Robust Real-Time Detection and Pose Estimation of general (Texture-less) 3D Objects

Stefan Hinterstoißer

Why is Robustness and Real-Time so Important?

How long do you want to wait for your beer?

Some Other Objects to Handle:

well textured objects

almost texture-less objects

Grabner (CVPR07)

SURF (ECCV08)

SIFT (IJCV04)

ORB (ICCV11)

FERNS (CVPR07)

HIPs (BMVC09)

GEPARD (CVPR10)

LEOPAR (CVPR08)

DTT (CVPR09): close[illegible]ntour, binary edge image

Amit (PA[illegible]: too slow

Steger [illegible]02): robu[illegible]too slow

HOG (C[illegible]): too slow

Shape[illegible] (IJCV09): too sl[illegible]

[Orient[illegible]Ha[illegible]dorff Distance[illegible]97,PAMI93): not ro[illegible]ainst clutter, slo[illegible]ary [illegible]ge image

[Orien[illegible]mfer Match[illegible]CV99,TIP97): not [illegible]gainst clutter, bi[illegible]y edg[illegible]mage

What are the Problems with Texture-Less Objects?

- Lack of Texture prevents using **feature points** for efficient reduction of the search space
  - How to guarantee robustness and speed?

- Lack of Texture prevents using **texture-based similarity measures** like NCC, SSD, SAD etc.
  - What kind of robust features to use instead (e.g. illumination invariant etc.)?
  - How to evaluate this similarity measure efficiently?
  - How to evaluate this similarity measure robustly?
  - How to deal with heavy background clutter?

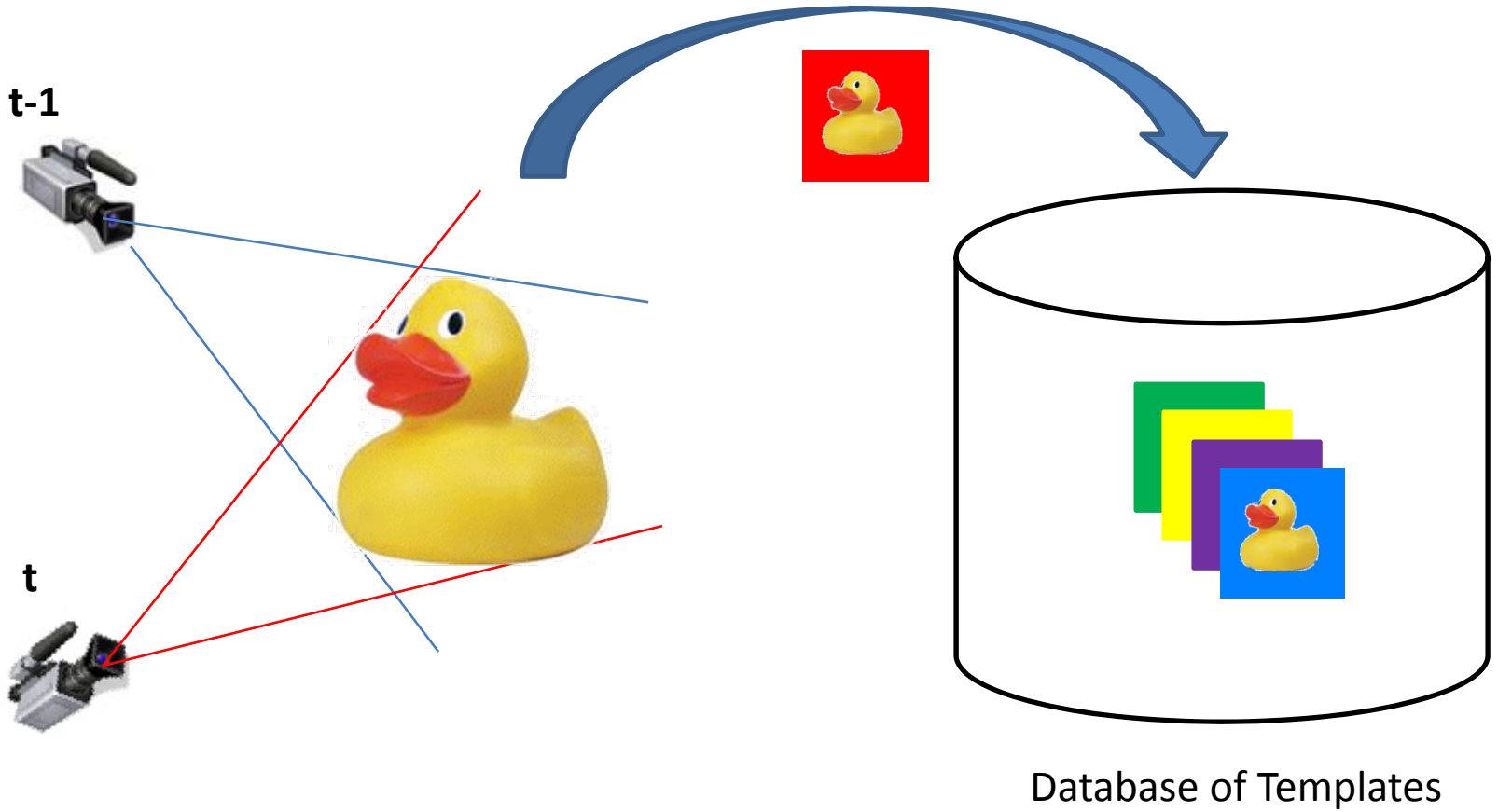
- Robots have to **quickly react** on fast changing environments
  - How to make the approach suitable for online learning?

LINE – A Very, Very Fast Template Matching Framework for Multiple Cues

Stefan Hinterstoißer, Stefan Holzer, Cedric Cagniart,

Vincent Lepetit, Peter Sturm, Slobodan Ilic, Pascal Fua, Nassir Navab

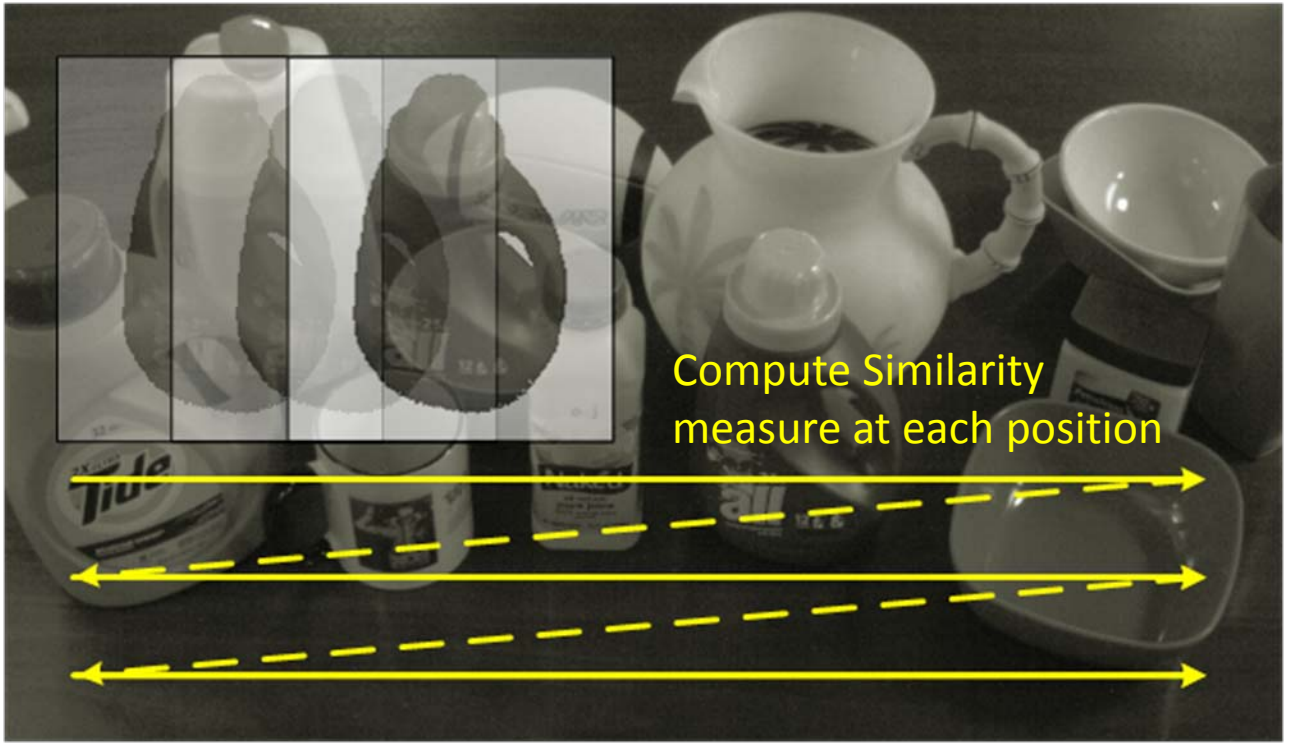
Online Learning: Using Template Matching



For Robust Matching: Use Sliding Window

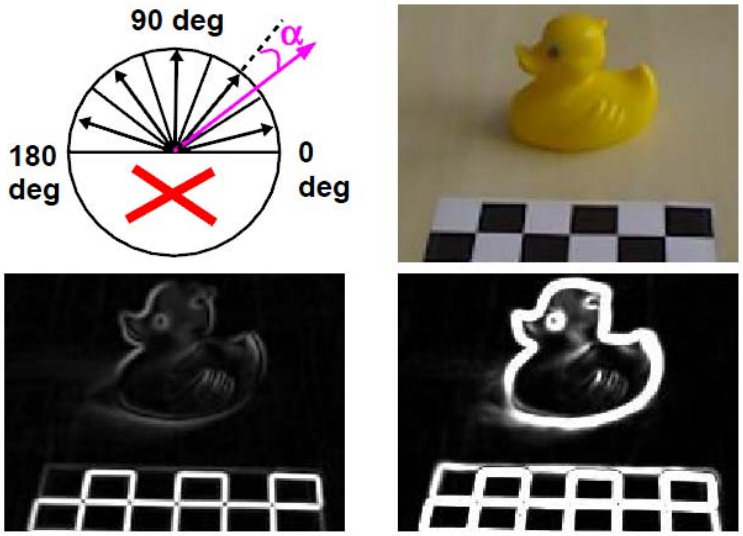
Template

Current Scene



What Features to Use?

2D Color gradients



Computation:

$$\mathcal{I}_{\mathcal{G}}(x) = \frac{\partial \hat{\mathcal{C}}}{\partial x}$$

$$\hat{\mathcal{C}}(x) = \underset{\mathcal{C}\in\{R,G,B\}}{\arg\max} \left\| \frac{\partial \mathcal{C}}{\partial x} \right\|$$

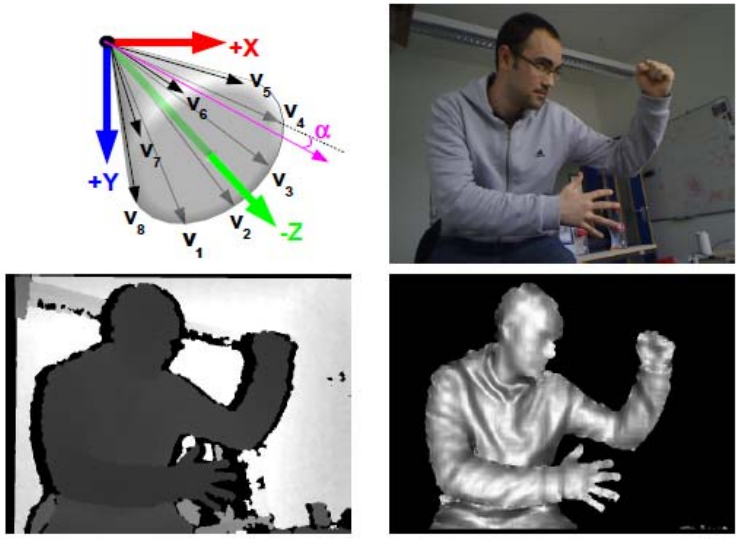
Similarity Measure:

$$f_{\mathcal{G}}(\mathcal{O}_{\mathcal{G}}(r), \mathcal{I}_{\mathcal{G}}(t)) = \left| \mathcal{O}_{\mathcal{G}}(r)^{\top} \mathcal{I}_{\mathcal{G}}(t) \right|$$

- fast to compute
- robust to illumination changes
- mainly lie on the contour of the object

What Features to Use?

3D Surface Normals



Computation:

$$\mathcal{D}(x+dx) - \mathcal{D}(x) = dx^\top \nabla \mathcal{D} + h.o.t.$$

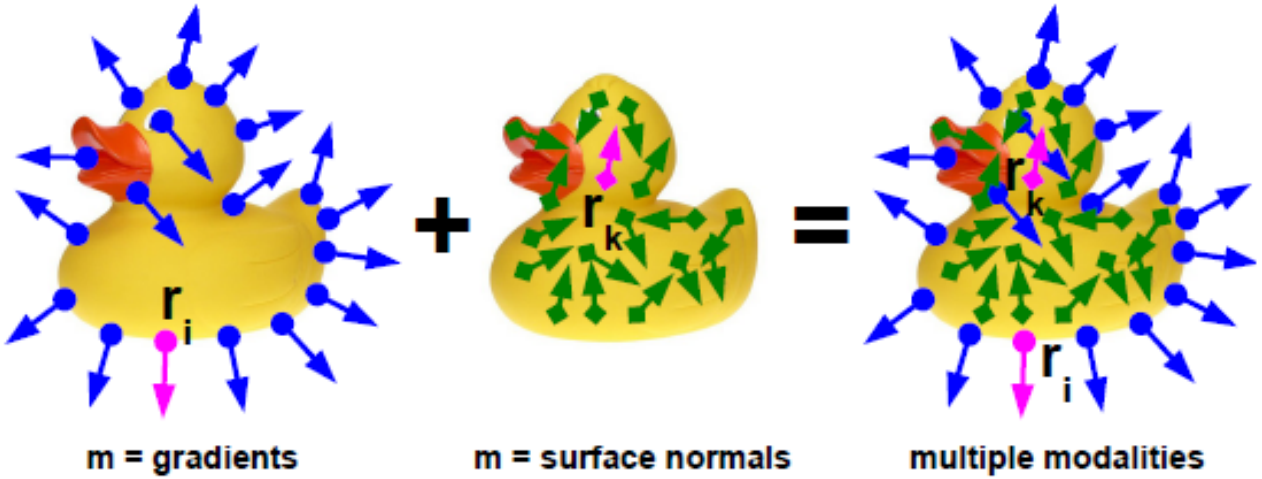
$$X = \vec{v}(x)\mathcal{D}(x),$$
$$X_1 = \vec{v}(x+[1,0]^\top)(\mathcal{D}(x) + [1,0]^\top \hat{\nabla}\mathcal{D}),$$
$$X_2 = \vec{v}(x+[0,1]^\top)(\mathcal{D}(x) + [0,1]^\top \hat{\nabla}\mathcal{D}).$$

Similarity Measure:

$$f_{\mathcal{D}}(\mathcal{O}_{\mathcal{D}}(r), \mathcal{I}_{\mathcal{D}}(t)) = \mathcal{O}_{\mathcal{D}}(r)^\top \mathcal{I}_{\mathcal{D}}(t)$$

- fast to compute
- robust on larger areas
- mainly lie on the interior of the object

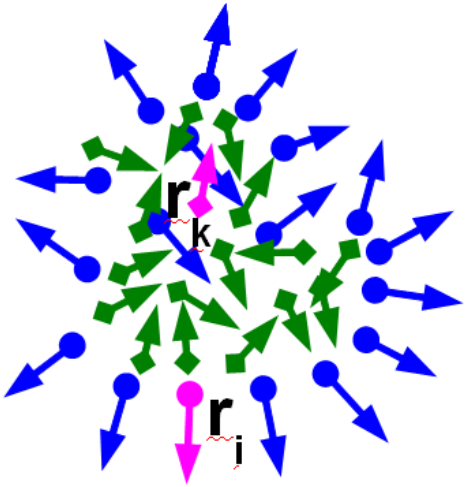
2D Color Gradients and 3D Surface Normals are Complementary!



Template $\mathcal{T} = (\{\mathcal{O}_m\}_{m \in \mathcal{M}}, \mathcal{P})$

$\mathcal{P}$ is a list of pairs (r, m)

reference images $\{\mathcal{O}_m\}_{m \in \mathcal{M}}$

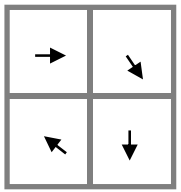


Similarity Measure: Invariant to Small Deformations

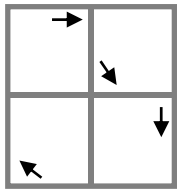
State-of-the-art template matching:

$$\mathcal{E}(\{\mathcal{I}_m\}_{m\in\mathcal{M}}, \mathcal{T}, c) = \sum_{(r,m)\in\mathcal{P}} \Big(f_m(\mathcal{O}_m(r), \mathcal{I}_m(c+r)) \Big)$$

Our approach (invariant to small deformations):



same score



$$\mathcal{E}(\{\mathcal{I}_m\}_{m\in\mathcal{M}}, \mathcal{T}, c) = \sum_{(r,m)\in\mathcal{P}} \Big(\max_{t\in\mathcal{R}(c+r)} f_m(\mathcal{O}_m(r), \mathcal{I}_m(t)) \Big)$$

$$f_{\mathcal{G}}(\mathcal{O}_{\mathcal{G}}(r), \mathcal{I}_{\mathcal{G}}(t)) = \left|\mathcal{O}_{\mathcal{G}}(r)^{\top}\mathcal{I}_{\mathcal{G}}(t)\right|$$

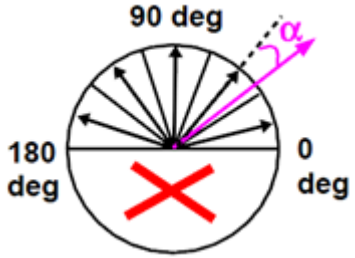
2D Gradient Measure

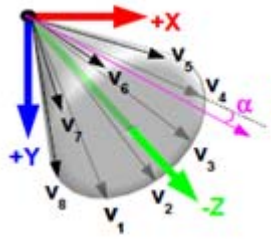
$$f_{\mathcal{D}}(\mathcal{O}_{\mathcal{D}}(r), \mathcal{I}_{\mathcal{D}}(t)) = \mathcal{O}_{\mathcal{D}}(r)^{\top}\mathcal{I}_{\mathcal{D}}(t)$$

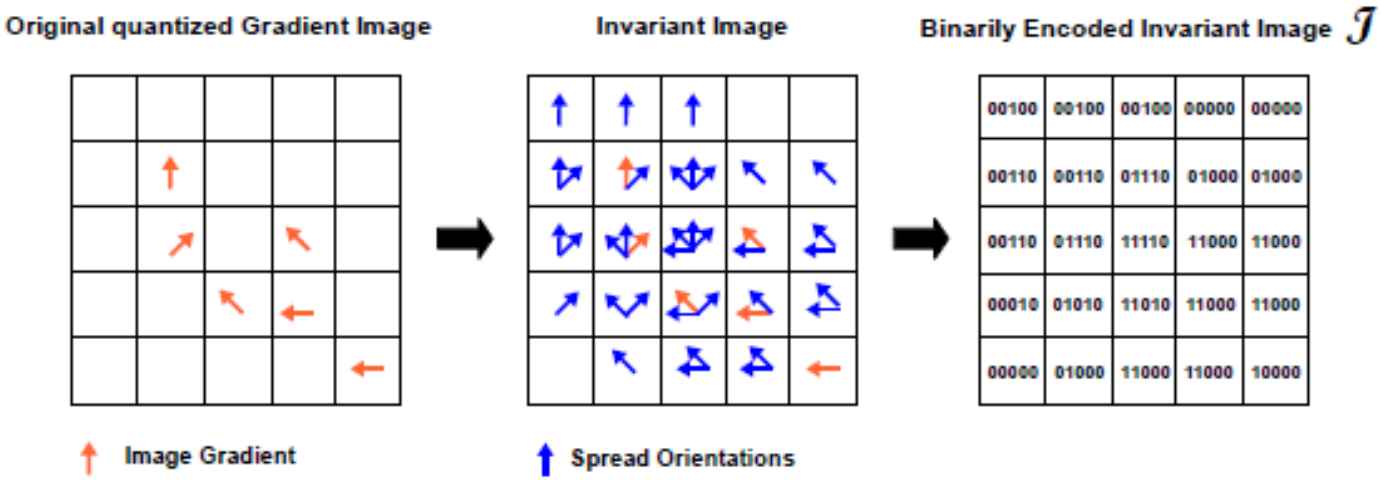
3D Surface Normal Measure

How do we realize it?

We first ***discretize*** the gradient orientation and …
… ***spread*** them around the initial position …





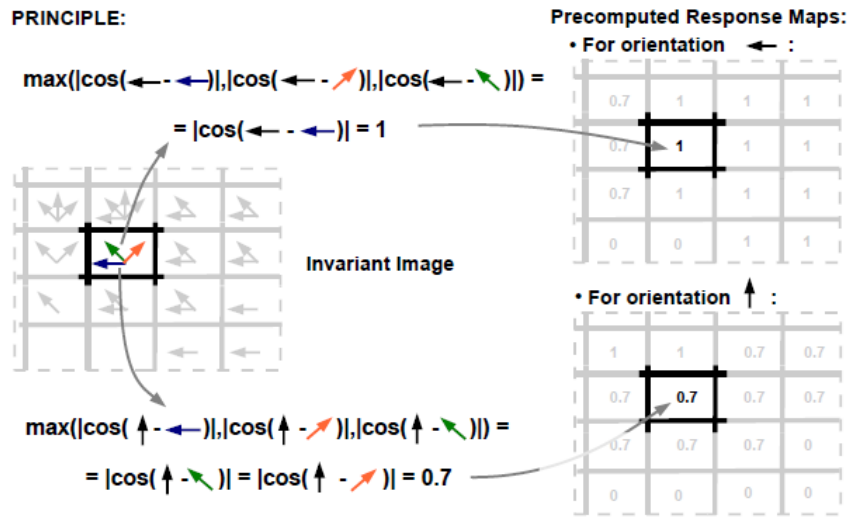


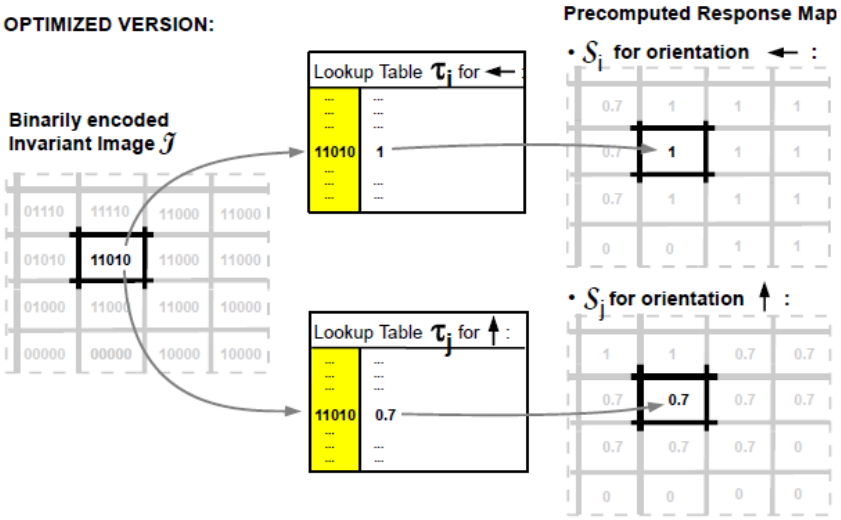
Evaluating the similarity score for each template and each image location separatly would be very inefficient!

How do we realize it?

…due to the discretization we can efficiently precompute **gradient response maps** for each feature **once** using precomputed **look-up tables** …

… instead of computing the similarity response for each template separatly and **over and over again.**



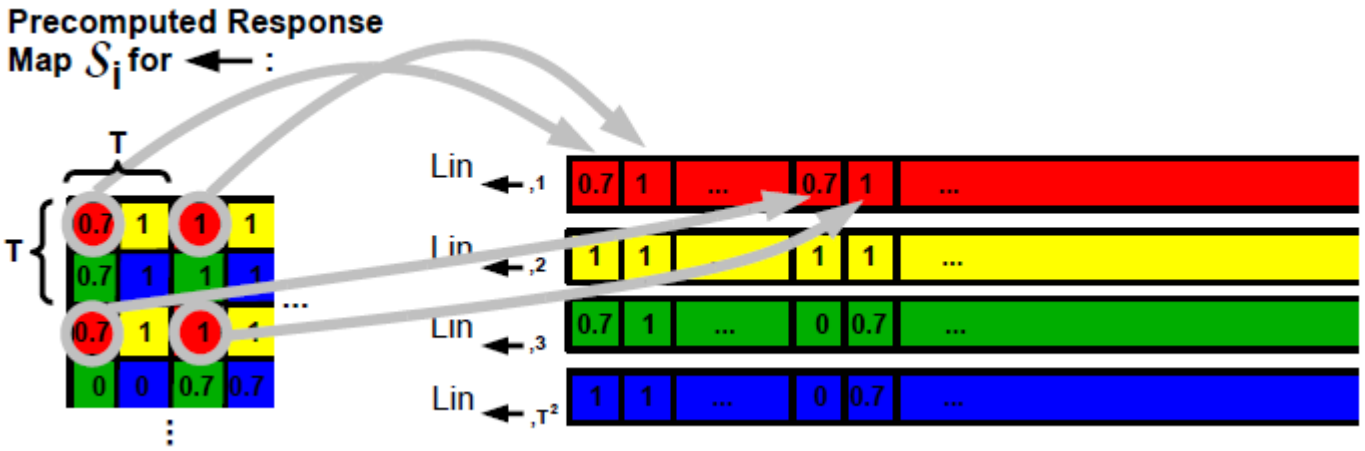


How do we realize it?

Due to the feature ***spreading*** we have ***invariance to small translations***.
Therefore we only have to consider each **i'th pixel**…

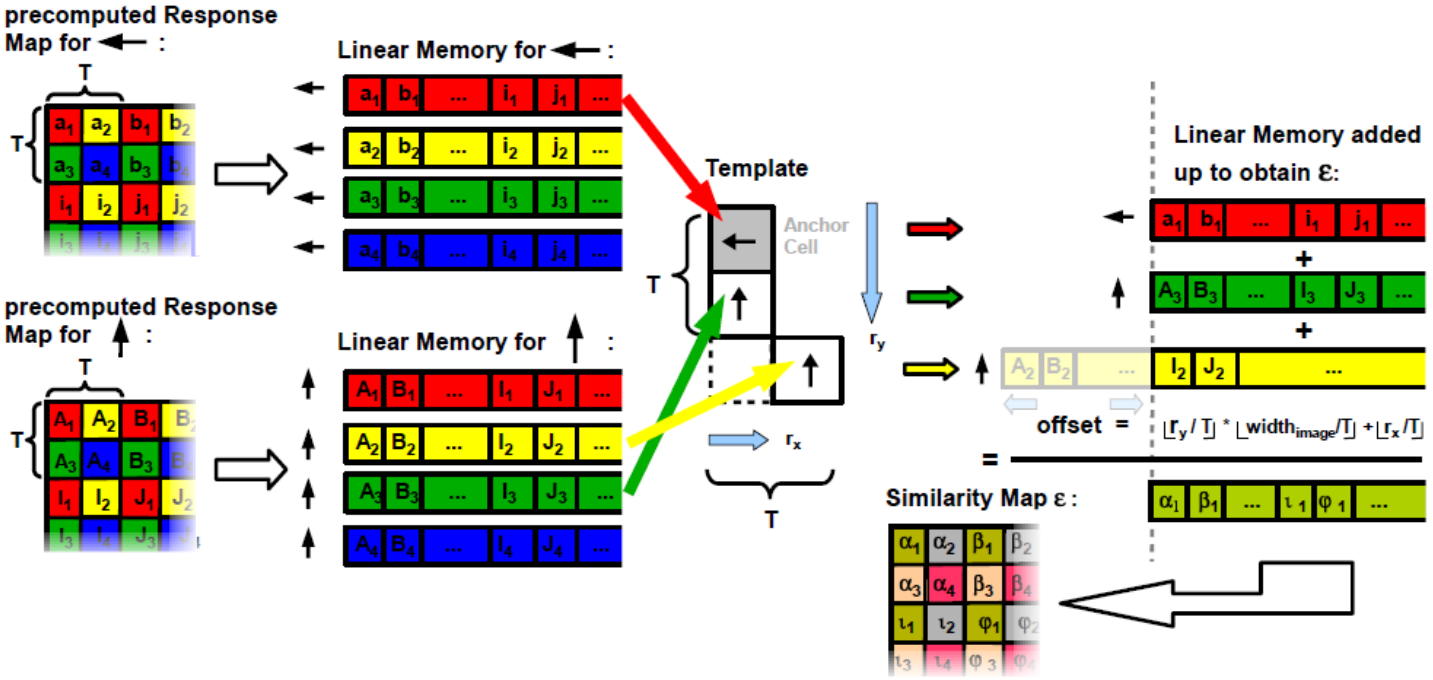
… since modern processors always load a whole cache line into memory and use vectorized operations we would not fully exploit this technology if considering only each i'th pixel …

… therefore we **linearize** the gradient response maps in order to be able to use **SSE instructions** and in order **to avoid cache misses.**

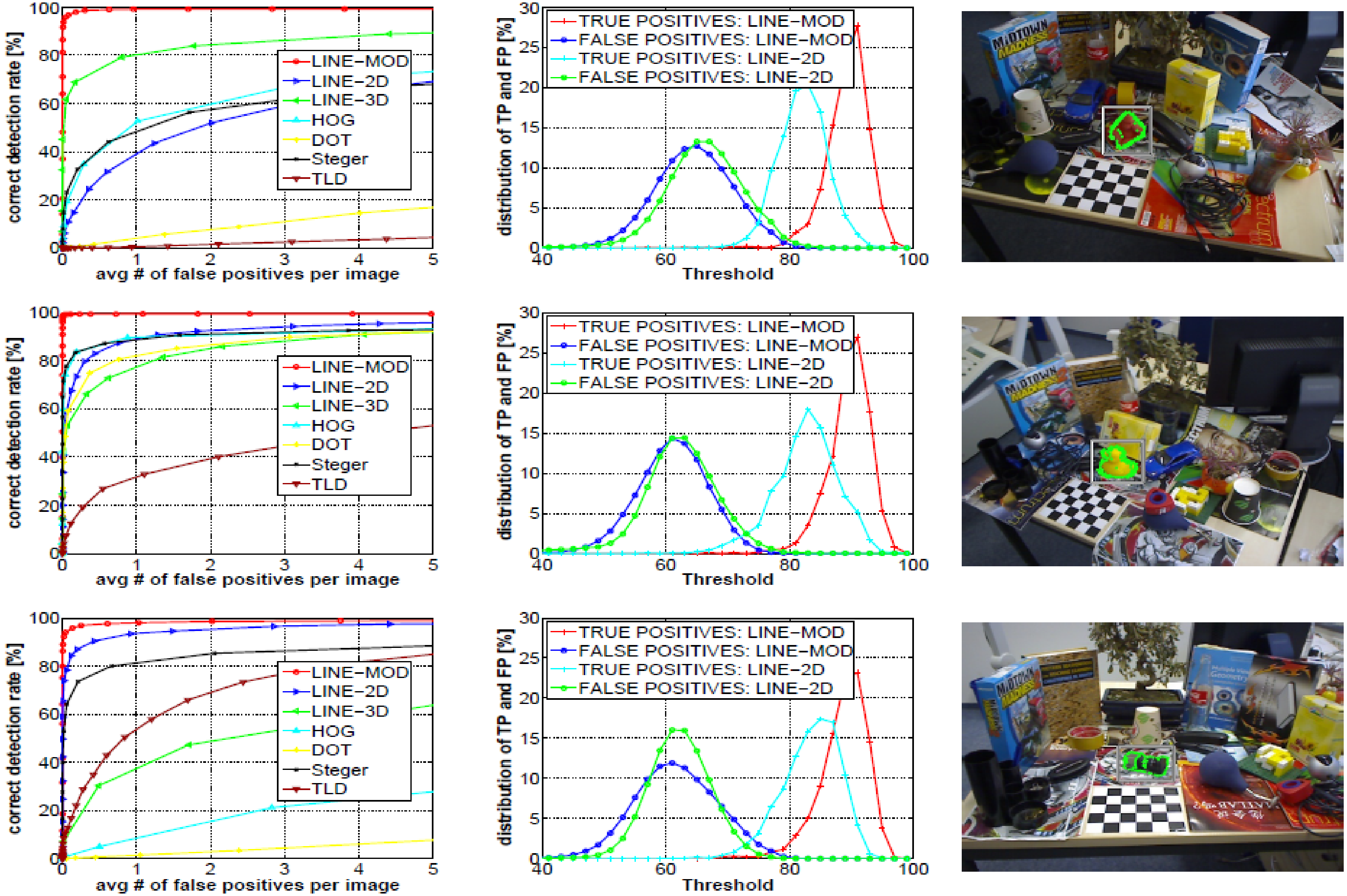


How do we realize it?

Matching then means only to **sum up** as many linear memories as we have different features (with a certain **offset** relative to the original feature location). E.g. for a VGA image we only need 300*G instruction calls per template (using SSE)!

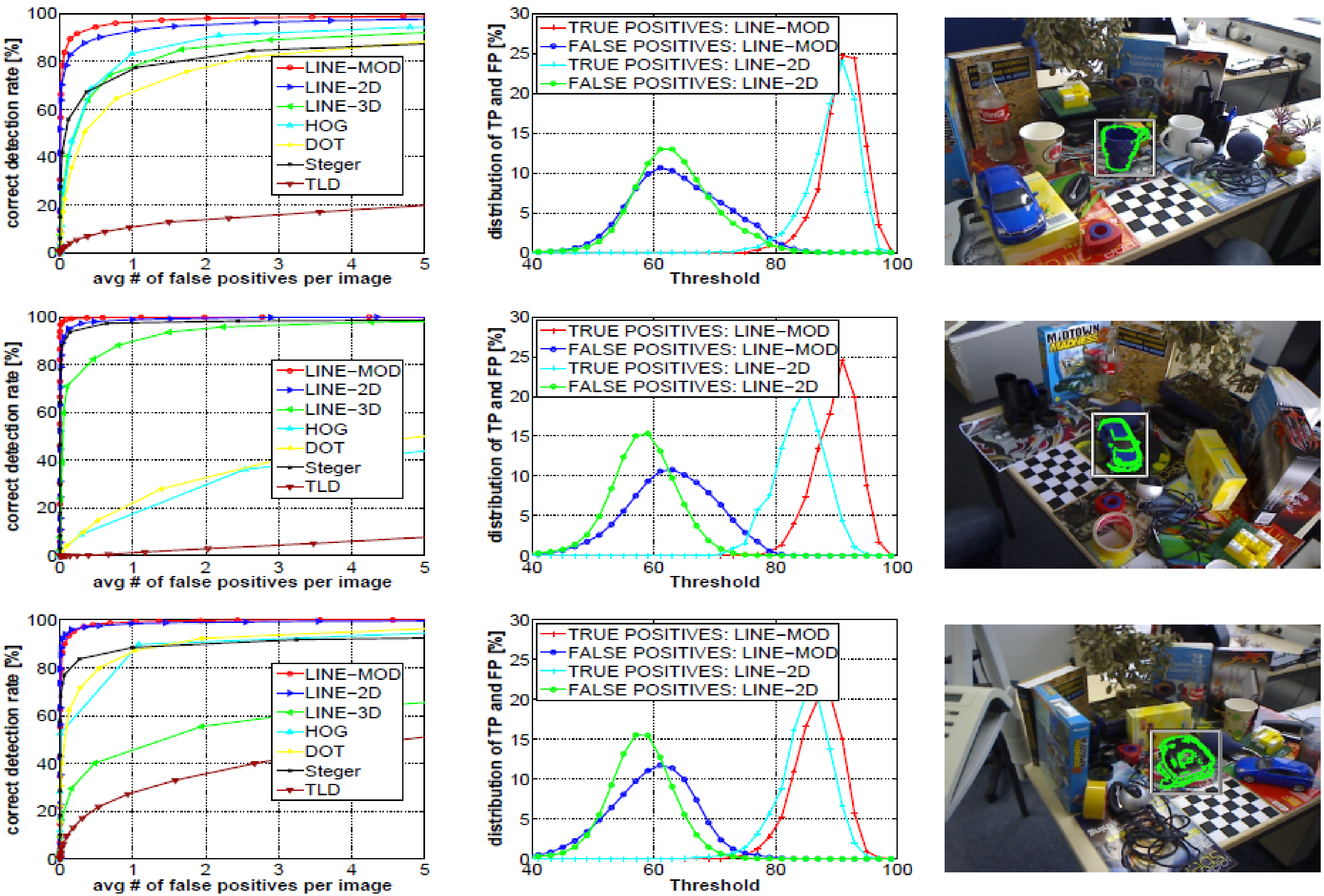


Results:



correct detection rate [%]
avg # of false positives per image
LINE−MOD
LINE−2D
LINE−3D
HOG
DOT
Steger
TLD
distribution of TP and FP [%]
Threshold
TRUE POSITIVES: LINE−MOD
FALSE POSITIVES: LINE−MOD
TRUE POSITIVES: LINE−2D
FALSE POSITIVES: LINE−2D

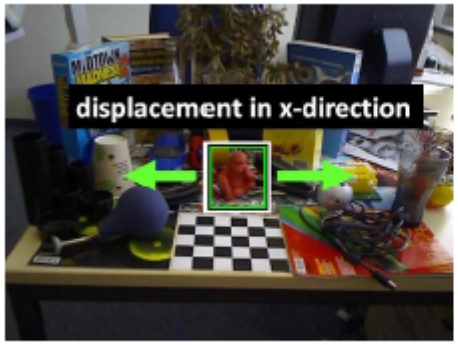
Results:

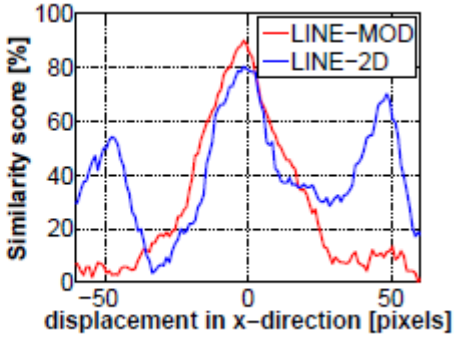


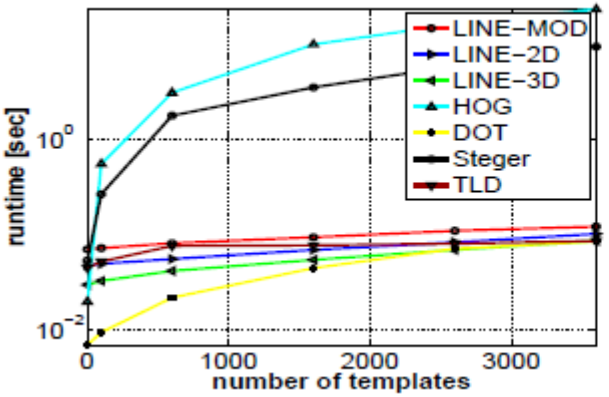
correct detection rate [%]
avg # of false positives per image
LINE−MOD
LINE−2D
LINE−3D
HOG
DOT
Steger
TLD
distribution of TP and FP [%]
Threshold
TRUE POSITIVES: LINE−MOD
FALSE POSITIVES: LINE−MOD
TRUE POSITIVES: LINE−2D
FALSE POSITIVES: LINE−2D
MIDTOWN MADNESS

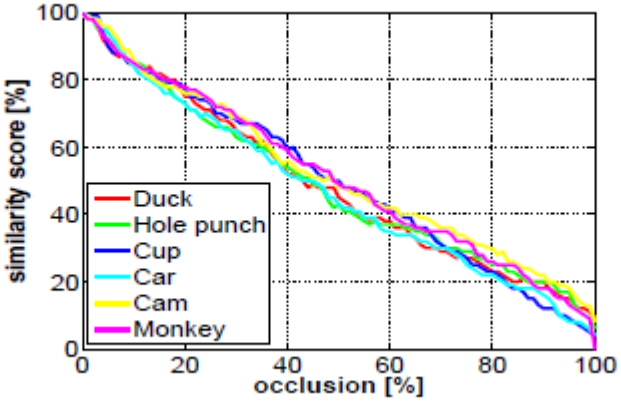
Results:

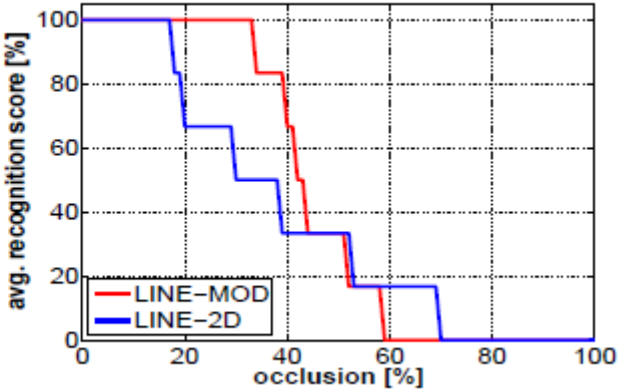
Sequence	LINE-MOD	LINE-3D	LINE-2D	HOG	DOT	Steger	TLD
Toy-Monkey (2164 pics)	97.9%–0.3%	86.1%–13.8%	50.8%–49.1%	51.8%–48.2%	8.6%–91.4%	69.6%–30.3%	0.8%–99.1%
Camera (2173 pics)	97.5%–0.3%	61.9%–38.1%	92.8%–6.7%	18.2%–81.8%	1.9%–98.0%	96.9%–0.4%	53.3%–46.6%
Toy-Car (2162 pics)	97.7%–0.0%	95.6%–2.5%	96.9%–0.4%	44.1%–55.9%	34.0%–66.0%	83.6%–16.3%	0.1%–98.9%
Cup (2193 pics)	96.8%–0.5%	88.3%–10.6%	92.8%–6.0%	81.1%–18.8%	64.1%–35.8%	90.2%–8.9%	10.4%–89.6%
Toy-Duck (2223 pics)	97.9%–0.0%	89.0%–10.0%	91.7%–8.0%	87.6%–12.4%	78.2%–21.8%	92.2%–7.6%	28.0%–71.9%
Hole punch (2184 pics)	97.0%–0.2%	70.0%–30.0%	96.4%–0.9%	92.6%–7.4%	87.7%–12.0%	90.3%–9.7%	26.5%–73.4%



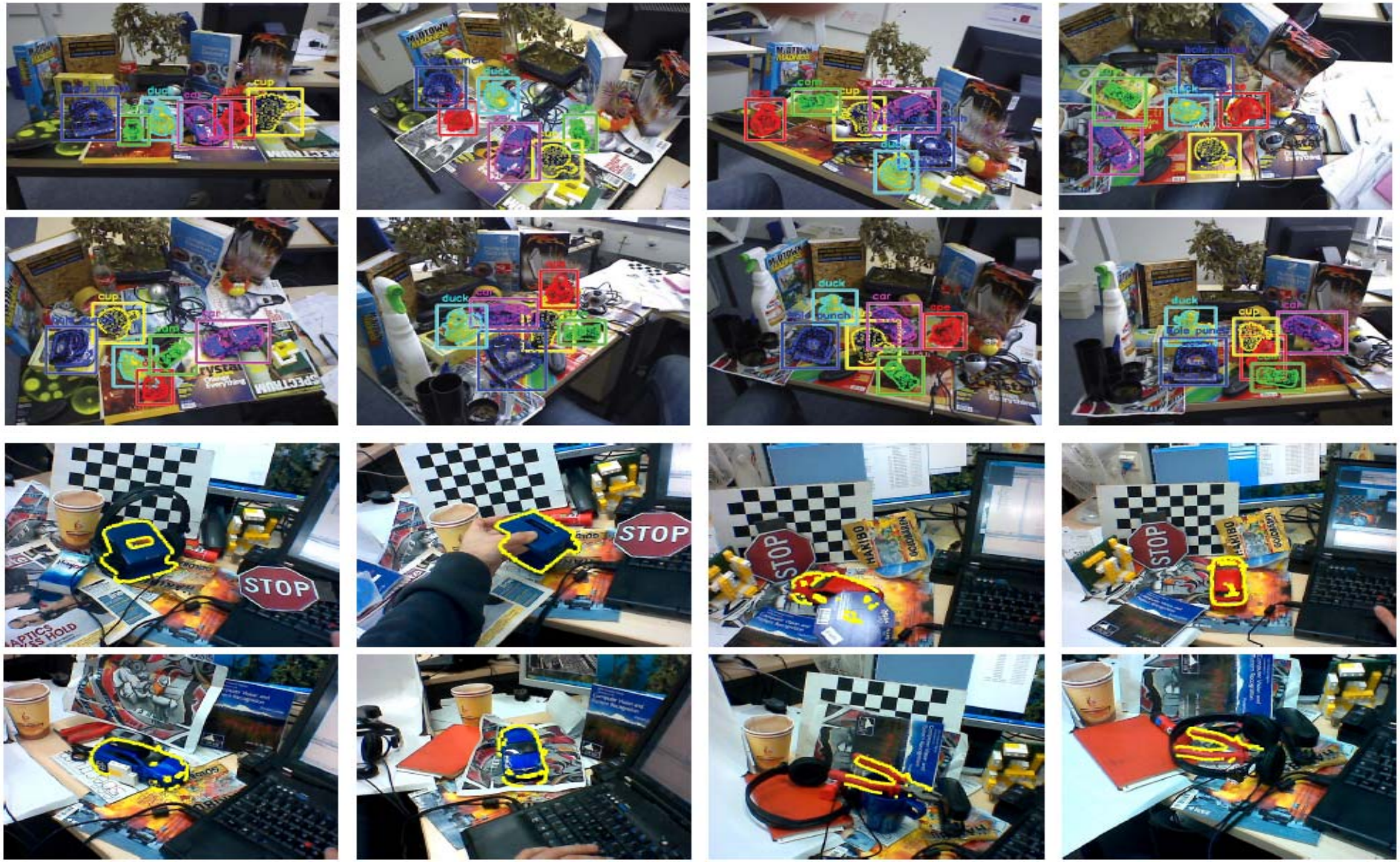








Results:



Results:

Results:

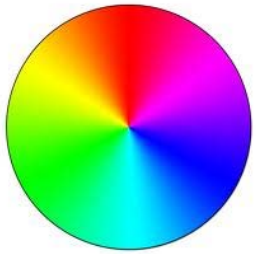
Results:

Revisited: What are the Problems/Solutions with Texture-less objects?

- How to guarantee robustness and speed: **sliding window** & **efficient similarity score**
- How to evaluate this similarity measure robustly: **invariance** & **robust features** & **align features exactly with the image**
- What kind of robust features to use instead: **2D color gradients** & **3D surface normals**
- How to evaluate this similarity measure efficiently: **using response maps** & **invariance** & **feature quantization**
- How to make the approach suitable for online learning: **use template matching**

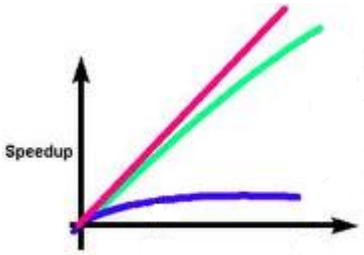
Future Work:

- Use color as feature



- Learn from a 3D model

- Make runtime sublinear



- Add refinement based on 3D model

Questions?

Thank you for your attention!