Image Pattern Matching with CAs

LEGO Brick Feature Extraction

Lynette van Zijl, Leendert Botha

Computer Science
Stellenbosch University

31 August 2009

Overview

Contribution

- Specific image matching problem – LEGO bricks
- Show that feature extraction phase – cellular automata

Overview

Contribution

- Specific image matching problem – LEGO bricks
- Show that feature extraction phase – cellular automata

Overview

- Semantic pattern matching of LEGO brick images
- Cellular automata
- Five phases:
  - Background removal
  - Edge detection
  - Stud location
  - Stud formation
  - Top surface
- Results and conclusion

Image matching on LEGO bricks

CBIR – content-based image retrieval

- Matching digital images in large databases
- Semantic match based on features such as colour, shape, texture
- Find all dogs in image database of animals

Image matching on LEGO bricks

CBIR – content-based image retrieval

- Matching digital images in large databases
- Semantic match based on features such as colour, shape, texture
- Find all dogs in image database of animals

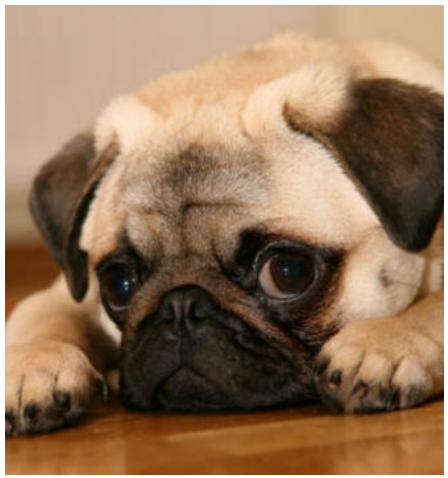


Image matching on LEGO bricks

CBIR – content-based image retrieval

- Matching digital images in large databases
- Semantic match based on features such as colour, shape, texture
- Find all dogs in image database of animals

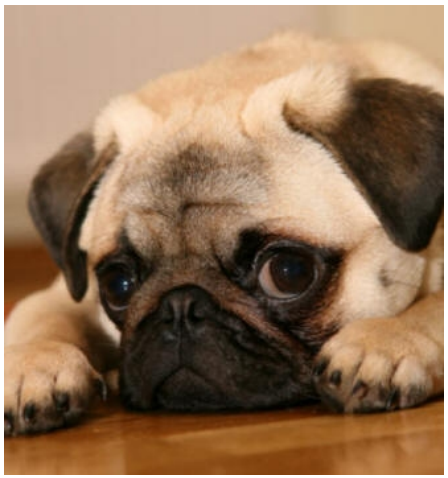


Image matching on LEGO bricks

CBIR – content-based image retrieval

- Matching digital images in large databases
- Semantic match based on features such as colour, shape, texture
- Find all dogs in image database of animals

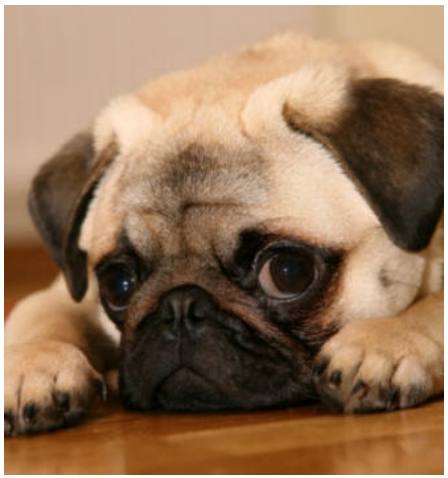


Image matching on LEGO bricks

Image matching on LEGO bricks

- Given image p, find semantically-equivalent images from P
- Restrict to "standard" bricks
- Semantic match defined on stud formation and brick form as features

Image matching on LEGO bricks

Image matching on LEGO bricks

- Given image p, find semantically-equivalent images from P
- Restrict to "standard" bricks
- Semantic match defined on stud formation and brick form as features

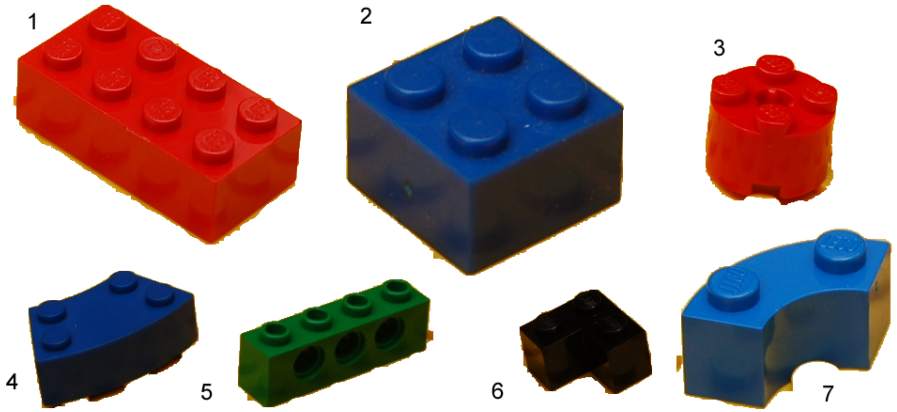
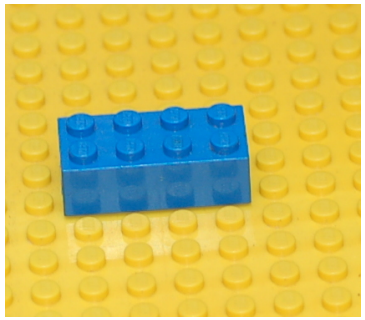


Image matching on LEGO bricks

Image matching on LEGO bricks

- Given image p, find semantically-equivalent images from P
- Restrict to "standard" bricks
- Semantic match defined on stud formation and brick form as features



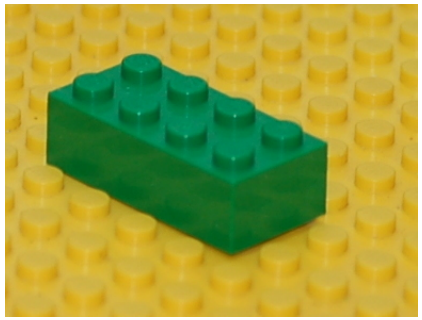


Image matching on LEGO bricks

Image matching on LEGO bricks

- Given image p, find semantically-equivalent images from P
- Restrict to "standard" bricks
- Semantic match defined on stud formation and brick form as features

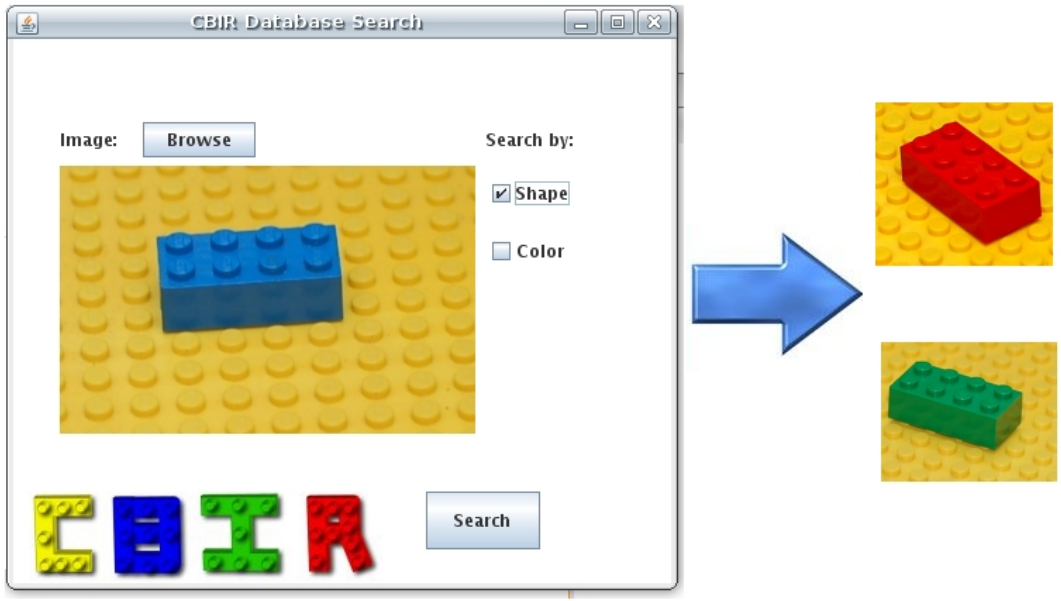


Image matching on LEGO bricks

Steps to do image matching on LEGO bricks

- Remove background
- Find edges of brick
- Find studs
- 3D to 2D: identify top of brick
- Identify stud arrangement
- Construct final feature vector

Image matching on LEGO bricks

Steps to do image matching on LEGO bricks

- Remove background
- Find edges of brick
- Find studs
- 3D to 2D: identify top of brick
- Identify stud arrangement
- Construct final feature vector

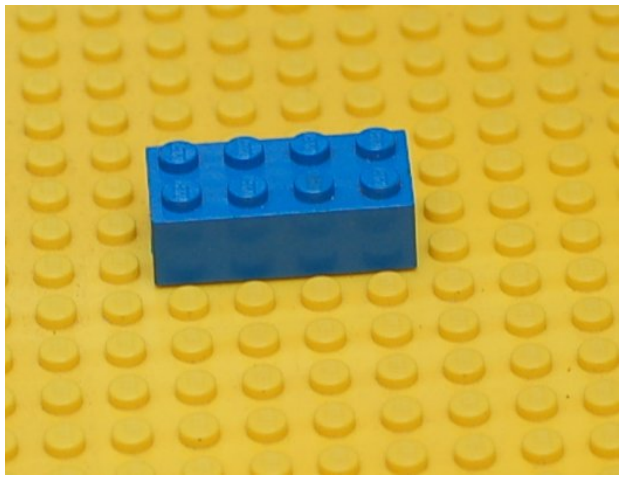


Image matching on LEGO bricks

Steps to do image matching on LEGO bricks

- Remove background
- Find edges of brick
- Find studs
- 3D to 2D: identify top of brick
- Identify stud arrangement
- Construct final feature vector

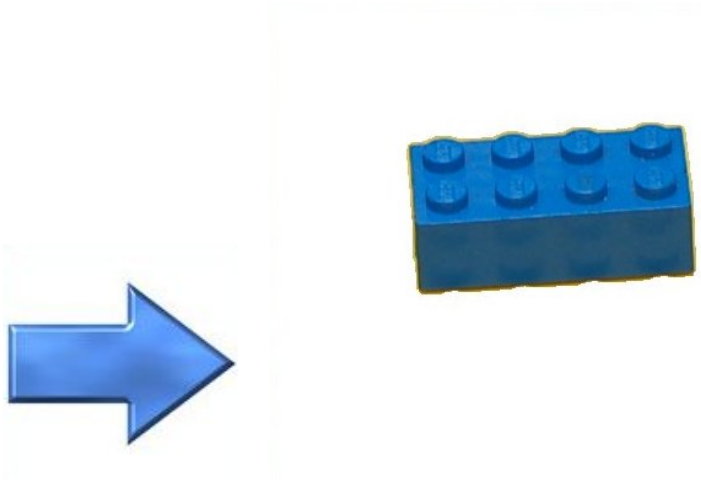


Image matching on LEGO bricks

Steps to do image matching on LEGO bricks

- Remove background
- Find edges of brick
- Find studs
- 3D to 2D: identify top of brick
- Identify stud arrangement
- Construct final feature vector

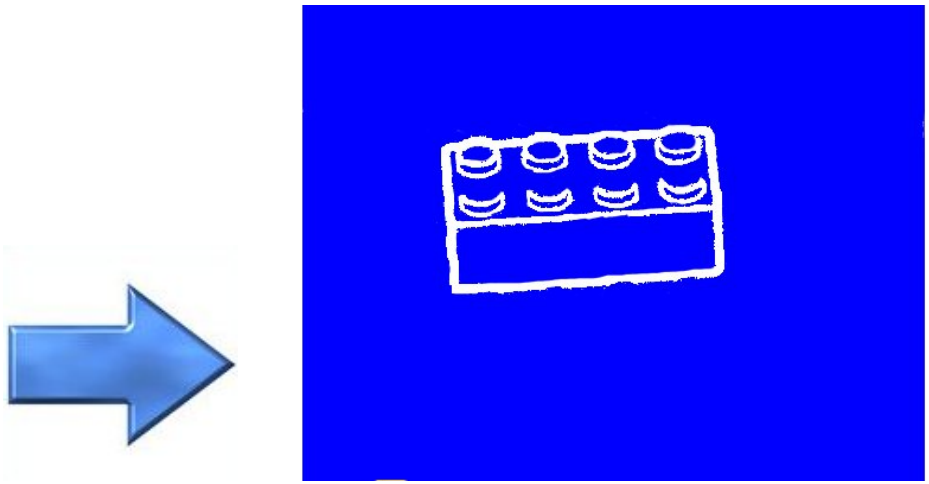


Image matching on LEGO bricks

Steps to do image matching on LEGO bricks

- Remove background
- Find edges of brick
- Find studs
- 3D to 2D: identify top of brick
- Identify stud arrangement
- Construct final feature vector

Image matching on LEGO bricks

Steps to do image matching on LEGO bricks

- Remove background
- Find edges of brick
- Find studs
- 3D to 2D: identify top of brick
- Identify stud arrangement
- Construct final feature vector

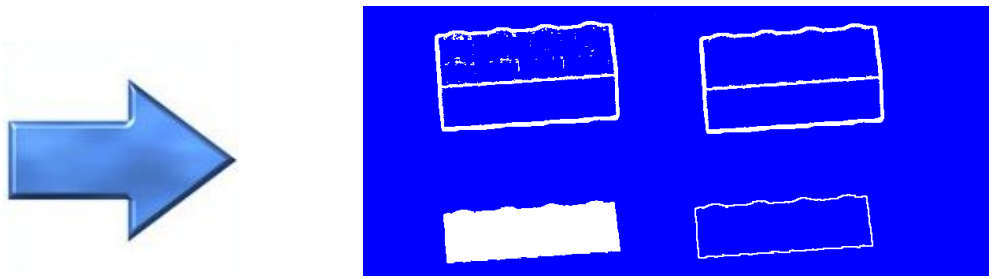
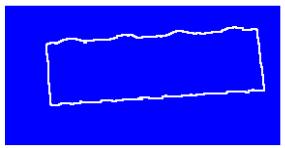


Image matching on LEGO bricks

Steps to do image matching on LEGO bricks

- Remove background
- Find edges of brick
- Find studs
- 3D to 2D: identify top of brick
- Identify stud arrangement
- Construct final feature vector



→ [-6.1 37.3 6.1 7.0 40.9 41.4 56.5]

Cellular automata

A 2D CA is a 3-tuple $M = (A, N, f)$, such that

- A is the finite nonempty state set,
- $N = (\vec{x}_1, \ldots, \vec{x}_n)$ is the neighbourhood vector consisting of vectors in $\mathbb{Z}^2$, and
- $f : A^n \longrightarrow A$ is the transition rule.

Cellular automata

A 2D CA is a 3-tuple $M = (A, N, f)$, such that

- A is the finite nonempty state set,
- $N = (\vec{x}_1, \ldots, \vec{x}_n)$ is the neighbourhood vector consisting of vectors in $\mathbb{Z}^2$, and
- $f : A^n \longrightarrow A$ is the transition rule.

Configuration

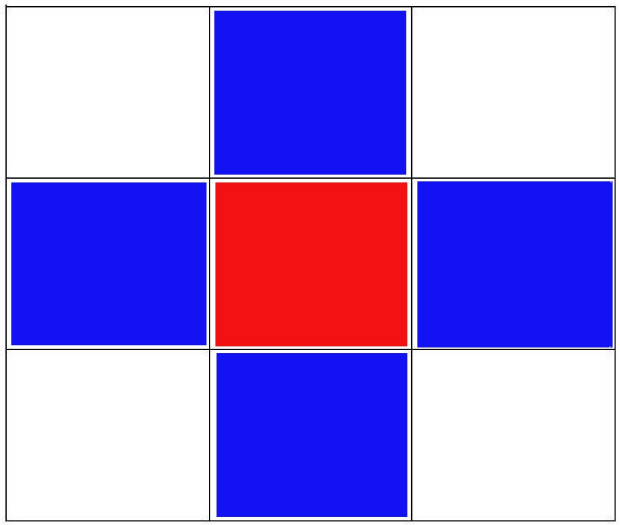
Given configuration c of cells at time t, configuration c' at time $t+1$ for each cell $\vec{x}$ calculated simultaneously as

$$c'(\vec{x}) = f(c(\vec{x}_1, \ldots, \vec{x}_n)) \quad .$$

Cellular automata

Neighbourhood

Example – Von Neumann neighbourhood for cell $x_{i,j}$ defined as $< x_{i-1,j}, x_{i,j-1}, x_{i,j+1}, x_{i+1,j} >$.



Background elimination

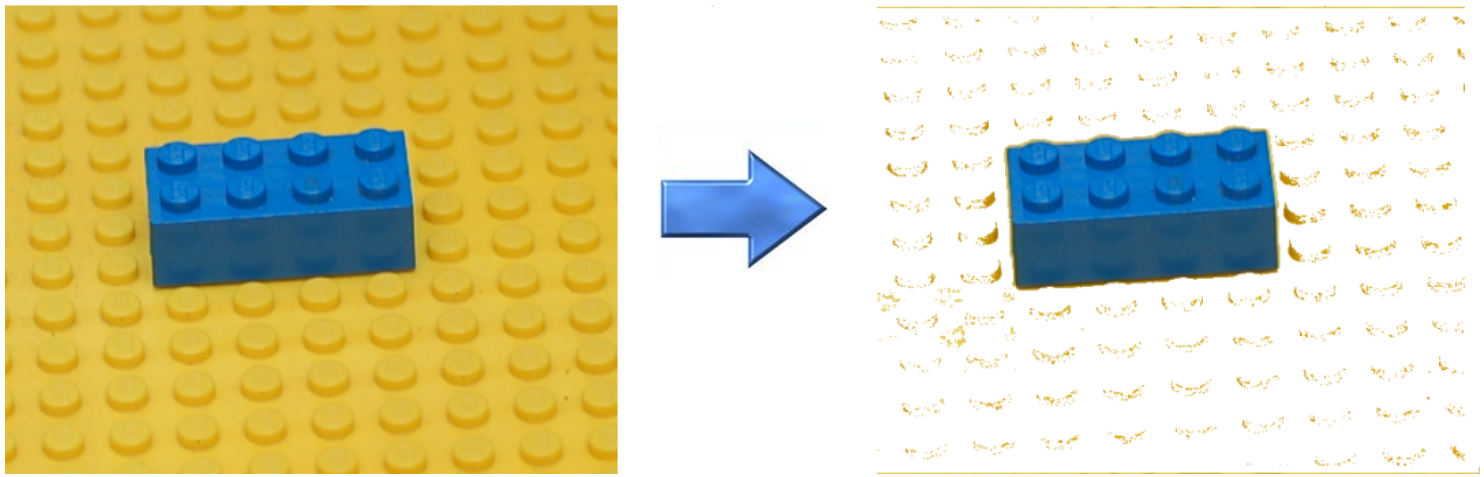
Steps for background elimination

- Baseboard colour subtraction

Background elimination

Steps for background elimination

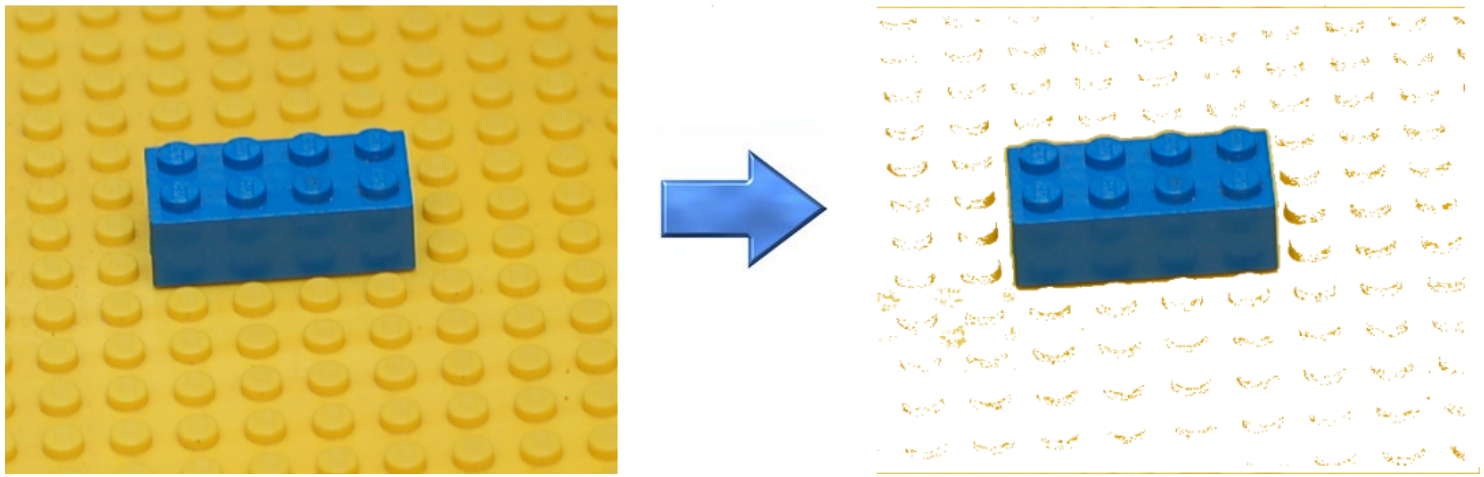
- Baseboard colour subtraction



Background elimination

Steps for background elimination

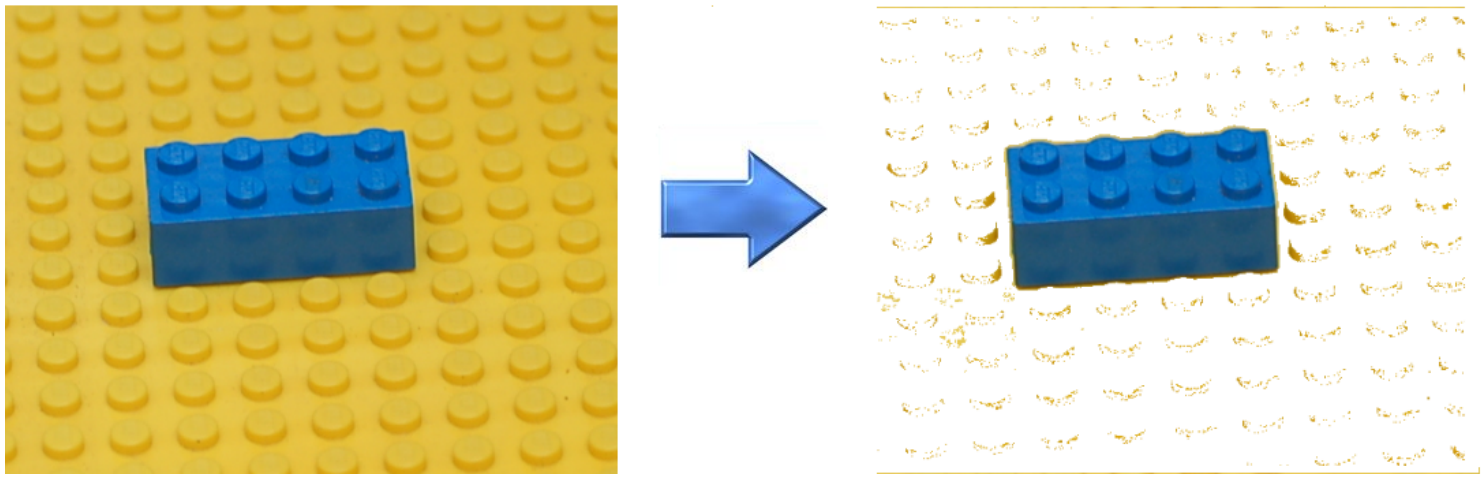
- Baseboard colour subtraction
- CA to remove shadow noise



Background elimination

Steps for background elimination

- Baseboard colour subtraction
- CA to remove shadow noise
- CA to remove straight lines



Background elimination – remove shadow noise

Background elimination – remove shadow noise

CA 1:

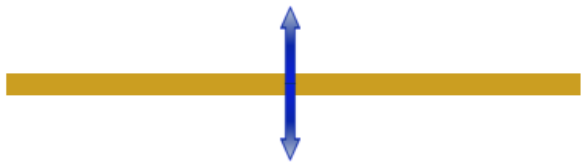
Background elimination – remove shadow noise

CA 1:

Background elimination – remove shadow noise

CA 1:

CA 2:



Background elimination – remove shadow noise

Von Neumann-type neighbourhood $\vec{n}_x$ for each cell $x_{i,j}$, such that $\vec{n}_x = (\vec{x}_N, \vec{x}_S, \vec{x}_W, \vec{x}_E)$, where

$$\begin{aligned} \vec{x}_N &= < x_{i-\delta_L,j}, \ldots, x_{i-1,j} > \\ \vec{x}_S &= < x_{i+1,j}, \ldots, x_{i+\delta_L,j} > \\ \vec{x}_W &= < x_{i,j-\delta_L}, \ldots, x_{i,j-1} > \\ \vec{x}_E &= < x_{i,j+1}, \ldots, x_{i,j+\delta_L} > \quad . \end{aligned}$$

Neighbourhood taken in four directions up to distance δ_L from current cell.

Background elimination – remove shadow noise

Define CA C_L with transition function

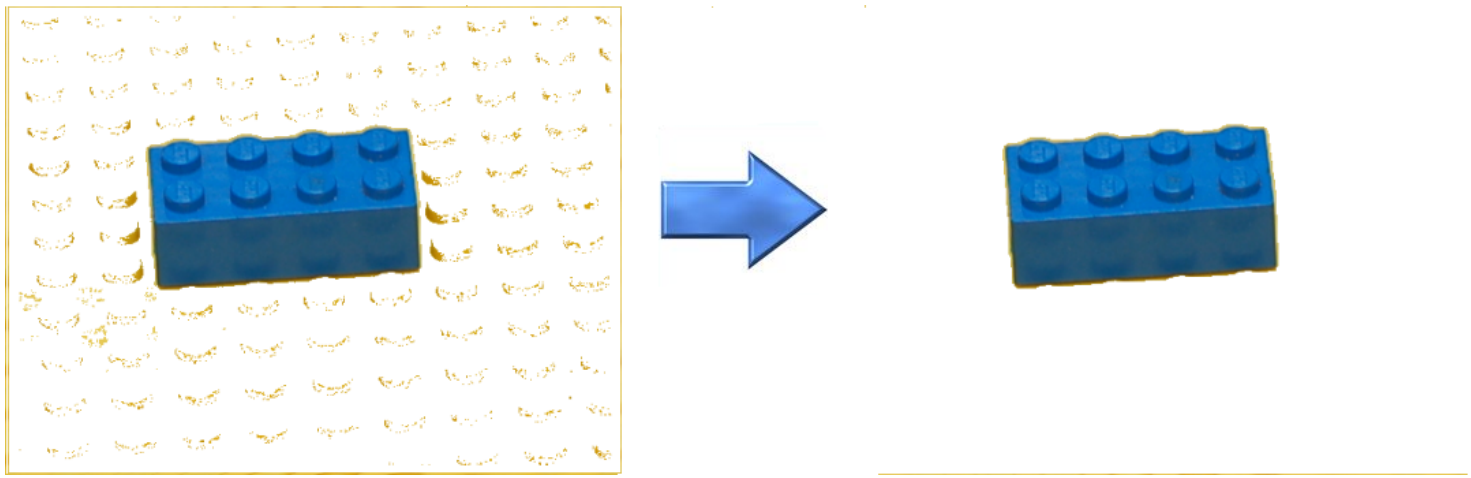
$$c'(x_{i,j}) = \begin{cases} 0, \text{ if } \exists k_N, k_S, k_W, k_E : & x_{i-k_N,j} = 0 \ \& \\ & x_{i+k_S,j} = 0 \ \& \\ & x_{i,j-k_W} = 0 \ \& \\ & x_{i,j+k_E} = 0 \\ 1, \text{ otherwise,} \end{cases}$$

where $1 \leq k_N, k_S, k_W, k_E \leq \delta_L$.

Background elimination – remove straight lines

Define CA C_S with transition function

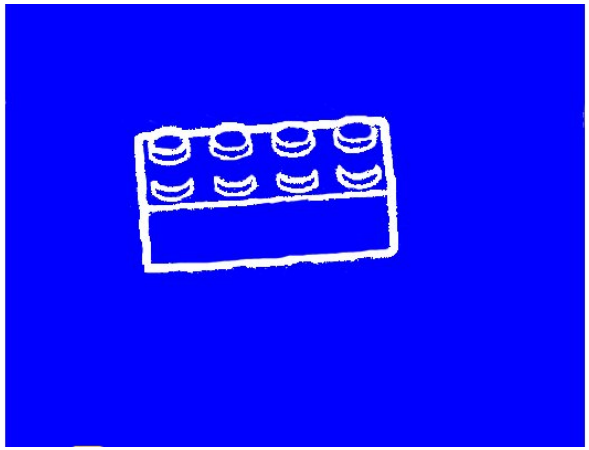
$$c'(x_{i,j}) = \begin{cases} 0, & \text{if } \exists k_N^S, k_S^S : x_{i-k_N^S,j} = 0 \ \& \ x_{i+k_S^S,j} = 0 \\ 0, & \text{if } \exists k_W^S, k_E^S : x_{i,j-k_W^S} = 0 \ \& \ x_{i,j+k_E^S} = 0 \\ 1, & \text{otherwise .} \end{cases}$$



Edge detection

Let $\varphi(a, b)$ be similarity function between pixels. Define CA C_e with transition function

$$c'(x_{i,j}) = \begin{cases} 0, \text{ if} & \varphi(x_{i,j}, x_{i,j-1}) < \epsilon \ \& \ \varphi(x_{i,j}, x_{i,j+1}) < \epsilon \ \& \\ & \varphi(x_{i,j}, x_{i-1,j}) < \epsilon \ \& \ \varphi(x_{i,j}, x_{i+1,j}) < \epsilon \\ x_{i,j}, \text{ otherwise .} & \end{cases}$$



Stud location

Stud matching function –

$$\mathrm{SE}(x,y) = \sum_{\alpha=1}^{N}\sum_{\beta=1}^{N}(f(x-\alpha, y-\beta) - T(\alpha,\beta))^2$$

where f is the image and T is the $N \times N$ template.

Stud formation

Stud formation

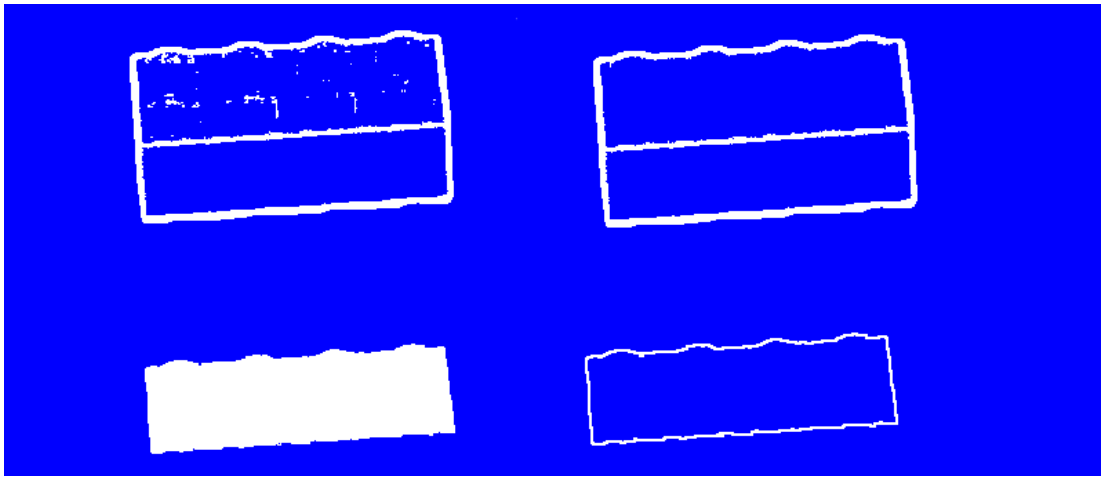
Find minimal set of straight lines $\mathcal{L} = \{l_1, l_2, \ldots, l_n\}$ where each stud on exactly one l_i.

- Line between each pair of two studs
- Repeatedly, find line covering most studs. Remove those studs and line, until no more studs left

Top surface (form)

Define C_f with Moore neighbourhood and cells *top surface=0*, *edge=1*, *background=2* and transition function

$$c'(x_{i,j}) = \begin{cases} 0, \text{ if} & c(x_{i,j}) \neq 1 \ \& \ c(x_{i,j}) \neq 2 \ \& \\ & (\Sigma_{m,n} x_{m,n} = 0) > th, \\ 1, \text{ otherwise .} & \end{cases}$$

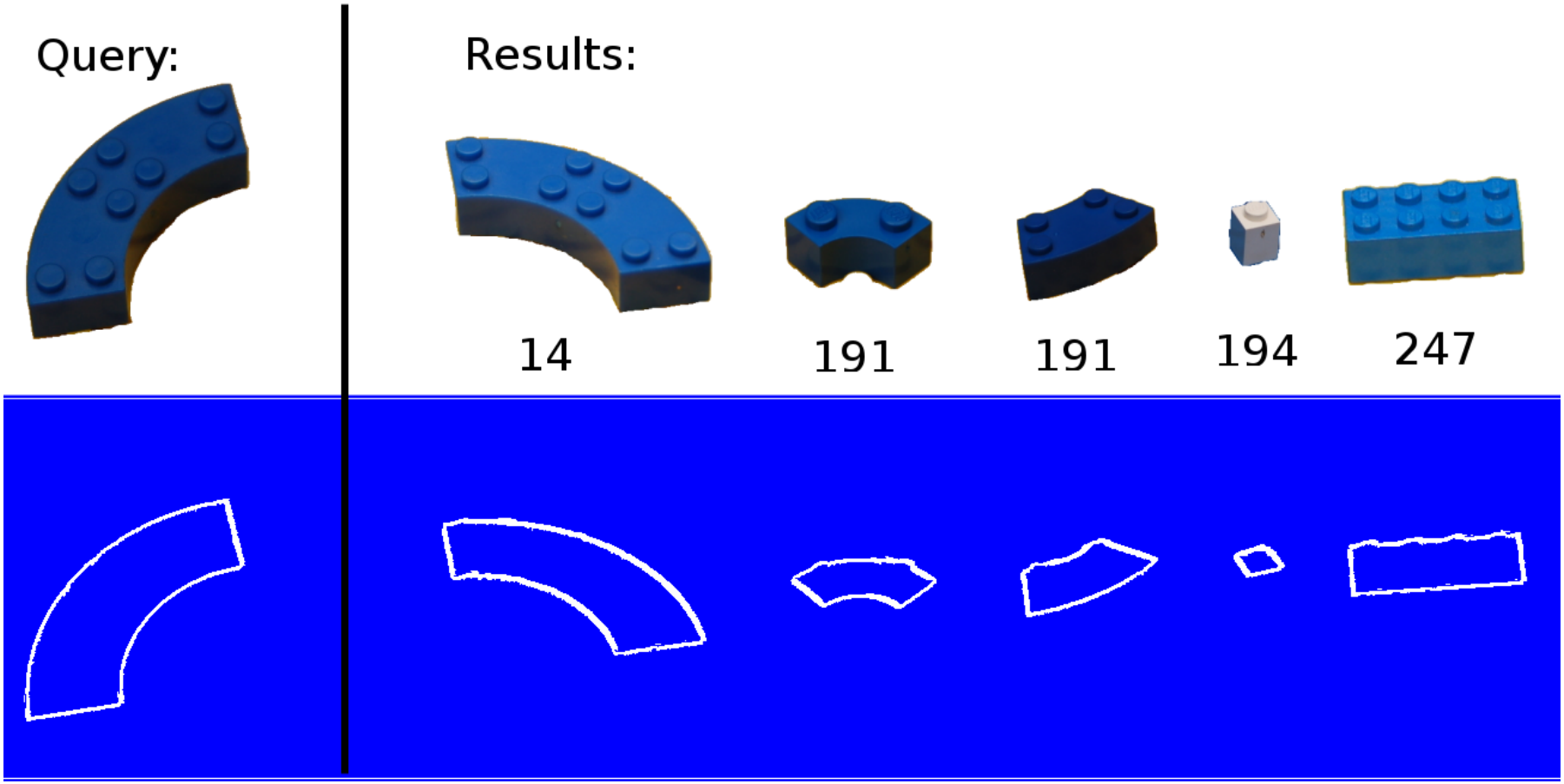


Feature vector

- Colour histogram
- Construct numerical feature vector – Hu set of invariant moments

Results

Example:



Failure on incorrect edge detection.

Conclusion

Conclusion

- Successful image pattern matching feature extraction with CA
- Fast (realtime)
- Dependant on edge detection

Future work

- Extend to other LEGO forms
- Investigate other specialized CBIR applications