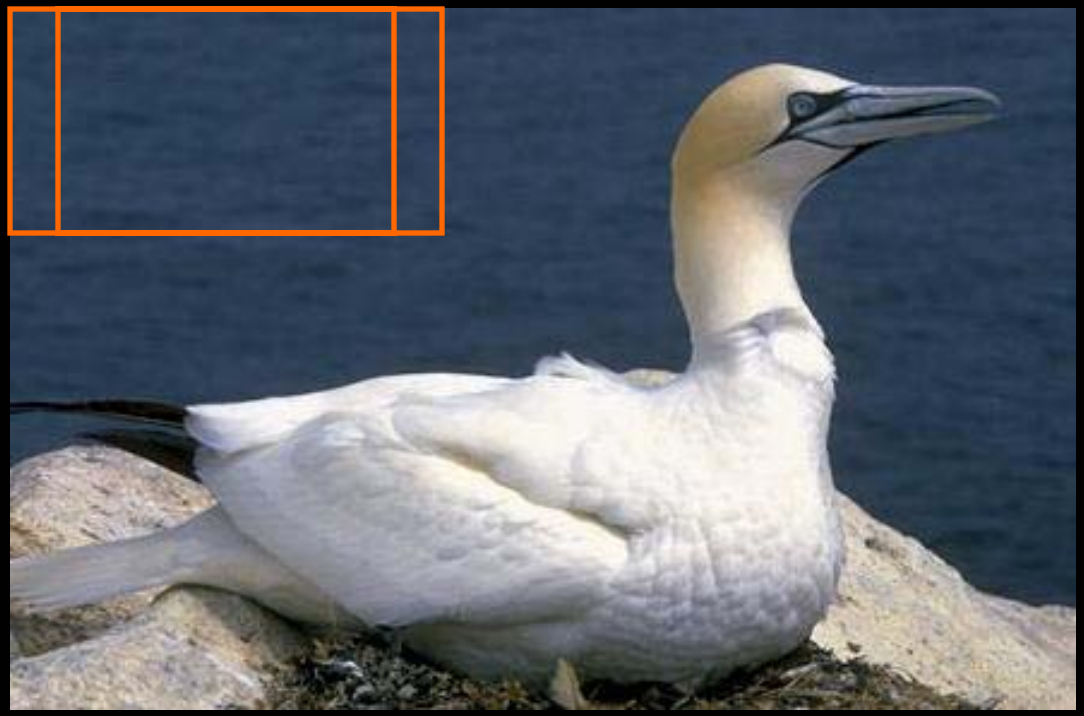
Sub Image:

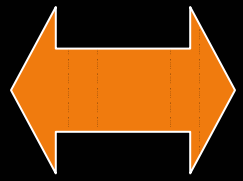


Reduction To Hypothesis Testing

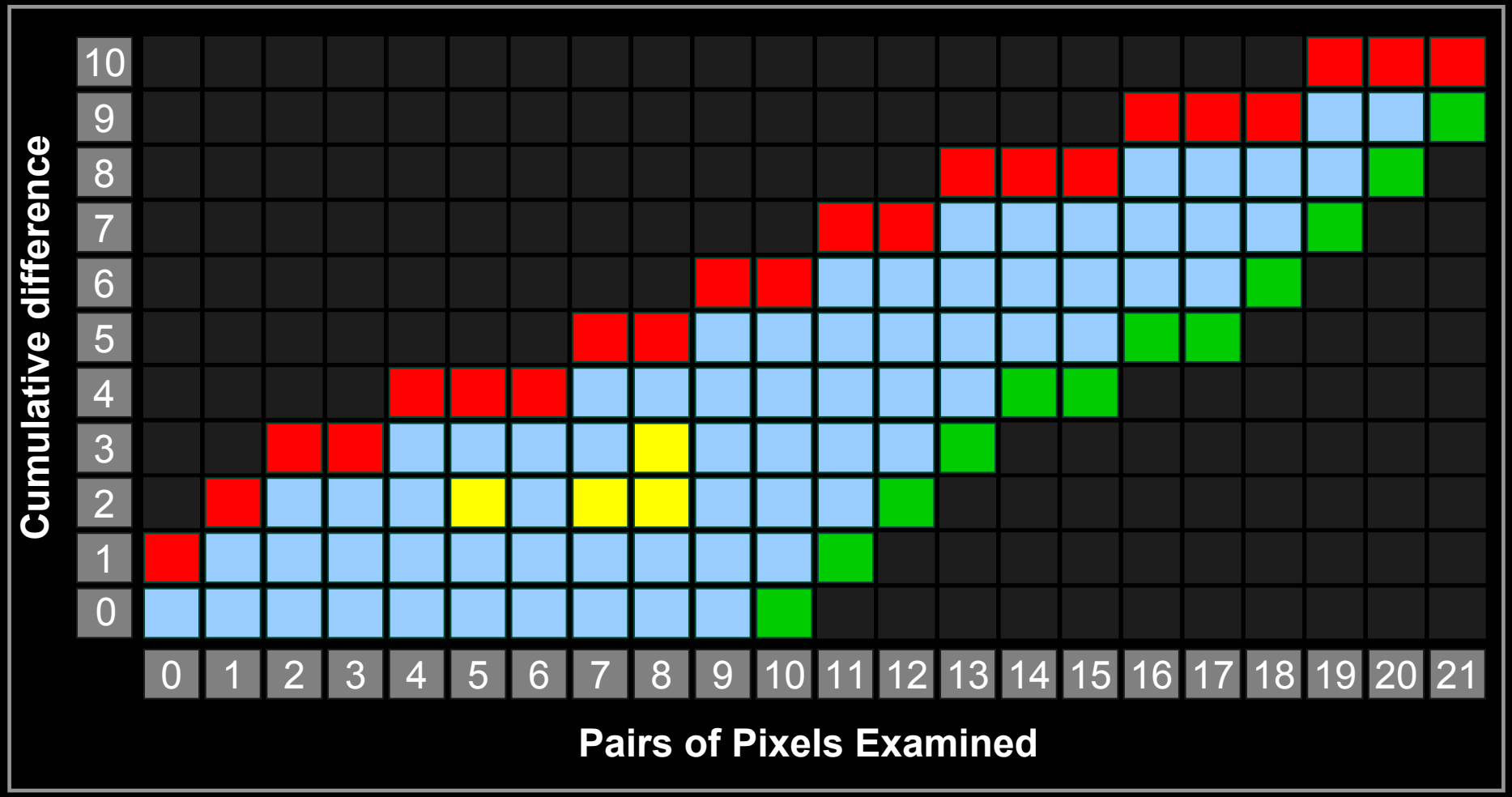
Template:

Image:





The Decision Matrix



Stop and return non-similar

Continue sampling

Stop and return similar

Do the exact computation

Rules of Game

- Each coin check cost R_I time units
- Each check of sack with n coins costs $R_{E(n)}$ time units

Optimization

$$\arg\min_{M} E_M(\text{running time})$$

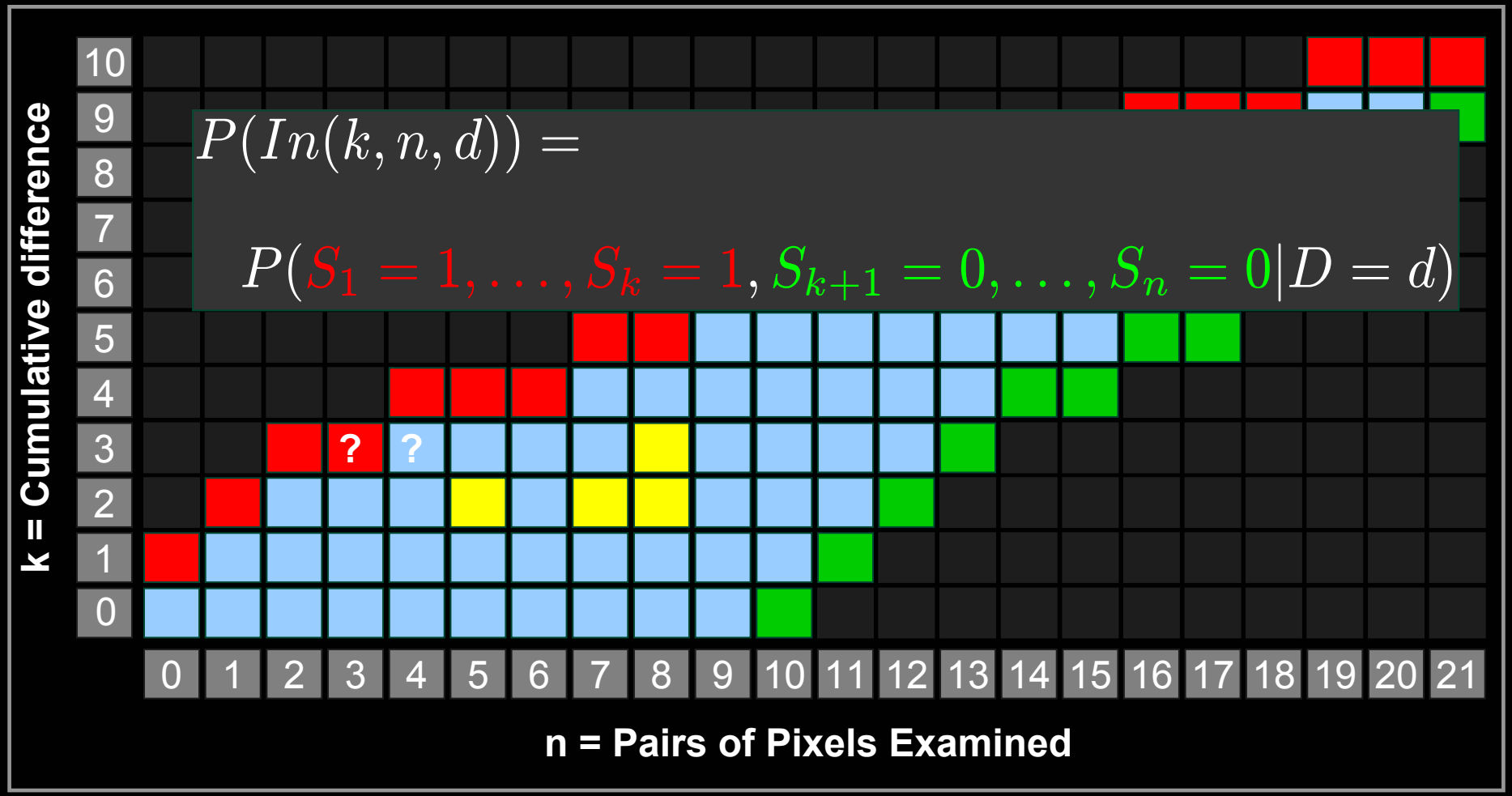
$$s.t:$$

$$P_M(\text{false negative}) \leq \alpha$$

$$P_M(\text{false positive}) \leq \beta$$

$$N \leq B$$

Computing Probabilities



Stop and return non-similar

Continue sampling

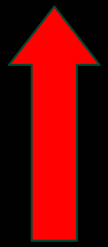
Stop and return similar

Do the exact computation

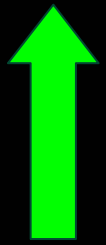
Computing Probabilities

$$P(In(k,n,d)) =$$

$$\begin{cases} \left(\prod_{i=0}^{k-1} \frac{d-i}{|A|-i}\right)\left(\prod_{i=0}^{n-k-1} \frac{|A|-d-i}{|A|-k-i}\right) & \text{if } (d \geq k)\&(|A|-d \geq n-k) \\ 0 & \text{otherwise} \end{cases}$$

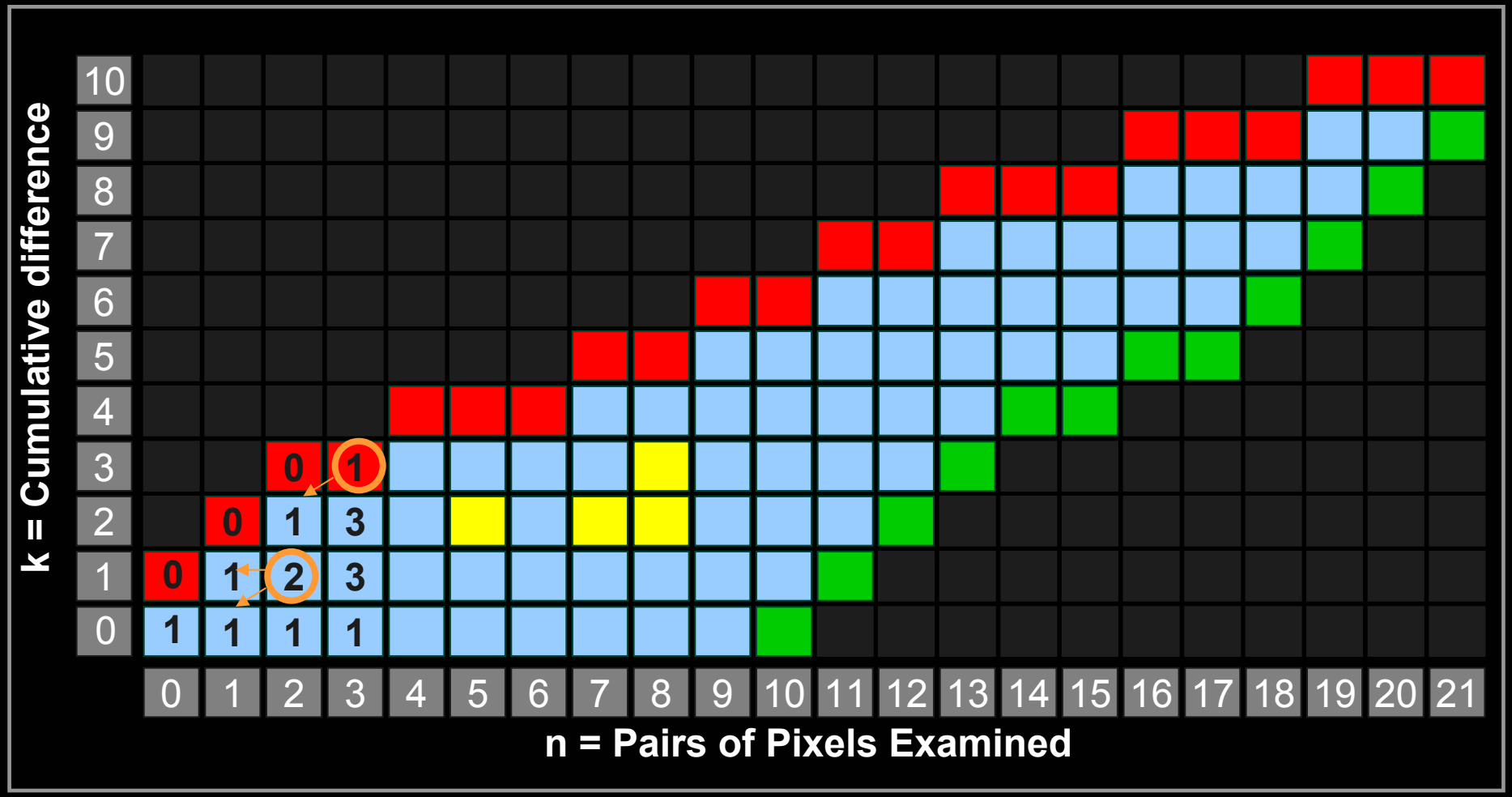


Non-similar pairs of pixels sampled



Similar pixels sampled

Computation of $\Psi_M[k, n]$



Computing Probabilities

- We define:

$$\Omega_S[k,n] = \sum_{d=0}^{t} P(In(k,n,d))P(D=d)$$

$$\Omega_{NS}[k,n] = \sum_{d=t+1}^{|A|} P(In(k,n,d))P(D=d)$$

Computing Probabilities – Closure

$$P_M(\text{false negative}) = \frac{\sum_{\substack{(k,n):\\ M(k,n)=NS}} \Psi[k,n]\Omega_S[k,n]}{P(D \leq t)}$$

$$P_M(\text{false positive}) = \frac{\sum_{\substack{(k,n):\\ M(k,n)=S}} \Psi[k,n]\Omega_{NS}[k,n]}{P(D > t)}$$

Computing Probabilities – Closure

$$
\begin{aligned}
E_M(\text{running time}) &= E_M(\#\text{ samples})R_I + P_M(\text{compute exact})R_{E(n)} \\
&= \sum_{\substack{(k,n):\\ M(k,n)\in\{S,NS,E\}}} \Psi[k,n](\Omega_S[k,n] + \Omega_{NS}[k,n])nR_I \\
&\quad + \sum_{\substack{(k,n):\\ M(k,n)=E}} \Psi[k,n](\Omega_S[k,n] + \Omega_{NS}[k,n])R_{E(n)}
\end{aligned}
$$

Optimization

$$\arg\min_{M} E_M(\text{running time})$$

$$s.t:$$

$$P_M(\text{false negative}) \leq \alpha$$

$$P_M(\text{false positive}) \leq \beta$$

$$N \leq B$$

Auxiliary problem

$$\arg\min_{M} loss(M, w_0, w_1)$$

$$s.t:$$

$$N \leq B$$

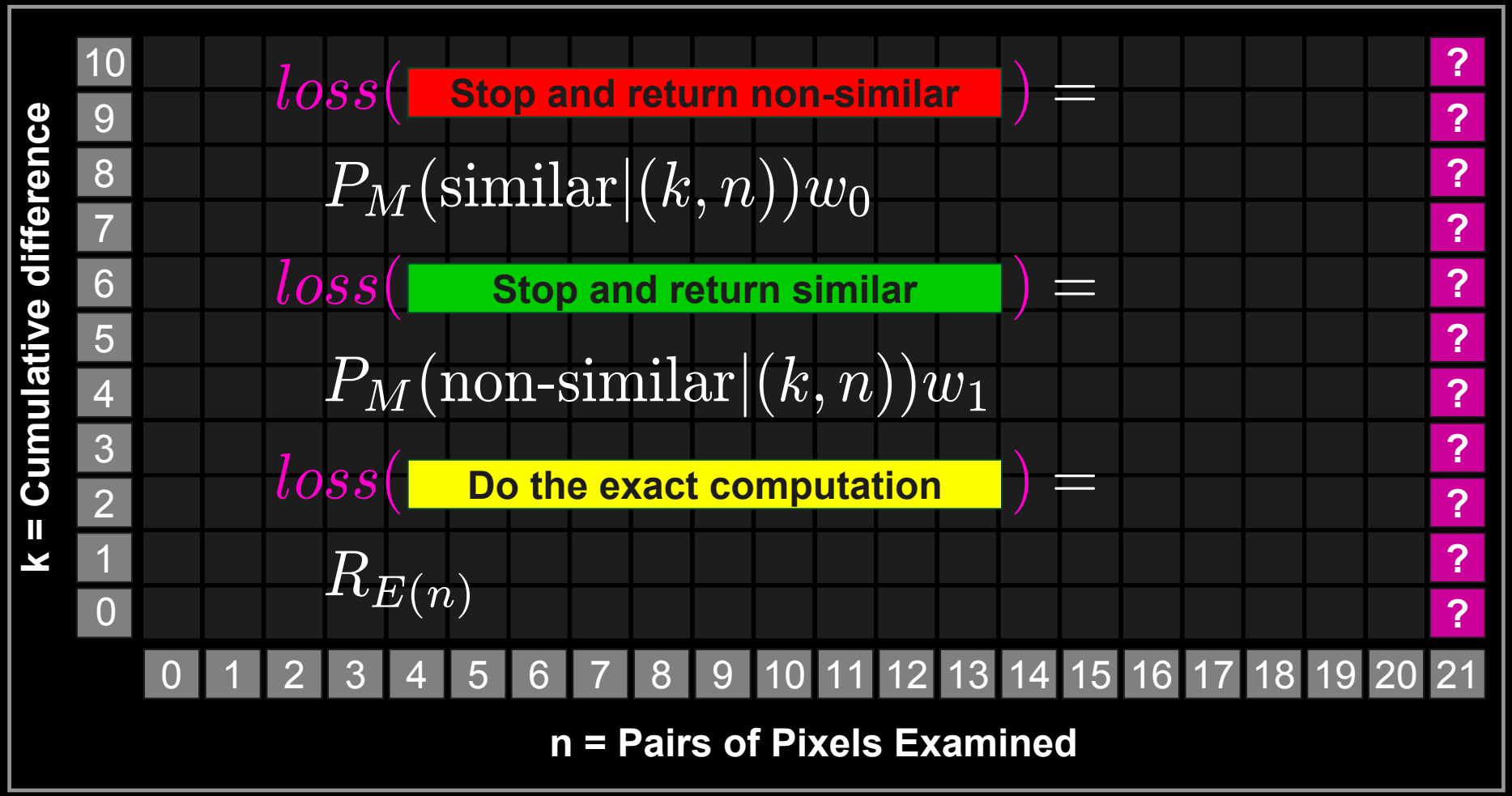
$$loss(M, w_0, w_1) =$$

$$E_M(\text{running time}) +$$

$$P_M(\text{false negative})P(\text{similar})w_0 +$$

$$P_M(\text{false positive})P(\text{non similar})w_1$$

Solving Auxiliary problem - Backward Induction



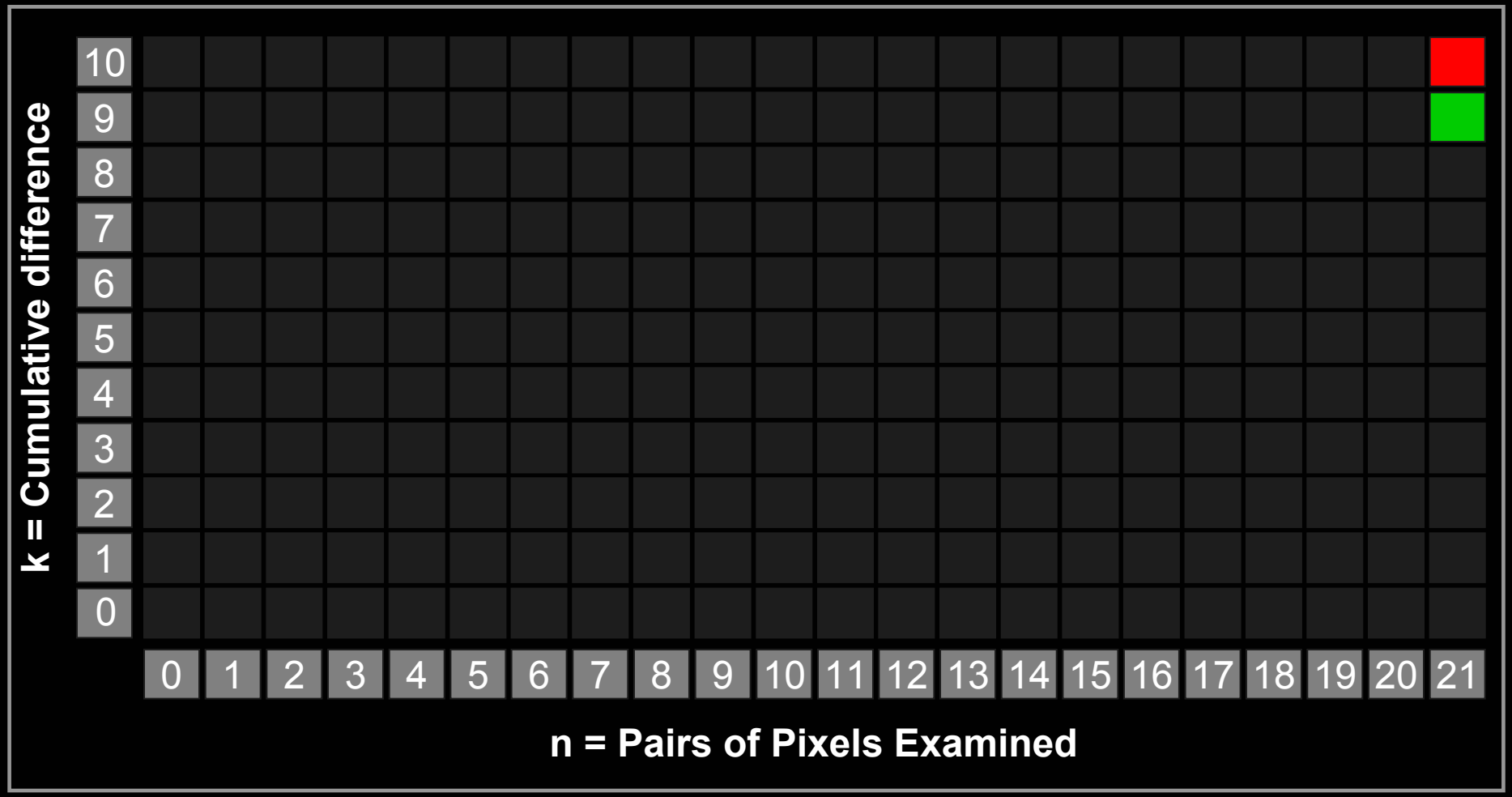
Stop and return non-similar

Stop and return similar

Continue sampling

Do the exact computation

Solving Auxiliary problem - Backward Induction



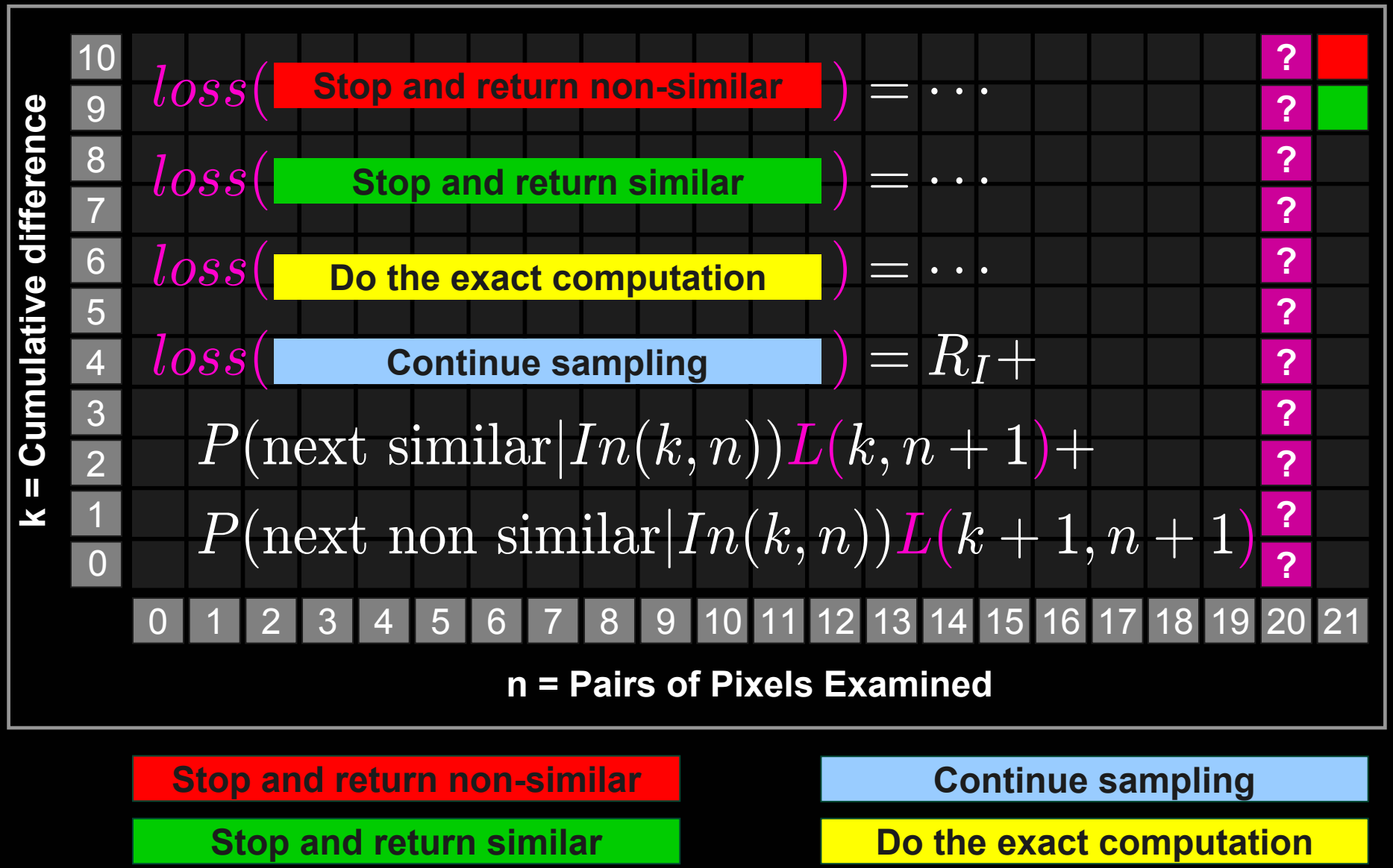
Stop and return non-similar

Continue sampling

Stop and return similar

Do the exact computation

Solving Auxiliary problem - Backward Induction



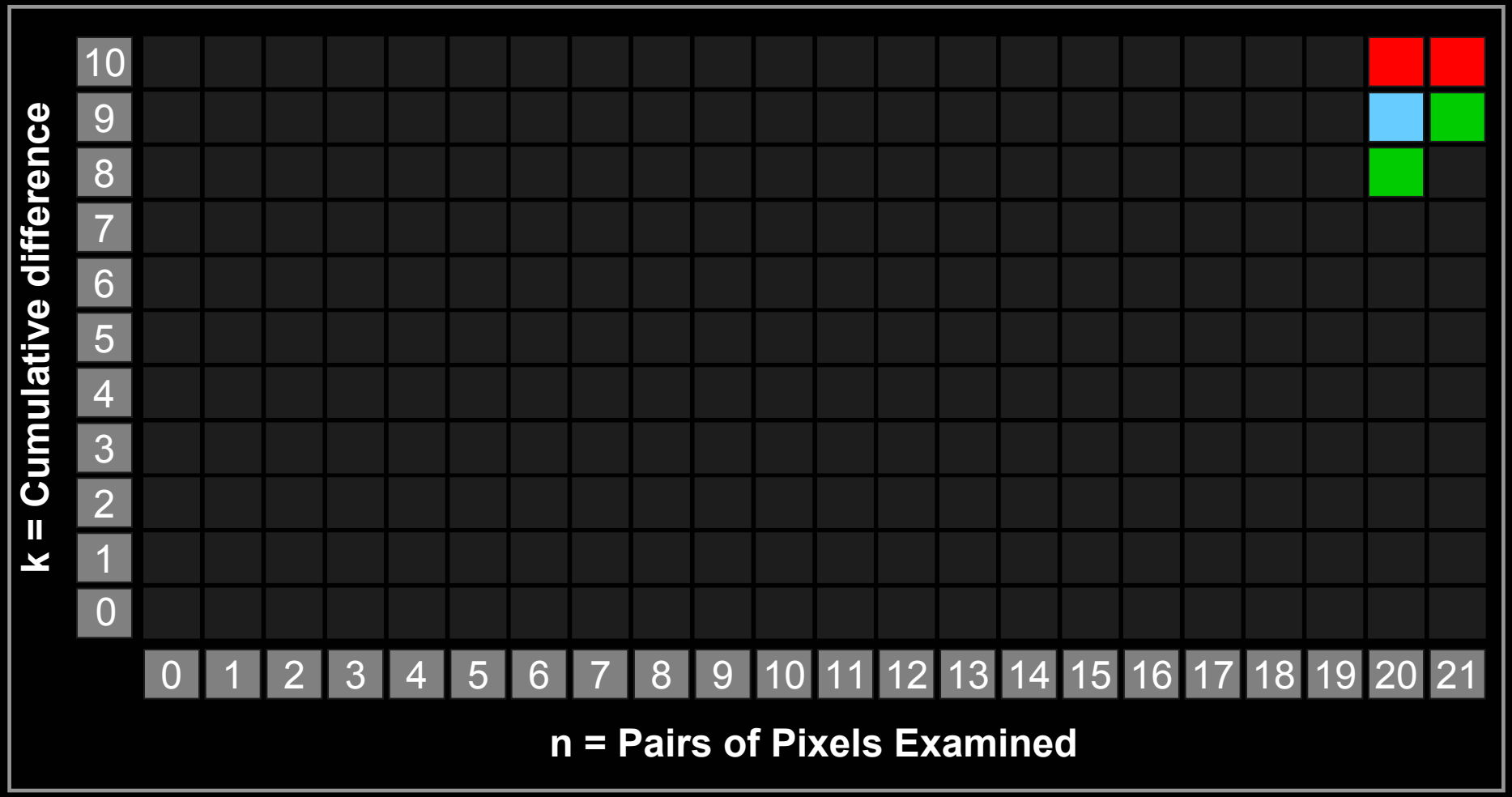
Stop and return non-similar

Continue sampling

Stop and return similar

Do the exact computation

Solving Auxiliary problem - Backward Induction



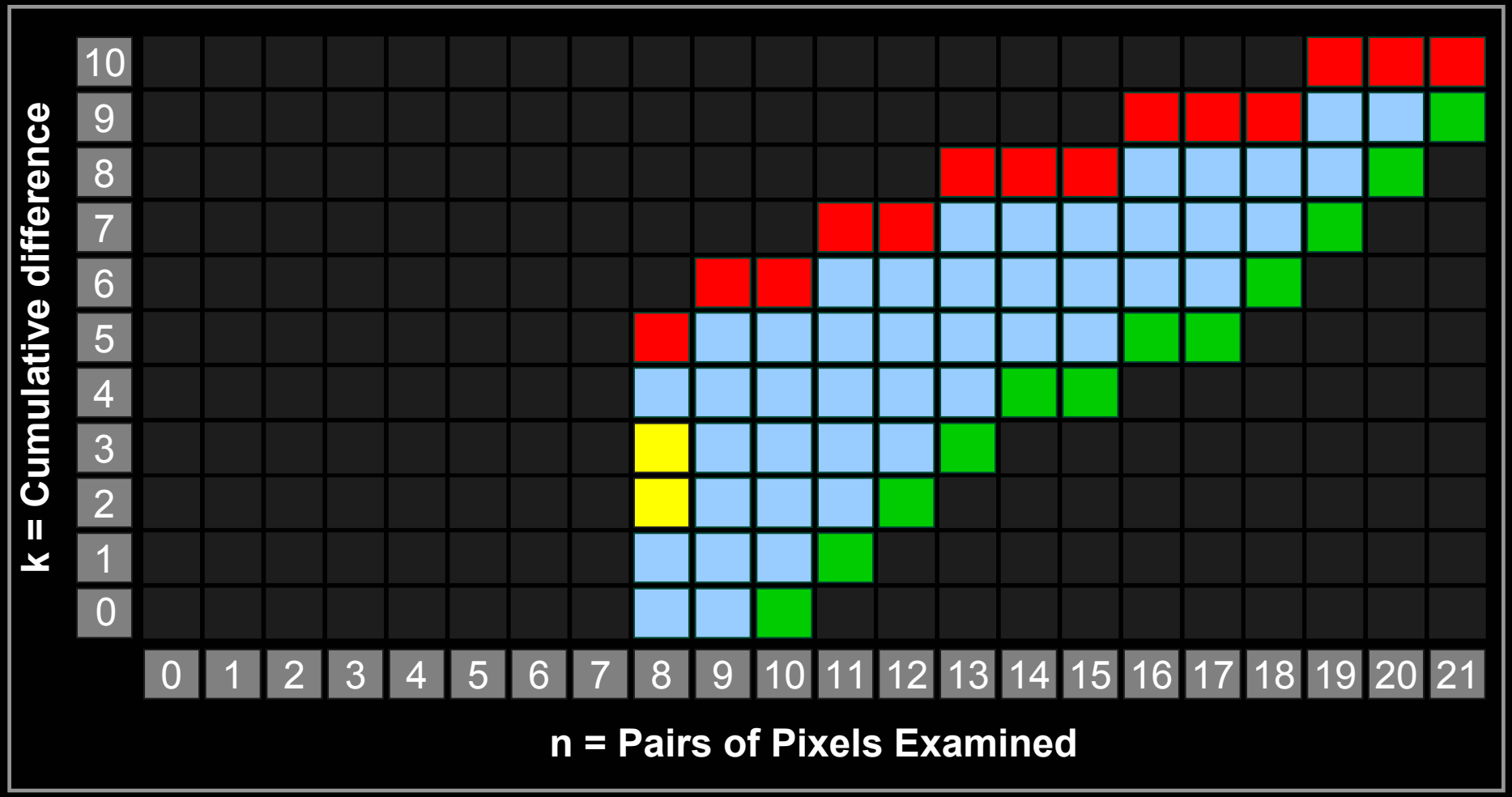
Stop and return non-similar

Continue sampling

Stop and return similar

Do the exact computation

Solving Auxiliary problem - Backward Induction



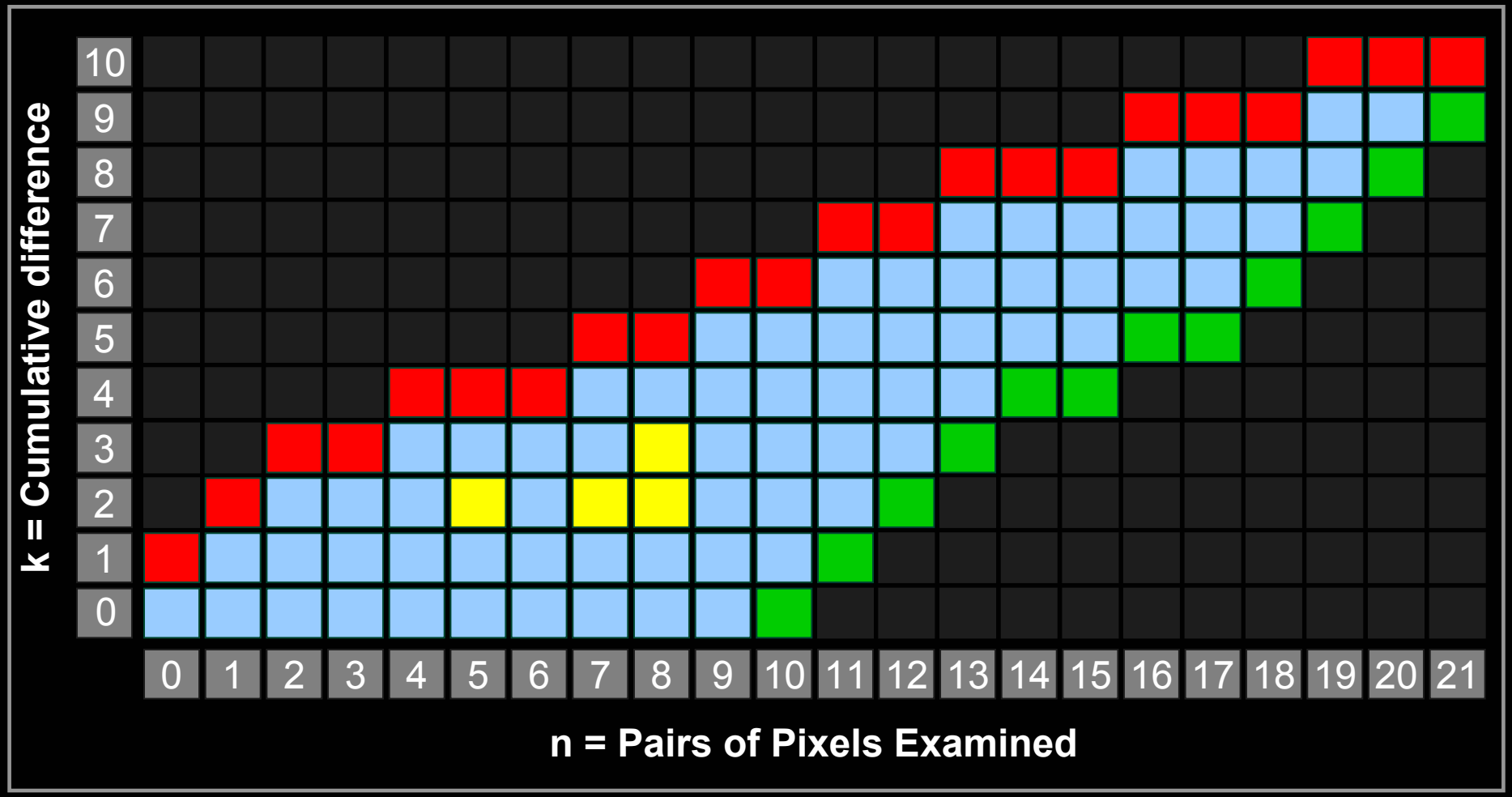
Stop and return non-similar

Continue sampling

Stop and return similar

Do the exact computation

Solving Auxiliary problem - Backward Induction



Stop and return non-similar

Continue sampling

Stop and return similar

Do the exact computation

Solving Optimization using Auxiliary problem

- Claim:

Let M^* be a decision matrix which is the solution to the auxilariy problem. Then it is the solution to the original minimization problem with
$\alpha = P_{M^*}(\text{false negative})$ and
$\beta = P_{M^*}(\text{false positive})$.

Solving Optimization using Auxiliary problem

- Claim:

Let M^* be a decision matrix which is the solution to the auxilariy problem. Then it is the solution to the original minimization problem with
$\alpha = P_{M^*}(\text{false negative})$ and
$\beta = P_{M^*}(\text{false positive})$.

Solving Optimization using Auxiliary problem

- Proof:

Let M' be another decision matrix of same size and smaller/equal error probabilities. Then:

Solving Optimization using Auxiliary problem

$$loss(M^*, w_0, w_1) = P_{M^*}(\text{false negative})P(\text{similar})w_0 + P_{M^*}(\text{false positive})P(\text{non similar})w_1 + E_{M^*}[N]R_I + P_{M^*}(\text{compute exact})R_{E(n)}$$

Solving Optimization using Auxiliary problem

$$
\begin{aligned}
loss(M^*, w_0, w_1) = {} & P_{M^*}(\text{false negative})P(\text{similar})w_0 + \\
& P_{M^*}(\text{false positive})P(\text{non similar})w_1 + \\
& E_{M^*}[N]R_I + P_{M^*}(\text{compute exact})R_{E(n)} \\
\leq {} & P_{M'}(\text{false negative})P(\text{similar})w_0 + \\
& P_{M'}(\text{false positive})P(\text{non similar})w_1 + \\
& E_{M'}[N]R_I + P_{M'}(\text{compute exact})R_{E(n)}
\end{aligned}
$$

M^* is optimal

Solving Optimization using Auxiliary problem

$$
\begin{aligned}
loss(M^*, w_0, w_1) = {} & P_{M^*}(\text{false negative})P(\text{similar})w_0 + \\
& P_{M^*}(\text{false positive})P(\text{non similar})w_1 + \\
& E_{M^*}[N]R_I + P_{M^*}(\text{compute exact})R_{E(n)} \\
\leq {} & P_{M'}(\text{false negative})P(\text{similar})w_0 + \\
& P_{M'}(\text{false positive})P(\text{non similar})w_1 + \\
& E_{M'}[N]R_I + P_{M'}(\text{compute exact})R_{E(n)} \\
\leq {} & P_{M^*}(\text{false negative})P(\text{similar})w_0 + \\
& P_{M^*}(\text{false positive})P(\text{non similar})w_1 + \\
& E_{M'}[N]R_I + P_{M'}(\text{compute exact})R_{E(n)}
\end{aligned}
$$

M' has smaller/equal error probabilities.

Solving Optimization using Auxiliary problem

$$
\begin{aligned}
loss(M^*, w_0, w_1) = {} & P_{M^*}(\text{false negative})P(\text{similar})w_0 + \\
& P_{M^*}(\text{false positive})P(\text{non similar})w_1 + \\
& E_{M^*}[N]R_I + P_{M^*}(\text{compute exact})R_{E(n)} \\
\leq {} & P_{M'}(\text{false negative})P(\text{similar})w_0 + \\
& P_{M'}(\text{false positive})P(\text{non similar})w_1 + \\
& E_{M'}[N]R_I + P_{M'}(\text{compute exact})R_{E(n)} \\
\leq {} & P_{M^*}(\text{false negative})P(\text{similar})w_0 + \\
& P_{M^*}(\text{false positive})P(\text{non similar})w_1 + \\
& E_{M'}[N]R_I + P_{M'}(\text{compute exact})R_{E(n)}
\end{aligned}
$$

$$
\begin{aligned}
& E_{M^*}[N]R_I + P_{M^*}(\text{compute exact})R_{E(n)} \leq \\
& E_{M'}[N]R_I + P_{M'}(\text{compute exact})R_{E(n)}
\end{aligned}
$$

Solving Optimization using Auxiliary problem

- Claim:

Let M^* be the optimal decision matrix returned by the backward induction algorithm, for some w_0, w_1.

If $w_0 = \frac{R_{E(0)}}{\alpha P(\text{similar})}$ then $P_{M^*}(\text{false negative}) \leq \alpha$.

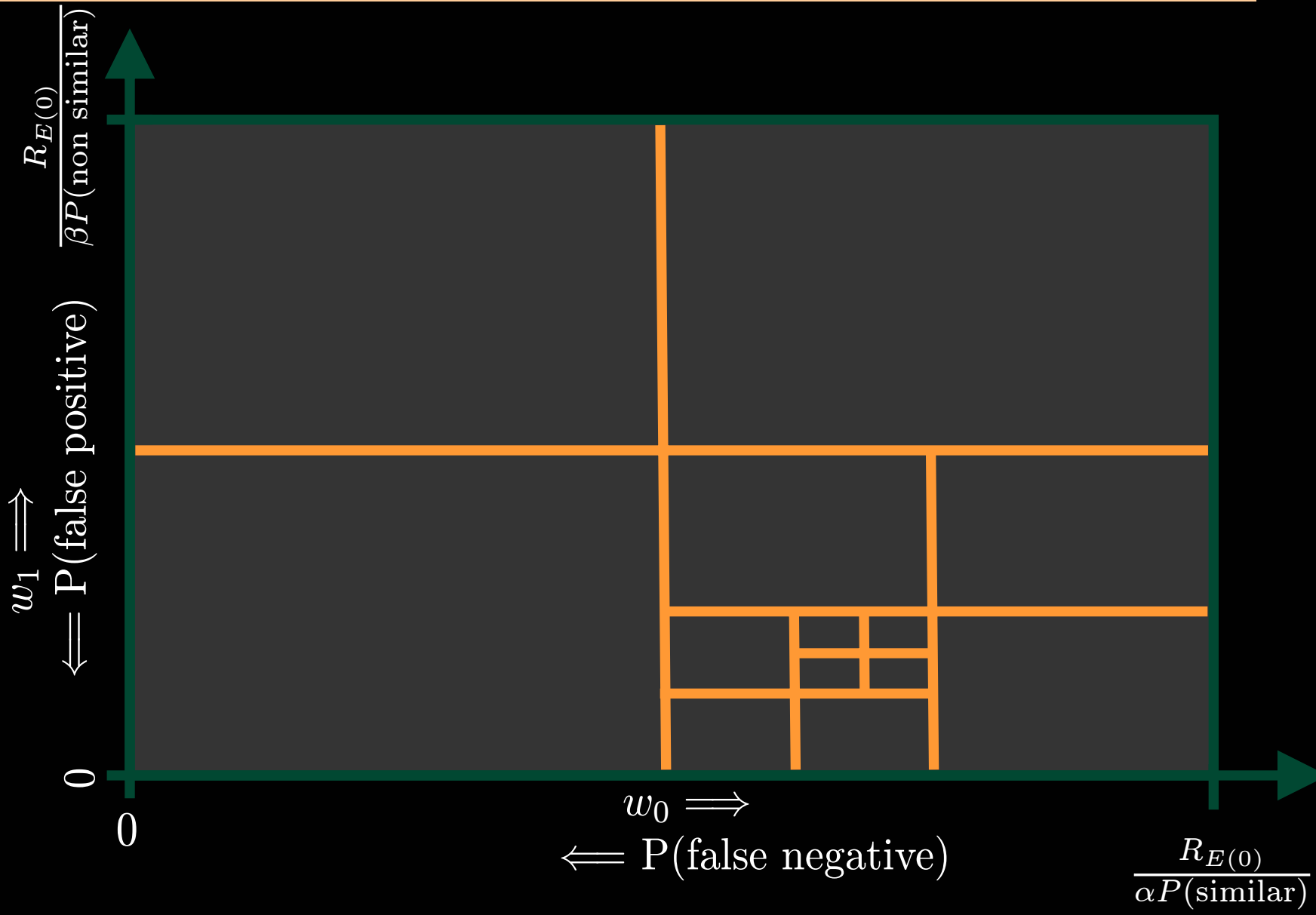
If $w_1 = \frac{R_{E(0)}}{\beta P(\text{non similar})}$ then $P_{M^*}(\text{false positive}) \leq \beta$

Solving Optimization using Auxiliary problem

Proof: Let M' be a decision matrix, where $M'[0,0] = E$:

$$R_{E(0)} = Loss(M', w_0, w_1)$$

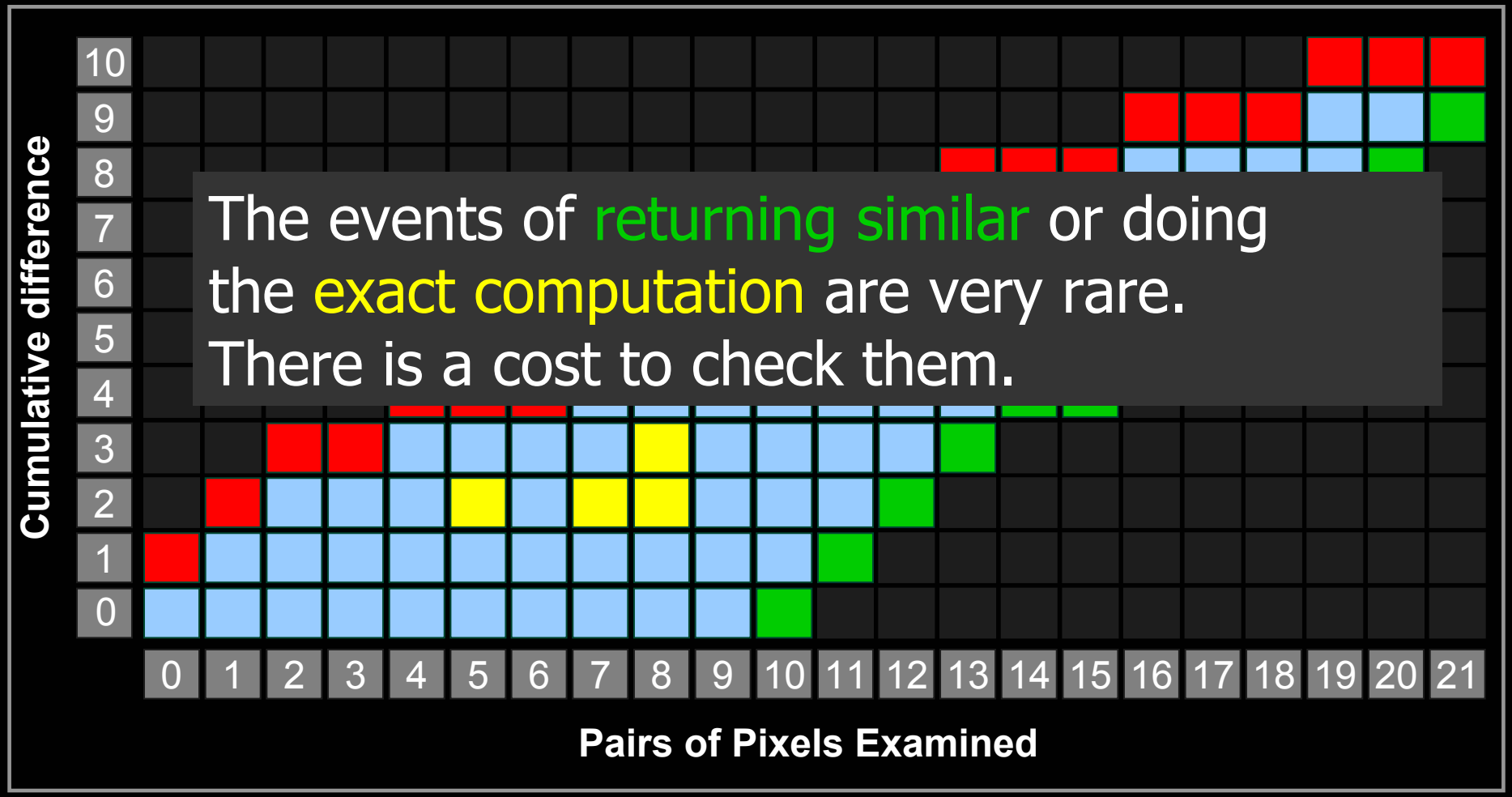
Finding the Right Error Weights



Offline Time Complexity

- $\Omega_S[k, n]$ and $\Omega_{NS}[k, n]$ are computed for all possible k and n with a dynamic programming algorithm with time complexity of $O(B^2|A|)$.
- The backward induction algorithm has time complexity of $O(B^2)$.
- Practical run time of several hours for $|A| = 2197$

Practical Observation



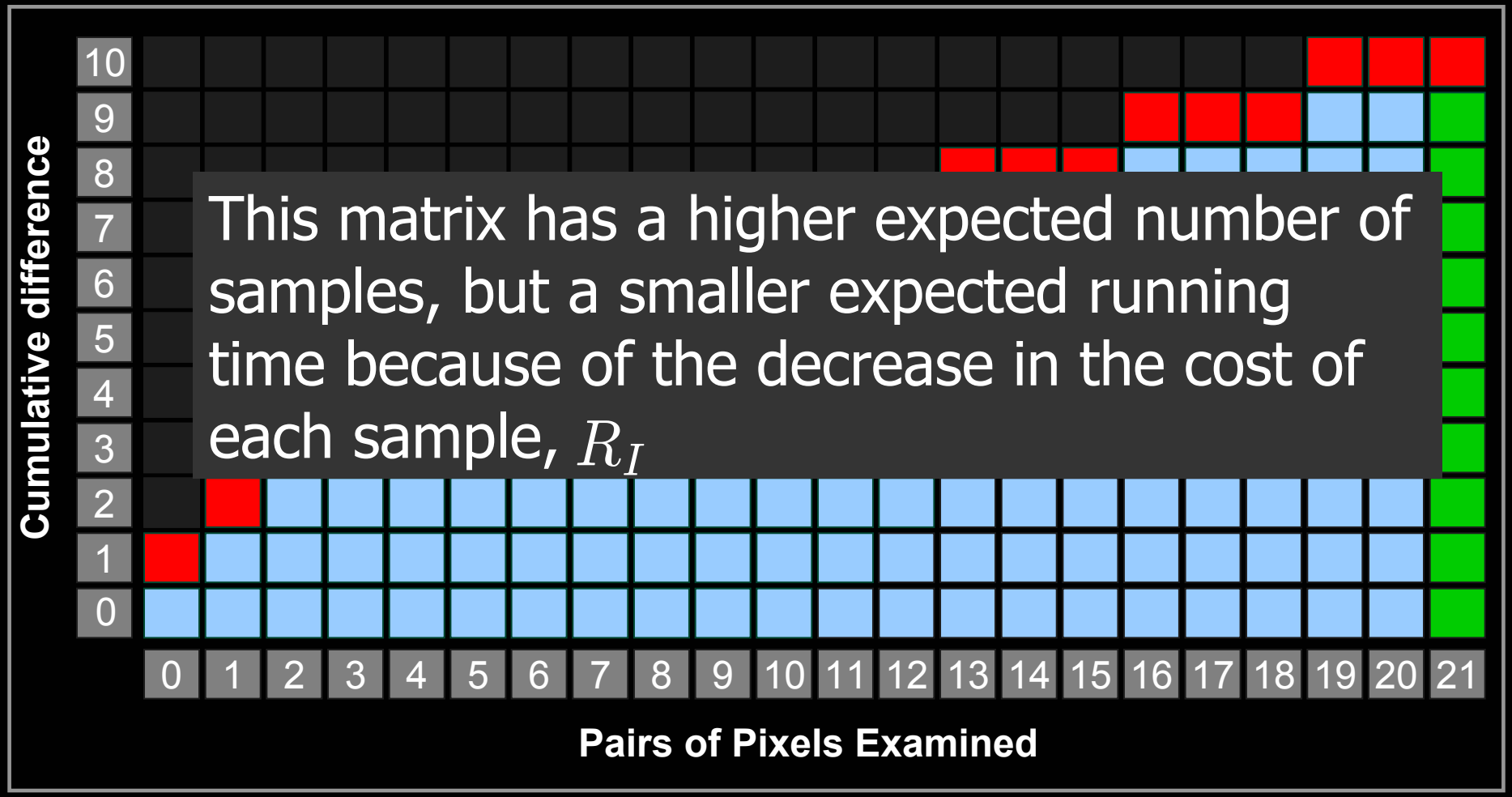
Stop and return non-similar

Continue sampling

Stop and return similar

Do the exact computation

Practical Observation



Stop and return non-similar

Continue sampling

Stop and return similar

Do the exact computation

Can we Reduce Offline Time ?

SPRT for Composite Hypotheses

- Sample both images as long as:

$$\mathcal{A} < \lambda(In(k,n)) < \mathcal{B}$$

SPRT for Composite Hypotheses

- Wald's approximation:

$$\mathcal{A} = \frac{\beta}{1-\alpha} \qquad \mathcal{B} = \frac{1-\beta}{\alpha}$$

$$\alpha = \text{Maximum false negative}$$

$$\beta = \text{Maximum false positive}$$

SPRT for Composite Hypotheses

- Computation of likelihood ratio:

$$\lambda(In(k,n)) = \frac{P(In(k,n)|D > t)}{P(In(k,n)|D \leq t)}$$

$$= \frac{\sum_{d=0}^{t} P(In(k,n), D = d|D > t)}{\sum_{d=t+1}^{|A|} P(In(k,n), D = d|D \leq t)}$$

$$= \frac{\sum_{d=0}^{t} P(In(k,n), D = d)P(D \leq t)}{\sum_{d=t+1}^{|A|} P(In(k,n), D = d)P(D > t)}$$

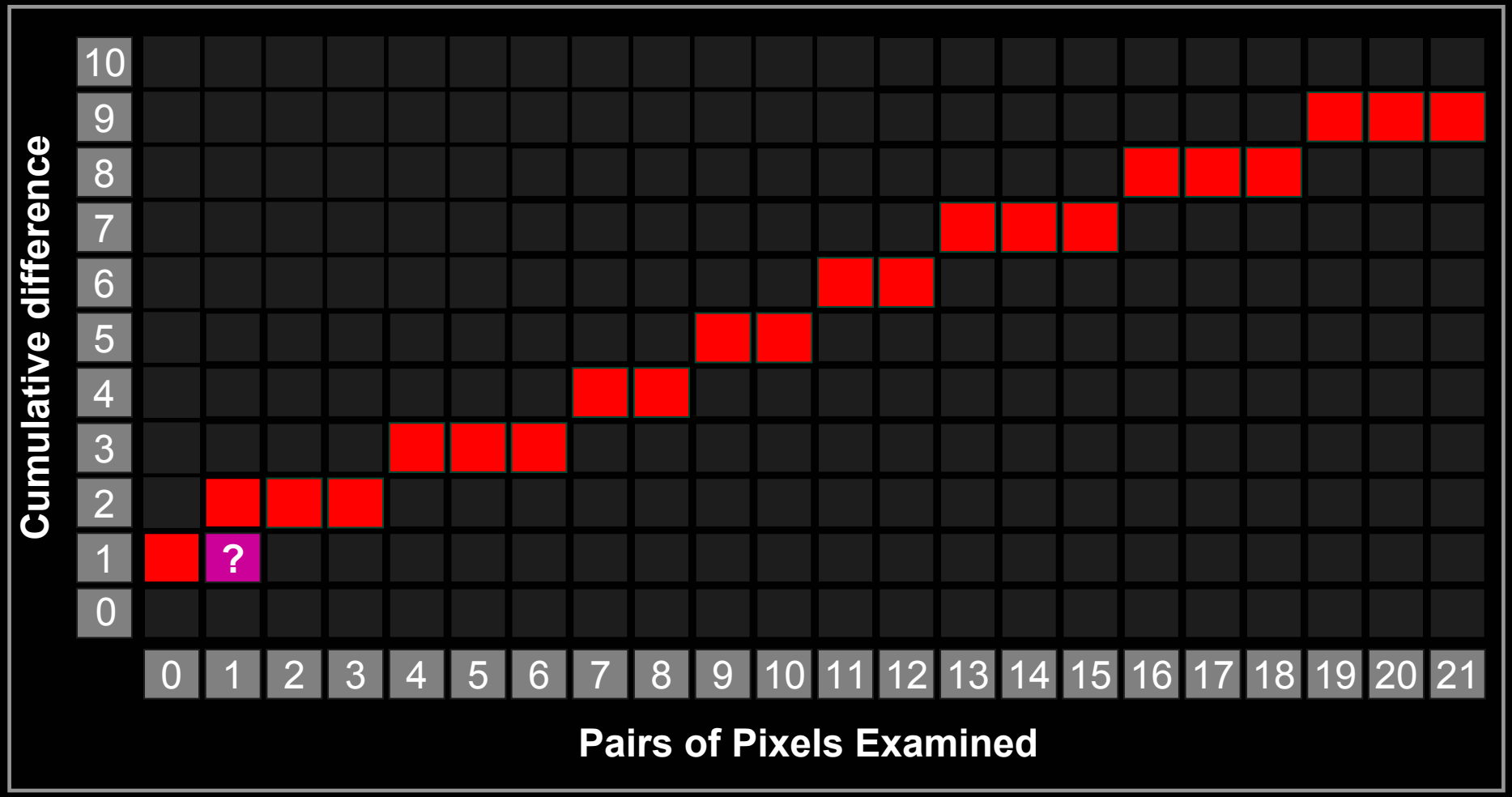
$$= \frac{\sum_{d=0}^{t} P(D = d|In(k,n))P(D \leq t)}{\sum_{d=t+1}^{|A|} P(D = d|In(k,n))P(D > t)}$$

Numerical stability

SPRT for Composite Hypotheses

- Assuming correct prior, our hypotheses are actually simple – there are no free parameters
- The proof for sub-optimality of SPRT is valid
- Sub-optimal because of "overshoot" effect
- Excellent practical results

Designing the SPRT



Stop and return non-similar

Continue sampling

Stop and return similar

Do the exact computation

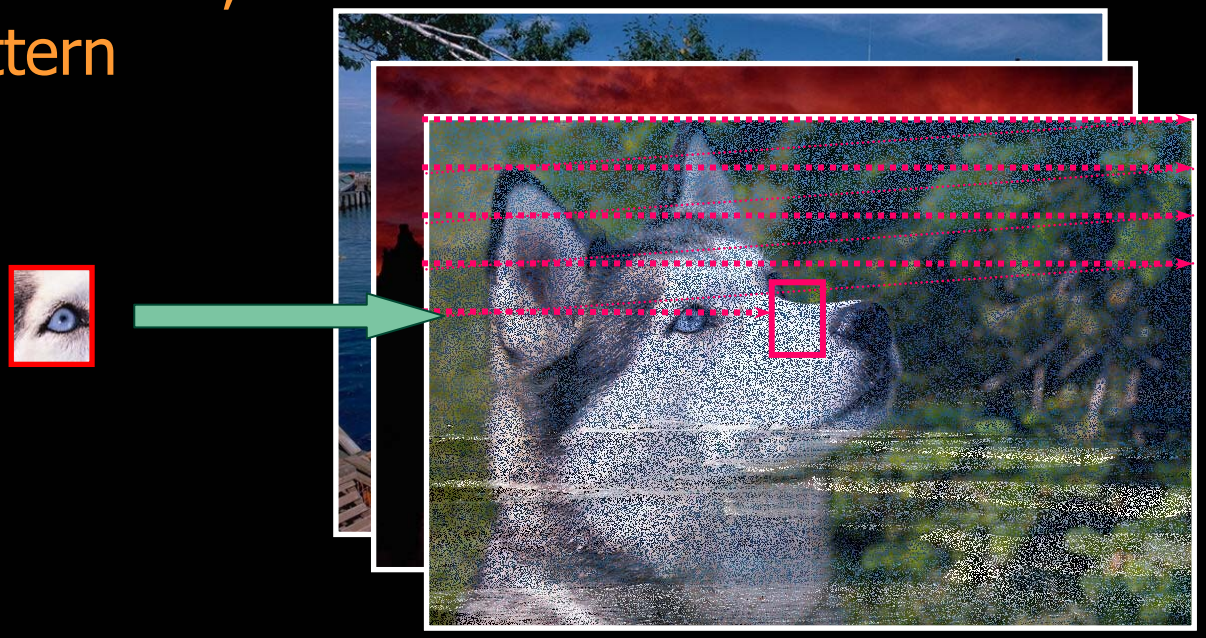
SPRT Offline Time Complexity

- $P(D = d|In(k, n))$ is updated each stage.
- $O(|A|^2)$ time complexity and $O(|A|)$ memory complexity
- Practical time of 0.067 seconds for $|A| = 2197$

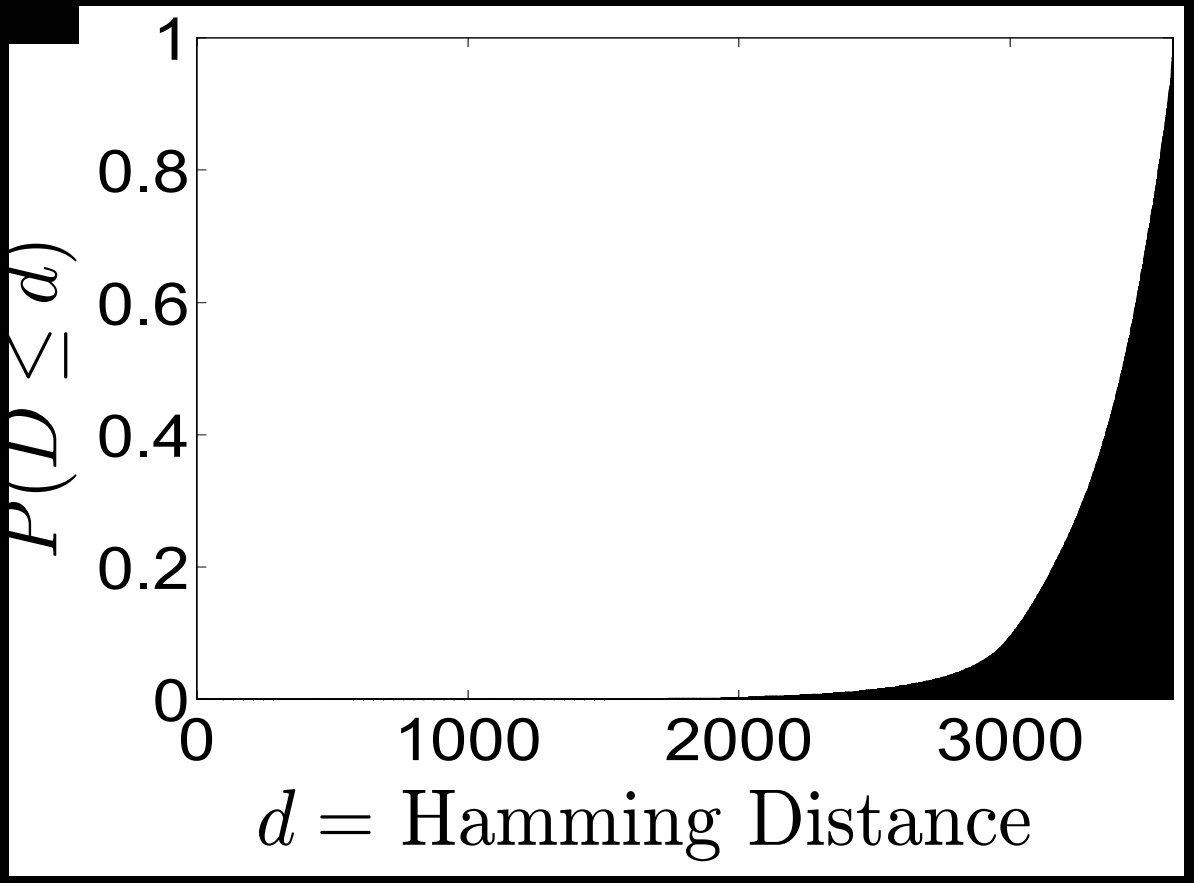
Estimating the Prior

Random,
not too smooth,
pattern

480 Training images



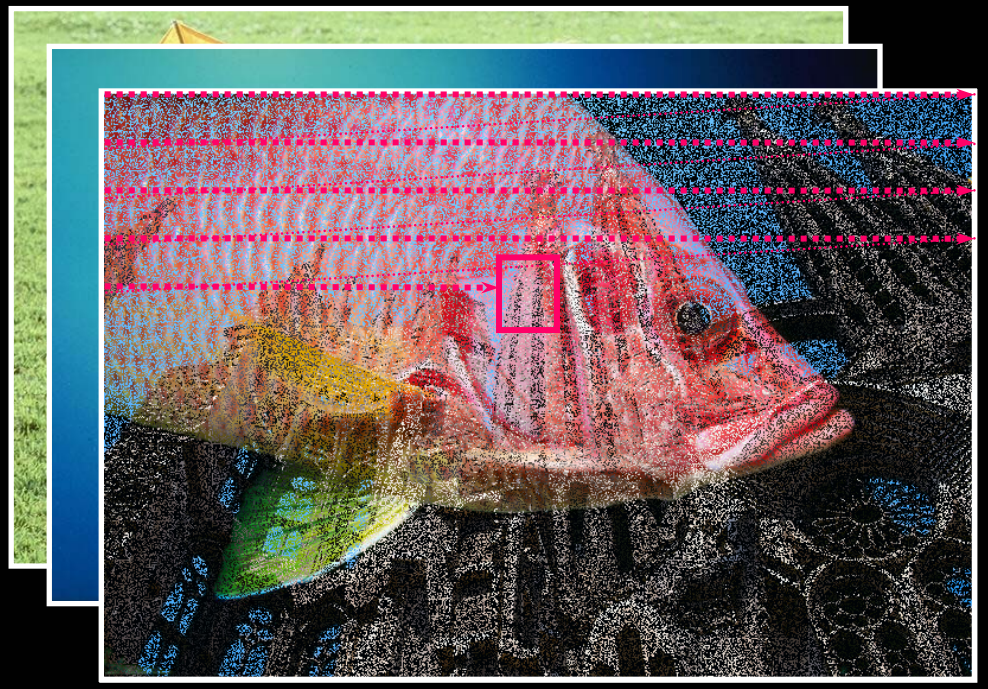
Estimating the Prior



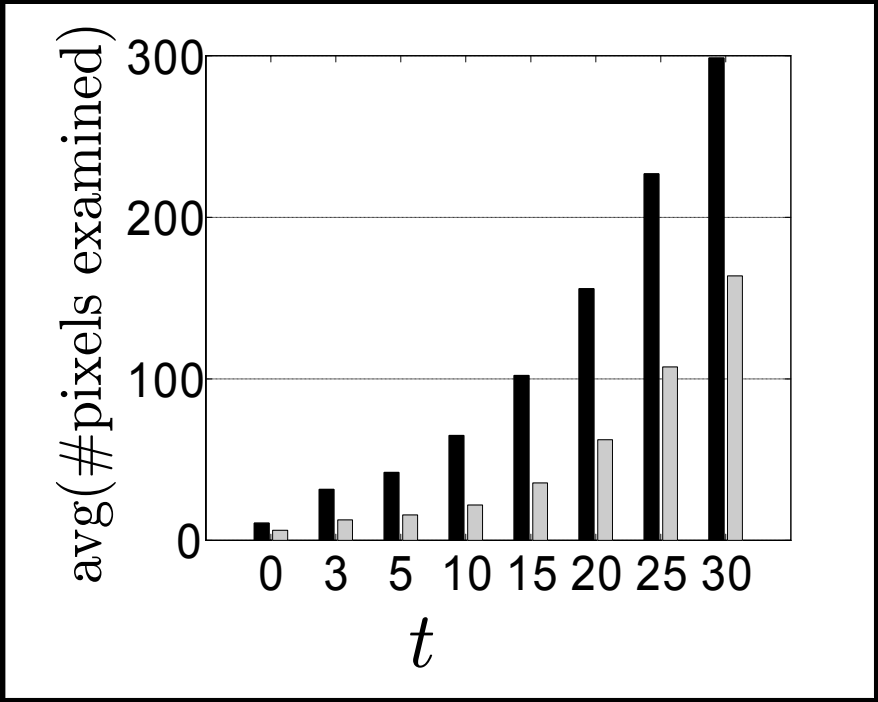
Experiments That Checks Sampling Performance

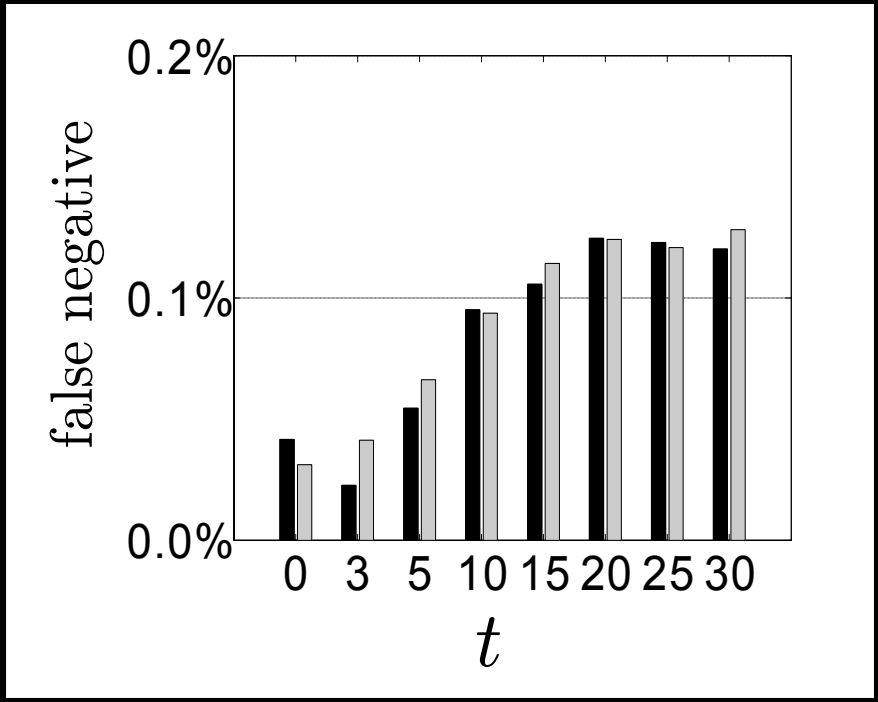
Random,
not too smooth,
pattern

480 Test Images



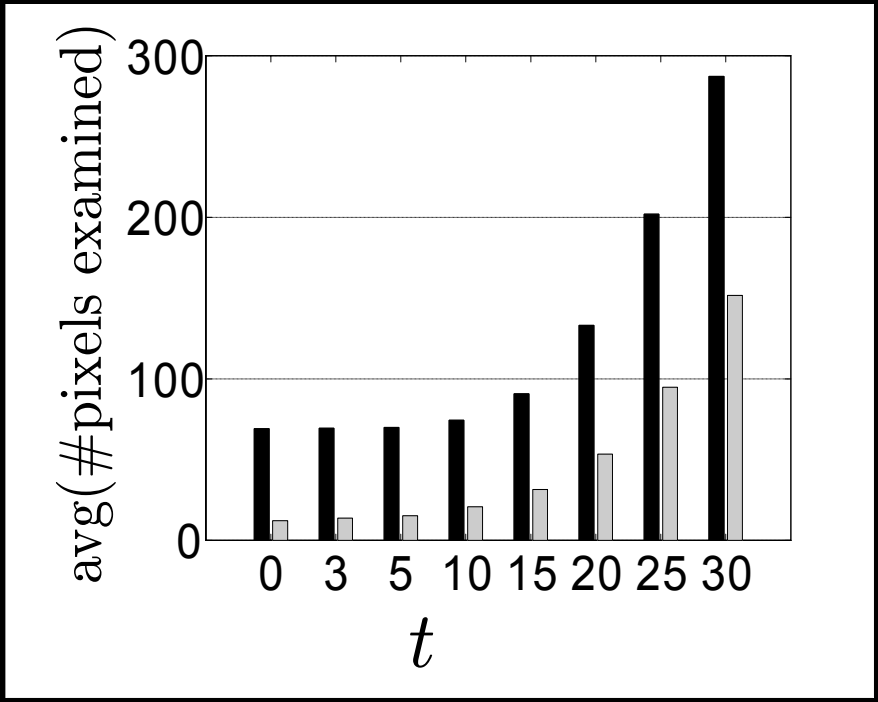
Fixed Size Sampling vs Sequential Sampling

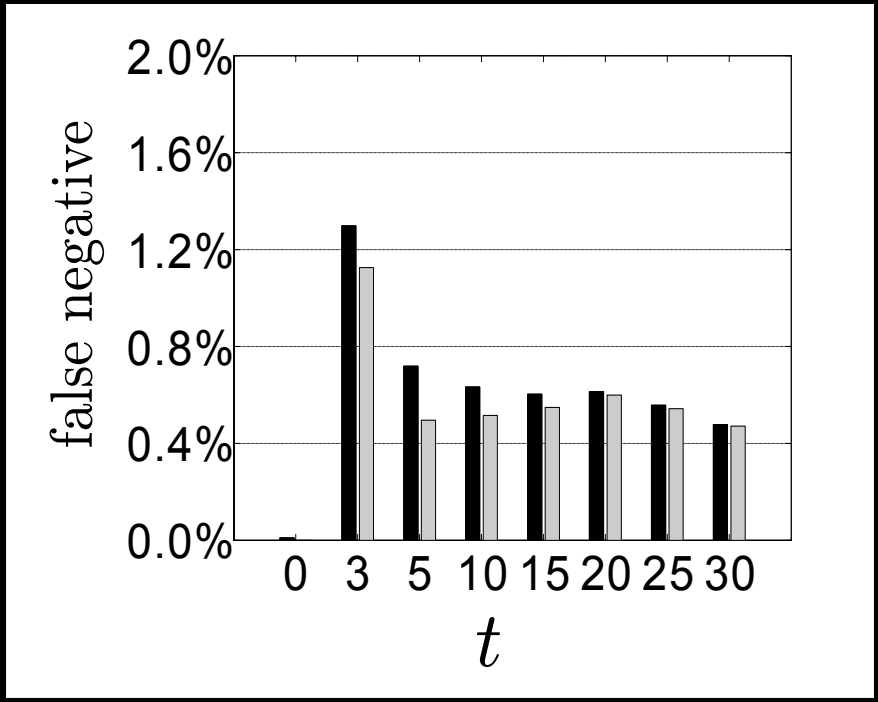




- 30x30 pattern size
- Thresholded Absolute Difference Distance
- **Estimated prior**

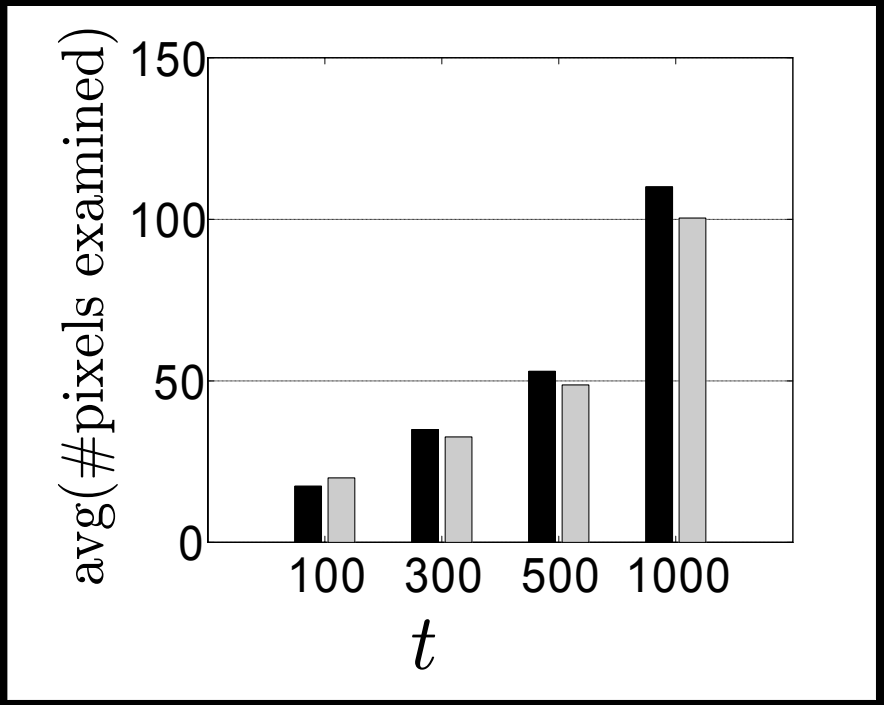
Fixed Size Sampling vs Sequential Sampling

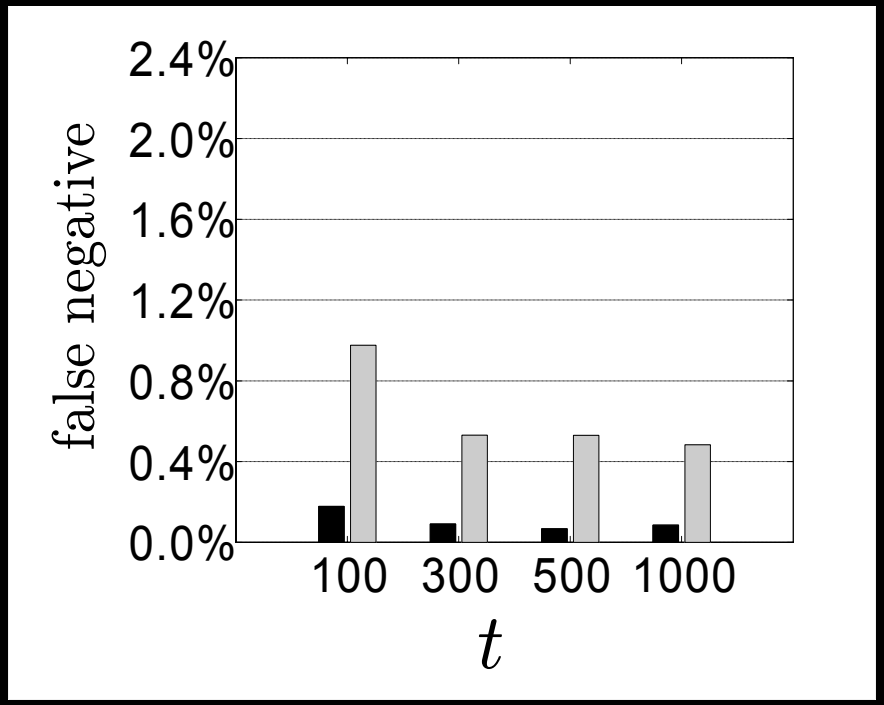




- 30x30 pattern size
- Thresholded Absolute Difference Distance
- **Uniform prior**

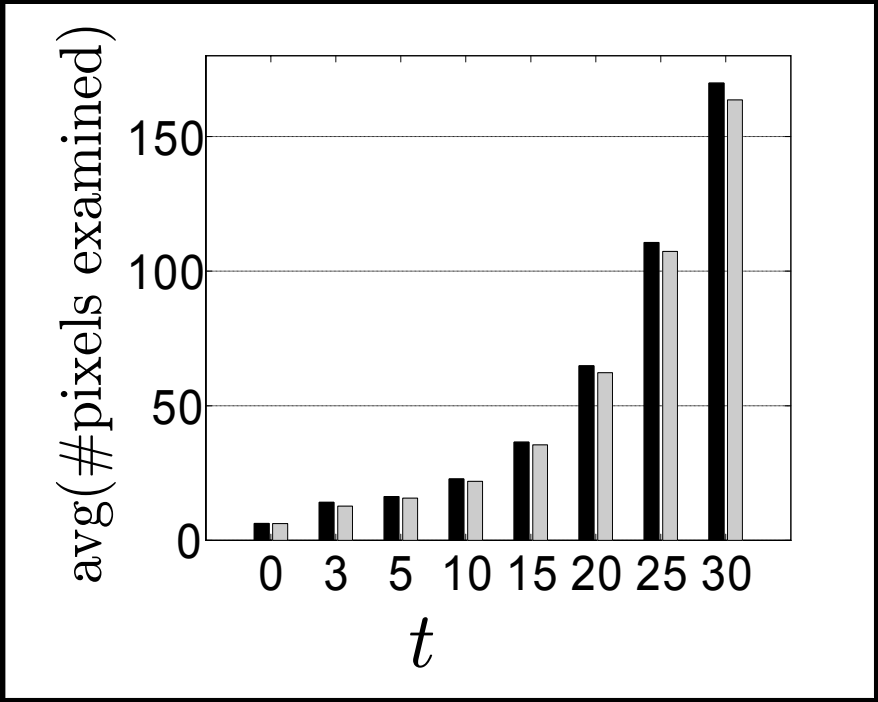
Estimated Prior vs Uniform Prior

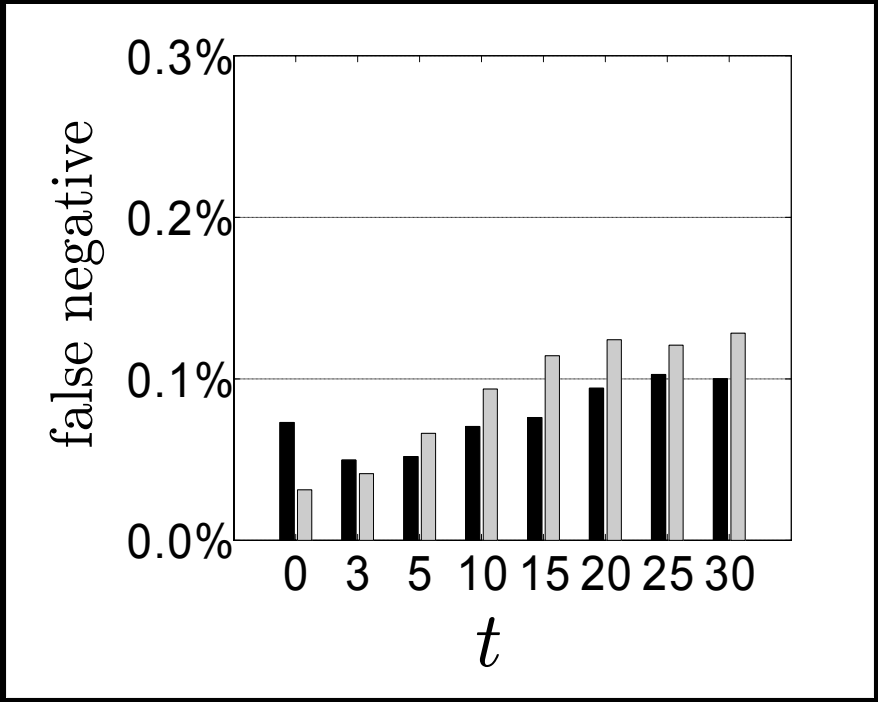




- 60x60 pattern size
- Threshold l_2 norm in L*a*b color space
- OPT

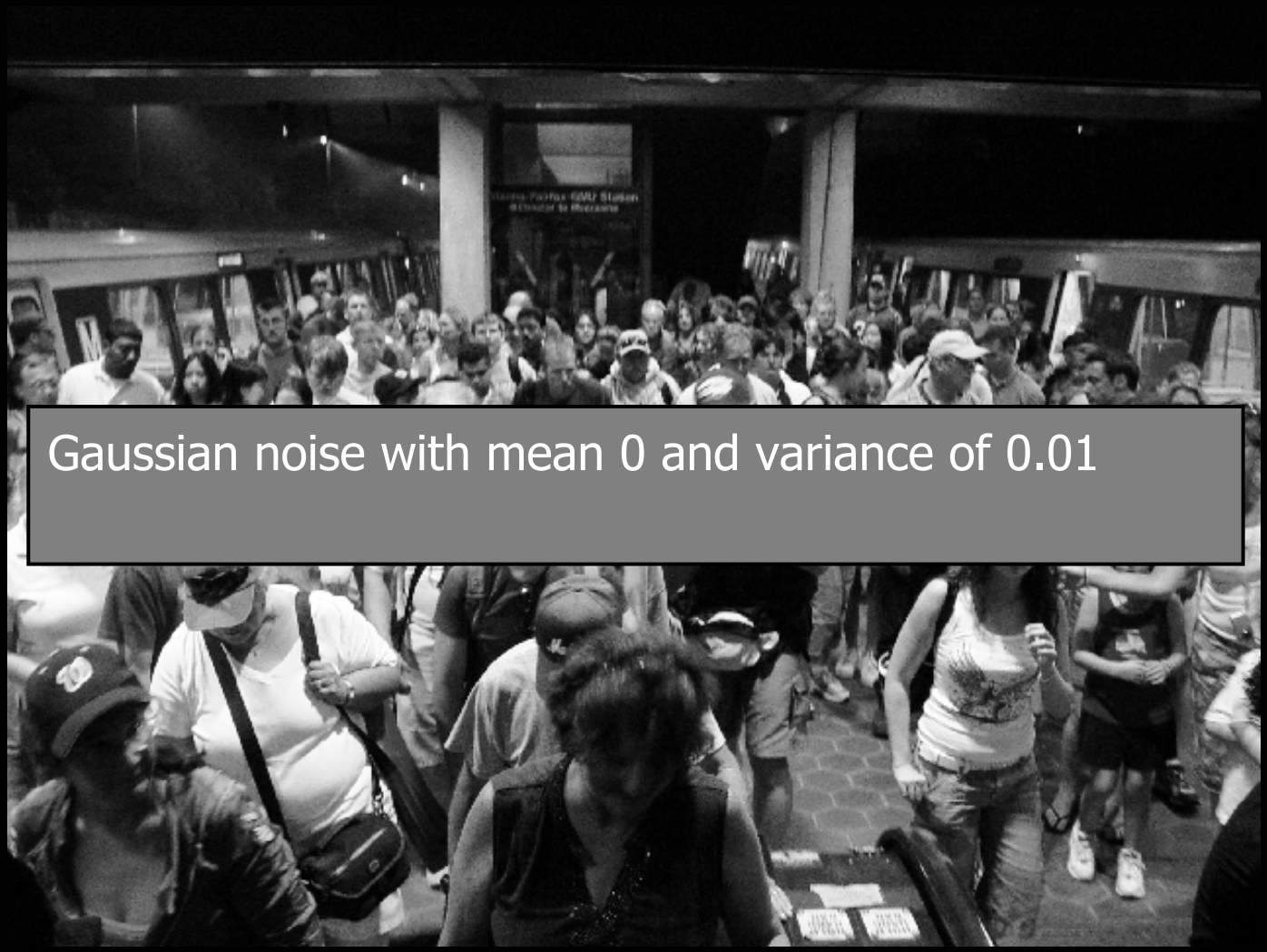
SPRT vs OPT





- 30x30 pattern size
- Thresholded Absolute Difference Distance
- Estimated prior

Gaussian Noise



- Image copyright by Ben Schumin. Taken from: http://en.wikipedia.org/wiki/Image:July_4_crowd_at_Vienna_Metro_station.jpg

Gaussian Noise

- Image copyright by Ben Schumin. Taken from: http://en.wikipedia.org/wiki/Image:July_4_crowd_at_Vienna_Metro_station.jpg

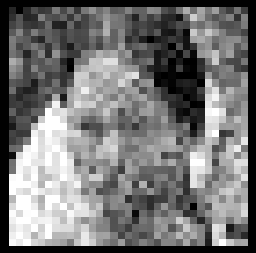
Gaussian Noise

- Image copyright by Ben Schumin. Taken from: http://en.wikipedia.org/wiki/Image:July_4_crowd_at_Vienna_Metro_station.jpg

Gaussian Noise

Pattern

Occurrences of pattern in image



- Image copyright by Ben Schumin. Taken from: http://en.wikipedia.org/wiki/Image:July_4_crowd_at_Vienna_Metro_station.jpg

Distance Used

$$\delta(|gray(Im_1(x,y)) - gray(Im_2(x,y))| > 20)$$

Performance

- Image Size: 640x480 pixels
- Pattern Size: 1089 pixels
- Average Sample Size: 17.86 pixels
- Offline Time: 0.018 sec
- Online Time: **0.019 sec**

Light Changes + Rotation

Light Changes + Rotation

Light Changes + Rotation

Pattern

Occurrences of pattern in image

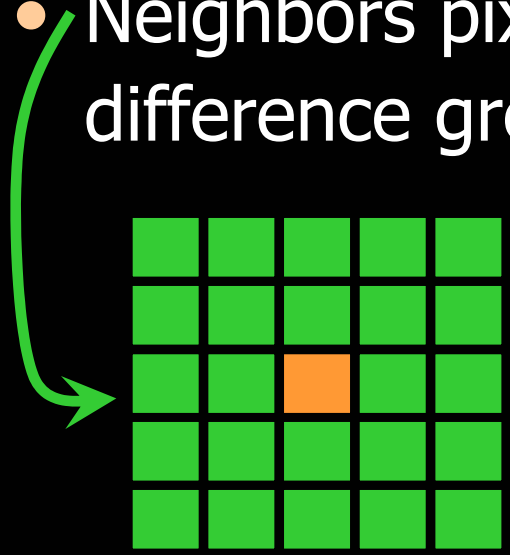
Light Changes + Rotation

Pattern

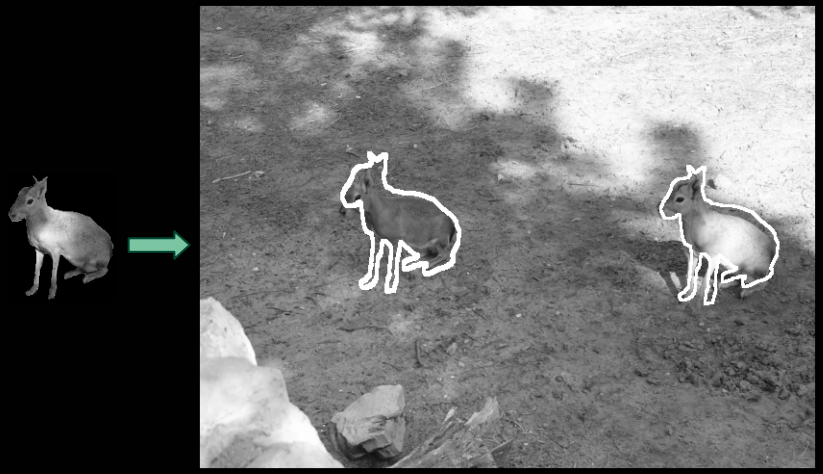
Occurrences of pattern in image

Distance Used

- Monotonic Relations
- Neighbors pixels with absolute intensity value difference greater than 80



Performance



- Image Size: 640x480 pixels
- Pattern Size: 631 pairs of pixels
- Average Sample Size: 35.28 pairs of pixels
- Offline Time: 0.009 sec
- Online Time: **0.021 sec**

Occlusion + Rotation

Occlusion + Rotation

Occlusion + Rotation

Pattern

Occurrences of pattern in images

Occlusion + Rotation

Pattern

Occurrences of pattern in images

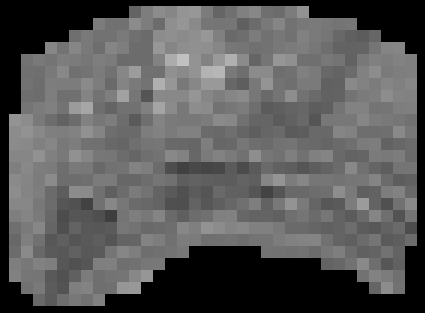
Distance Used

$$\delta(|gray(Im_1(x,y)) - gray(Im_2(x,y))| > 20)$$

Performance

- Image Size: 640x480 pixels
- Pattern Size: 2197 pixels
- Average Sample Size: 19.7 pixels
- Offline Time: 0.067 sec
- Online Time: **0.022 sec**
- No motion considerations.

Non Rigid Deformations

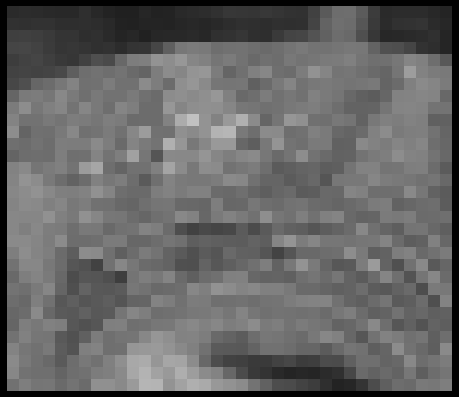


Non Rigid Deformations

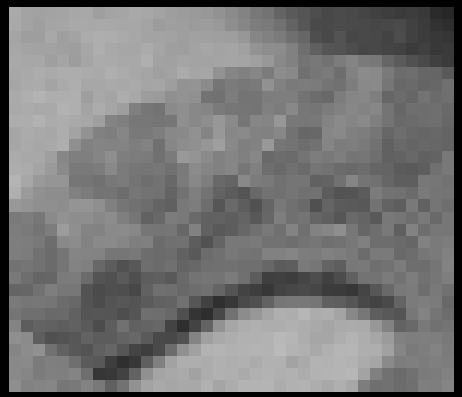
Non Rigid Deformations

Non Rigid Deformations

Pattern

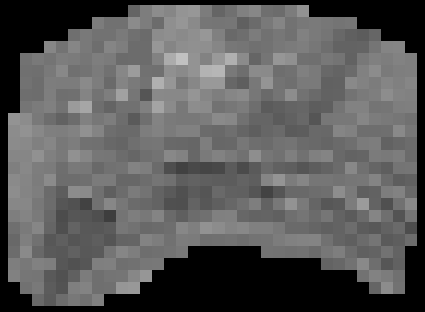


Occurrence of pattern in image

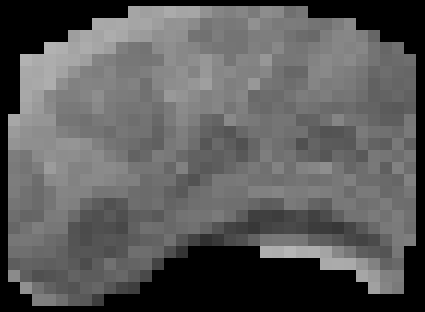


Non Rigid Deformations

Pattern

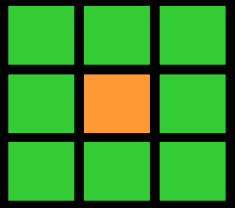


Occurrence of pattern in image



Distance Used

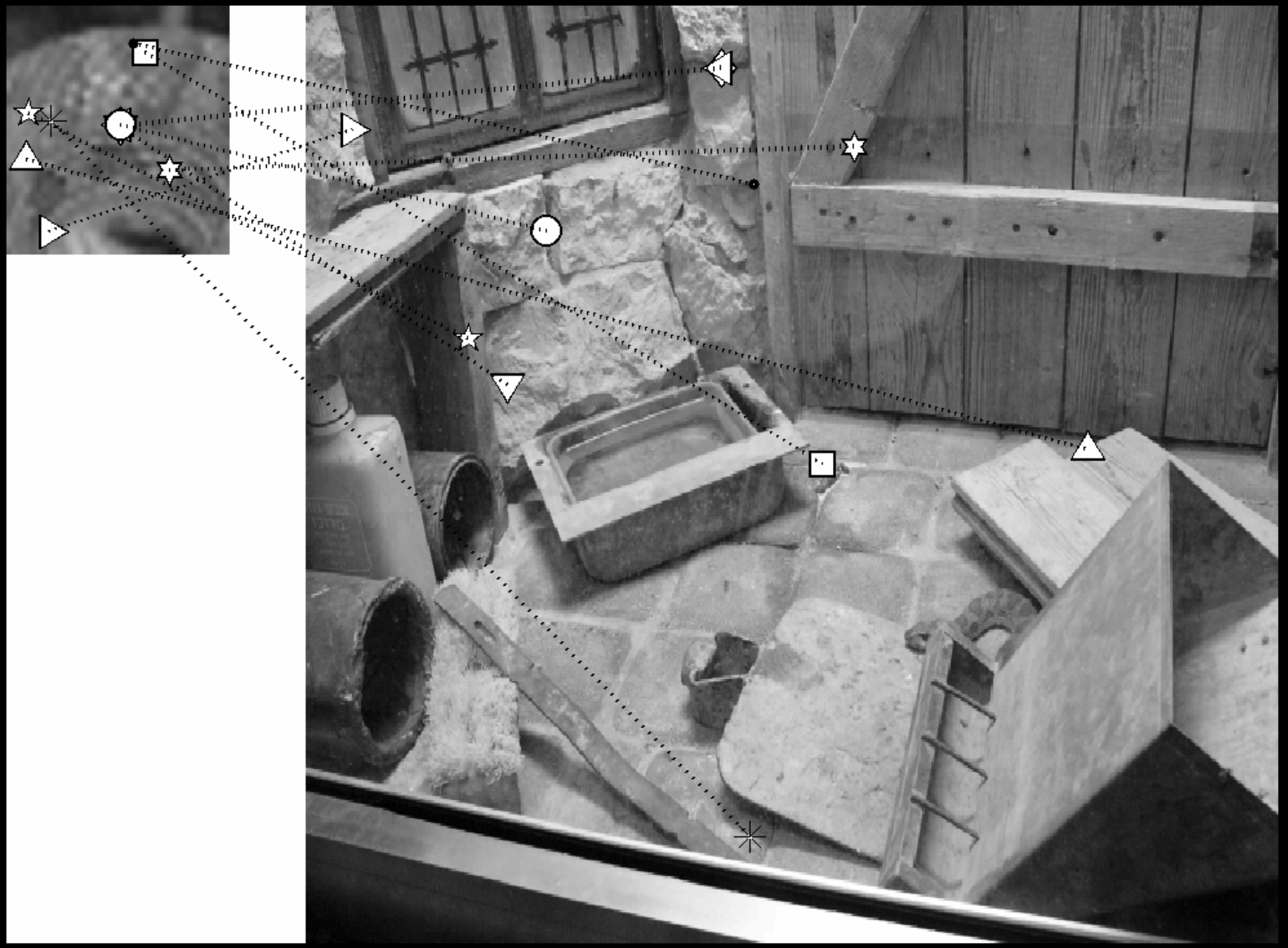
$$\delta(|gray(Im_1(x,y)) - gray(Im_2(x \pm 1, y \pm 1))| > 20)$$



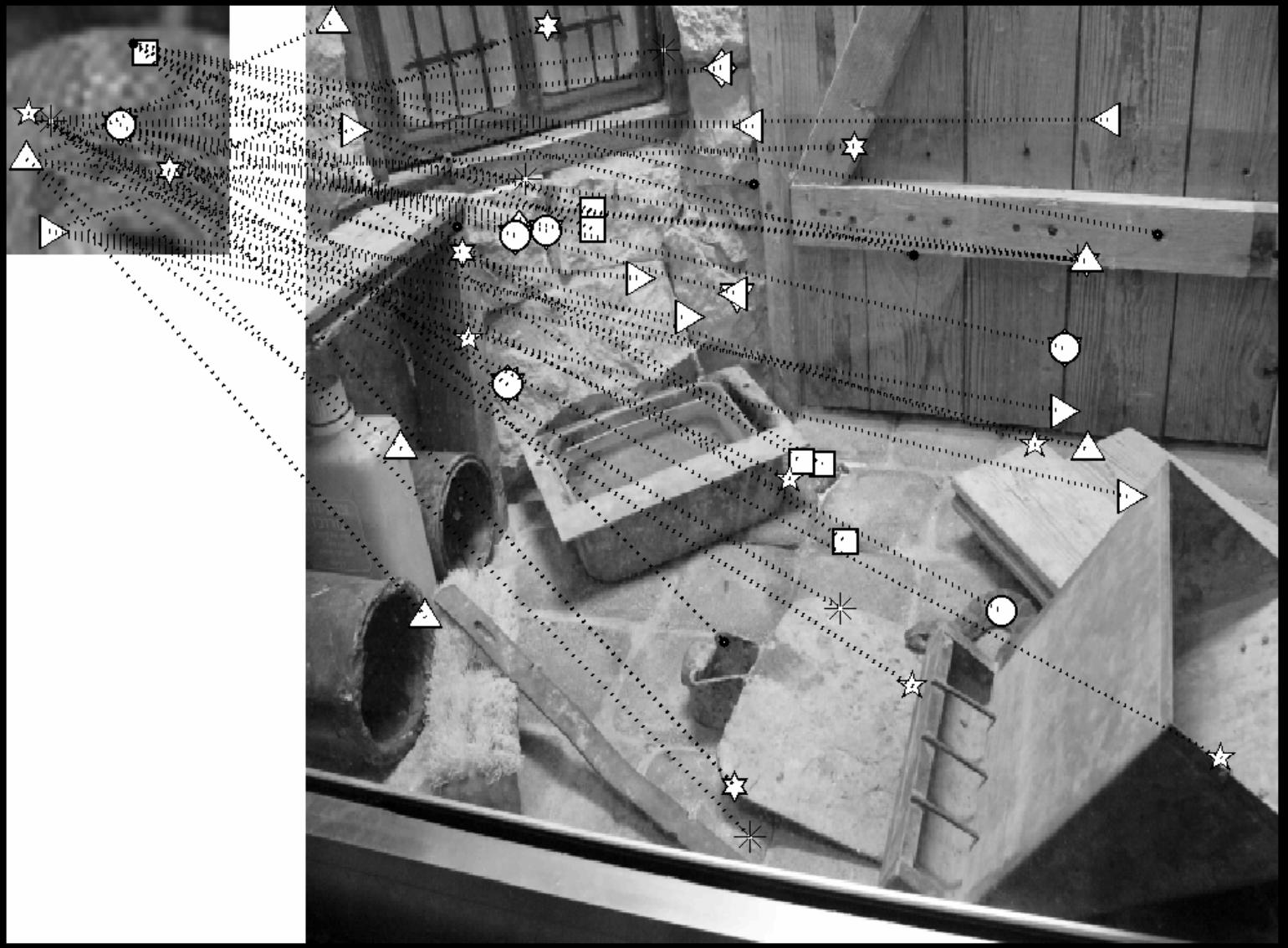
Performance

- Image Size: 640x480 pixels
- Pattern Size: 714 pixels
- Average Sample Size: 17.23 pixels
- Offline Time: 0.007 sec
- Online Time: **0.064 sec**

Non Rigid Deformations – comparison to SIFT



Non Rigid Deformations – comparison to SIFT



Performance

- Image Size: 640x480 pixels
- Pattern Size: 9409 pairs of pixels
- Average Sample Size: 39.98 pairs of pixels
- Offline Time: 1.219 sec
- Online Time: **0.037 sec**

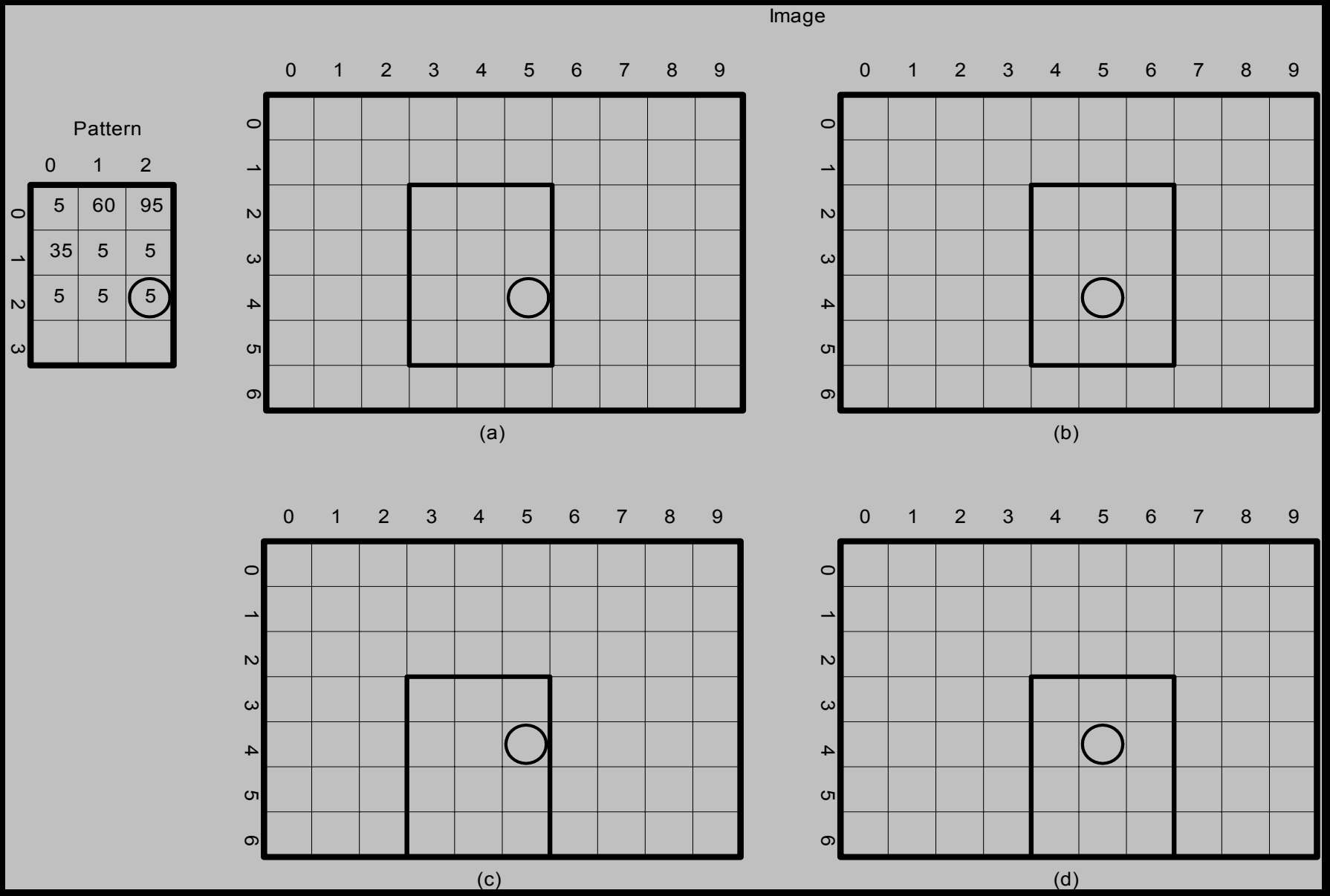
Sliding Window Approach Questions

- Which similarity measure ?
- Can we apply it fast ?
- **Can we use information from one window to the next ?**

Sliding Window Approach Questions

- Which similarity measure ?
- Can we apply it fast ?
- **Can we use information from one window to the next – Using smoothness**

LU Rank of Pixels



Computing LU Rank of Pixels

- Calculate it for each pixel. If we denote by $\bar{R}$ the average LU rank and by $|A|$ the number of pixels in the image, then the average time complexity is $O(|A|\bar{R}^2)$.
- Check for each LU rank which pixels have this value. Find in O(1) the 2d min and max for each value of R (Gil and Werman 93). If we denote by $R_{\max}$ the maximum R value, then the time complexity is $O(|A|R_{\max})$

Acknowledgments

- Professor Ester Samuel-Cahn
- Refael Vivanti
- Amichai Zisken, Ori Maoz, Amit Gruber, Amnon Aaronsohn, Aviv Hurvitz and Eran Maryuma

Hamming Distance + Sequential Sampling =
Robust Real Time Pattern Matching