Computer Vision and Image Processing for Automated Surveillance

Conrad Sanderson

e-mail: conradsand [at] ieee [dot] org
web: http://conradsanderson.id.au
Part 1: Robust Foreground Detection
Part 2: Person Re-Identification
Part 3: Object Tracking on Manifolds
Part 4: Related Work on Surveillance Technologies
Part 5: Rethinking Approaches to Computer Vision Research
Part 1: Robust Foreground Detection

Published in:

  **Improved Foreground Detection via Block-based Classifier Cascade with Probabilistic Decision Integration.**  

  - official version: [http://dx.doi.org/10.1109/TCSVT.2012.2203199](http://dx.doi.org/10.1109/TCSVT.2012.2203199)

Algorithm integrated into a commercial surveillance product!
Early approaches to foreground segmentation:
- obtain clear view of the background
- given a frame, subtract the background
- leftover pixels: foreground

Problems:
- background changes with time (eg. illumination changes)
- noisy ∴ clean up pixels using ad-hoc post-processing (eg. erosion)

Common approach:
- model each background location with a stochastic model
- pixels with low probability: foreground
- adapt model to take into account background changes
- better, but still noisy, still requires ad-hoc post-processing

Core problems:
- classification done at the pixel level
- rich contextual information not taken into account
- foreground segmentation ≠ background subtraction
Proposed Method

1. divide given image into *overlapping* blocks
   - generate low-dimensional descriptors for each block

2. classify each block into foreground/background
   - use a *classifier cascade*

3. for each pixel *integrate* block level decisions
   - results in pixel-level foreground/background segmentation
   - ad-hoc post-processing *not necessary*

4. background model re-initialisation
   - for scenarios with sudden and significant scene changes
Step 1

- divide given image into overlapping blocks
- block size: 8 × 8
- generate low-dimensional descriptors for each block:

\[ d_{(i,j)} = \begin{bmatrix} c_0^r, \ldots, c_3^r, & c_0^g, \ldots, c_3^g, & c_0^b, \ldots, c_3^b \end{bmatrix} \]

\[ c_n^{[k]} = n\text{-th 2D DCT coefficient for the } k\text{-th colour } \{r, g, b\} \]
Step 2

- initial classification of each block into foreground/background
- use a classifier cascade
  - as soon as one classifier classifies as background
- each stage analyses a block from a unique perspective:
  
  **i: texture analysis:** \( p(d_{i,j}) = \mathcal{N}(d_{i,j} \mid \mu_{i,j}, \Sigma_{i,j}) \)
    
    - background model for each block
    - background model trained using a robust method, capable of using a cluttered background
    - background model is adapted during execution

  if \( p(d_{i,j}) > T_{i,j} \) : classify as background

  **ii: shadow:** if \( \text{cosdist}(d_{i,j}, \mu_{i,j}) < C_1 \) : classify as background

  **iii: temporal correlation check:** classify block as background if:
    
    (a) \( d_{i,j}^{[\text{prev}]} \) was classified as background, AND
    
    (b) \( \text{cosdist}(d_{i,j}^{[\text{prev}]}, d_{i,j}) \leq C_2 \)
Step 3

- **no overlapping:**
  misclassification inevitable at the pixel level

- **with overlapping:**
  for each pixel, integrate initial classifications of all relevant blocks:

\[
P(\text{fg} \mid I(x,y)) = \frac{B_{(x,y)}^\text{fg}}{B_{(x,y)}^\text{total}} = \frac{\text{num. of foreground blocks containing pixel } I(x,y)}{\text{total num. of blocks containing pixel } I(x,y)}
\]

- classify pixel \( I(x,y) \) as foreground if \( P(\text{fg} \mid I(x,y)) \geq 0.90 \)

- **no need for any ad-hoc post-processing!**
Trade-Off: Accuracy vs Speed

- sliding block-by-block analysis
- each block is $8 \times 8$
- blocks are overlapping
- 1 pixel advance = max overlap
- 8 pixel advance = no overlap
- $F$-measure $= 2 \frac{\text{recall} \cdot \text{precision}}{\text{recall} + \text{precision}}$
- Achieves real-time processing at the cost of slightly reduced accuracy (2 pixel advance)

image | ground truth | advance: 1 pixel | 2 pixels | 4 pixels | 8 pixels
Comparative Evaluation

Datasets:

- I2R: http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html
- Wallflower: research.microsoft.com/en-us/um/people/jckrumm/WallFlower/TestImages.htm

Compare with:

- GMM based \[^1\] (with morphological post-processing)
- feature histograms \[^2\]
- Normalised Vector Distances (NVD) \[^3\] (block based approach)
- Probabilistic Self-Organizing Maps (SOM) \[^4\]
- Stochastic Approximation (SA) \[^5\]

---


<table>
<thead>
<tr>
<th>original images</th>
<th>ground truth</th>
<th>GMM (with post-processing!)</th>
<th>NVD</th>
<th>proposed method</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image 1" /></td>
<td><img src="image2.png" alt="Image 2" /></td>
<td><img src="image3.png" alt="Image 3" /></td>
<td><img src="image4.png" alt="Image 4" /></td>
<td><img src="image5.png" alt="Image 5" /></td>
</tr>
<tr>
<td><img src="image6.png" alt="Image 6" /></td>
<td><img src="image7.png" alt="Image 7" /></td>
<td><img src="image8.png" alt="Image 8" /></td>
<td><img src="image9.png" alt="Image 9" /></td>
<td><img src="image10.png" alt="Image 10" /></td>
</tr>
<tr>
<td><img src="image11.png" alt="Image 11" /></td>
<td><img src="image12.png" alt="Image 12" /></td>
<td><img src="image13.png" alt="Image 13" /></td>
<td><img src="image14.png" alt="Image 14" /></td>
<td><img src="image15.png" alt="Image 15" /></td>
</tr>
</tbody>
</table>
Quantitative evaluation on I2R dataset:

<table>
<thead>
<tr>
<th>Sequence</th>
<th>0.0</th>
<th>0.2</th>
<th>0.4</th>
<th>0.6</th>
<th>0.8</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>WS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

F-Measure
- GMM based method
- Histogram of features
- NVD based method
- SOM
- SA
- Proposed method

Quantitative evaluation on Wallflower dataset:

<table>
<thead>
<tr>
<th>Sequence</th>
<th>0.0</th>
<th>0.2</th>
<th>0.4</th>
<th>0.6</th>
<th>0.8</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Waving Trees</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FG. Aperture</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Light Switch</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Camouflage Sequence</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bootstrapping</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time of Day</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

F-Measure
- GMM based method
- Histogram of features
- NVD based method
- SOM
- SA
- Proposed method

Conrad Sanderson - Computer Vision and Image Processing for Automated Surveillance - NICTA
On average, the proposed method obtains more accurate foreground detection

More consistent performance across various environments

Does not require any ad-hoc post-processing

Can achieve real-time processing

C++ implementation available as open source code: http://arma.sourceforge.net/foreground/
Part 2: Person Re-Identification

Published in:

- official version: http://dx.doi.org/10.1109/ICIP.2013.6738731
- arXiv pre-print: http://arxiv.org/pdf/1403.0699v1
Given images of a person from camera view 1, find matching person from camera view 2

Difficult:
- imperfect person detection / localisation
- large pose changes
- occlusions
- illumination changes
- low resolution
Popular Previous Approaches

Partial Least Squares (PLS) based [6]

- decompose an image into overlapping blocks
- extracts features from each block: textures, edges, colours
- concatenated into one feature vector (high dimensional)
- learn discriminative dimensionality reduction for each person
- classification: projection to each model + Euclidean distance

**downsides:**

- concatenation = fixed spatial relations between blocks
- **does not allow for movement of blocks!**
- **easily affected** by imperfect localisation and pose variations

---

Symmetry-Driven Accumulation of Local Features (SDALF)\cite{7}

- foreground detection
- two horizontal axes of asymmetry to isolate: head, torso, legs
- use vertical axes of appearance symmetry for torso and legs
- extract: HSV histogram, stable colour regions, textures
- estimation of symmetry affected by deformations & pose variations:
  - noisy features

\footnotesize \textsuperscript{7} M. Farenzena \textit{et al.}: Person re-identification by symmetry-driven accumulation of local features. In: \textit{CVPR} (2010).
Proposed Method

- Aim to obtain a compact & robust representation of an image:
  - allow for imprecise person detection
  - allow for deformations
  - ∴ do not use rigid spatial relations
  - do not use brittle feature extraction based on symmetry

- Steps:
  1. foreground estimation
  2. for each foreground pixel, extract feature vector containing colour and local texture information
  3. represent the set of feature vectors as a covariance matrix
  4. covariance matrix is a point on a Riemannian manifold
  5. map matrix from R. manifold to vector in Euclidean space, while taking into account curvature of the manifold!
  6. use standard machine learning for classification
Feature Extraction

- For each foreground pixel, extract feature vector:

\[ f = [x, y, HSV_{xy}, \Lambda_{xy}, \Theta_{xy}]^T \]

where

- \( HSV_{xy} = [H_{xy}, S_{xy}, \hat{V}_{xy}] = \) colour values of the HSV channels
- \( \Lambda_{xy} = [\lambda_{xy}^R, \lambda_{xy}^G, \lambda_{xy}^B] = \) gradient magnitudes
- \( \Theta_{xy} = [\theta_{xy}^R, \theta_{xy}^G, \theta_{xy}^B] = \) gradient orientations

- (not limited to above, can certainly use other features)

- Given set \( F = \{f_i\}_{i=1}^N \), calculate covariance matrix:

\[ C = \frac{1}{N-1} \sum_{i=1}^{N} (f_i - \mu)(f_i - \mu)^T \]

- Low dimensional representation, independent of image size
How to Compare Covariance Matrices?

- Naive method:
  - brute-force vectorisation of matrix
  - use Euclidean distance between resultant vectors

- Naive method kind-of works, BUT:
  - covariance matrix = symmetric positive definite (SPD) matrix
  - space of SPD matrices = interior of a convex cone in $\mathbb{R}^{D^2}$
  - space of SPD matrices = Riemannian manifold\[^8\]
  - $\therefore$ covariance matrix = point on a Riemannian manifold
  - naive method **disregards** curvature of manifold!
  - geodesic distance: shortest path along the manifold (eg. on a sphere)

How to Measure Distances on Riemannian Manifolds?

- Use Affine Invariant Riemannian Metric (AIRM) \([9]\):

\[
\delta_R(A, B) = \left\| \log \left( B^{-\frac{1}{2}} AB^{-\frac{1}{2}} \right) \right\|_F
\]

- intensive use of matrix inverses, square roots, logarithms \([10]\)
- \(\therefore\) computationally demanding!

- Choose a tangent pole, and map all points to tangent space.

  - tangent space is Euclidean space
  - faster, but less precise
  - **true geodesic distances are only to the tangent pole!**

---


Stein Divergence

- Related to AIRM, but much faster \[11\]
  \[
  \delta_S(A, B) = \log \left( \det \left( \frac{A+B}{2} \right) \right) - \frac{1}{2} \log \left( \det (AB) \right)
  \]
- divergence, **not a true distance!**

**Proposed: Relational Divergence Classification**

- Obtain a set of training covariance matrices \( \{ T \}_{i=1}^N \)
- For matrix \( C \), calculate its Stein divergence to each training covariance matrix:
  \[
  \left[ \delta_S(C, T_1) \; \delta_S(C, T_2) \; \cdots \; \delta_S(C, T_N) \right] \in \mathbb{R}^N
  \]
- In effect, we have **mapped** matrix \( C \) from manifold space to Euclidean space, while (approximately) taking into account manifold curvature
- Can now use **standard** machine learning methods

Comparative Evaluation

- After mapping from manifold space to Euclidean space, use LDA based classifier
- Use ETHZ dataset [12]
  - captured from a moving camera
  - occlusions and wide variations in appearance
- Compare with:
  - directly using the Stein divergence
  - Histogram Plus Epitome (HPE) [13]
  - Partial Least Squares (PLS)[14]
  - Symmetry-Driven Accumulation of Local Features (SDALF)[15]

---

RDC = Relational Divergence Classification (proposed method)
Stein = direct use of Stein divergence (no mapping)
HPE = Histogram Plus Epitome
PLS = Partial Least Squares
SDALF = Symmetry-Driven Accumulation of Local Features
Part 3: Object Tracking on Manifolds

Published in:


Object tracking is hard:

- occlusions
- deformations
- variations in pose
- variations in scale
- variations in illumination
- imposters / similar objects
Tracking algorithms can be categorised into:

1. **generative tracking**
   - represent object through a particular appearance model
   - search for image area with most similar appearance
   - examples: mean shift tracker [16] and FragTrack [17]

2. **discriminative tracking**
   - treat tracking as binary classification task
   - discriminative classifier trained to explicitly separate object from non-object areas
   - example: Multiple Instance Learning (MILTrack) [18]
   - example: Tracking-Learning-Detection (TLD) [19]
   - requires larger training dataset than generative tracking

---

Promising approach for generative tracking:

→ model object appearance via subspaces

- originated with the work of Black and Jepson \[20\]
- apply eigen decomposition on a set of object images
- resulting eigen vectors define a linear subspace
- subspaces able to capture perturbations of object appearance

Many developments to address limitations:

- sequentially update the subspace \cite{21}\cite{22}
- more robust update of the subspace \cite{23}\cite{24}\cite{25}
- online updates using distances to subspaces on Grassmann manifolds \cite{26}

But still not competitive with discriminative methods!

\begin{itemize}
\item \text{Danijel Skocaj et al.: Weighted and robust incremental method for subspace learning.} In: ICCV (2003).
\item \text{Yongmin Li: On incremental and robust subspace learning.} In: Pattern Recognition 37.7 (2004).
\item \text{Jongwoo Lim et al.: Incremental learning for visual tracking.} In: NIPS (2004).
\item \text{D.A. Ross et al.: Incremental learning for robust visual tracking.} In: IJCV 77.1-3 (2008).
\item \text{T. Wang et al.: Online subspace learning on Grassmann manifold for moving object tracking in video.} In: IEEE ICASSP (2008).
\end{itemize}
Two major shortcomings in all subspace based trackers:

1. **mean** of the image set is not used
   - the mean can hold useful discriminatory information!

2. search for object location is typically done using **point-to-subspace distance**
   - compare a candidate image area from ONE frame against the model (multiple frames)
   - easily affected by drastic appearance changes (e.g., occlusions)
Point-to-subspace distance

- each image is represented as a point
- object model (subspace) is conceptually represented as a line

- previously tracked frames are disregarded when comparing candidate frames to object model
- reduces memory of the system
- can easily lead to incorrect frame selection

Minimum point to subspace distance

Conrad Sanderson - Computer Vision and Image Processing for Automated Surveillance - NICTA
Proposed Tracking Approach

Comprised of 4 intertwined components:

1. particle filtering framework (for efficient search)

2. model appearance of each particle as an affine subspace
   - takes into account tracking history (longer memory)
   - takes into account the mean

3. object model: bag of affine subspaces
   - continuously updated set of affine subspaces
   - longer memory
   - handles drastic appearance changes

4. likelihood of each particle according to object model:
   (i) distance between means
   (ii) distance between bases: subspace-to-subspace distance
1. Particle Filtering Framework

- Using standard particle filtering framework \[^{27}\]
- History of object’s location is parameterised as a distribution
  - set of particles represents the distribution
  - each particle represents a location and scale:
    \[
    z_i^{(t)} = [x_i^{(t)}, y_i^{(t)}, s_i^{(t)}]
    \]
- Use distribution to create a set of candidate object locations in a new frame
- Obtain appearance of each particle: \(A_i^{(t)}\)
- Choose new location of object as the particle with highest likelihood according to object model \(\mathcal{B}\):
  \[
  z_\ast^{(t)} = z_j^{(t)}, \quad \text{where} \quad j = \arg\max_i p(A_i^{(t)}|\mathcal{B})
  \]

2. Model Appearance of Each Particle as an Affine Subspace

- Affine subspace represented as a 2-tuple:

\[ A_i(t) = \{ \mu_i(t), U_i(t) \} \]

- \( \mu \): mean
- \( U \): subspace basis

- Appearance includes:
  1. appearance of the \( i \)-th candidate location
  2. appearance of tracked object in several preceding frames
3. Object Model: Bag of Affine Subspaces

- Drastic appearance changes (eg. occlusions) adversely affect subspaces.
- Instead of modelling the object using only one subspace, use a bag of subspaces:

\[ \mathcal{B} = \{A_1, \cdots, A_K\} \]

- Simple model update: the bag is updated every \( W \) frames by replacing the oldest affine subspace with the newest.
4. Likelihood of Each Particle According to Object Model

- Particle filtering framework requires: \( p\left( A_i(t) | B \right) \)
- Appearance of each candidate area: \( A_i(t) = \{ \mu_i(t), U_i(t) \} \)
- Object model: \( B = \{ A_1, \cdots, A_K \} \)
- Our definition: \( p\left( A_i(t) | B \right) = \sum_{k=1}^{K} \hat{p}\left( A_i(t) | B[k] \right) \)

- \( B[k] \) is the \( k \)-th affine subspace in bag \( B \)
- \( \hat{p}\left( A_i(t) | B[k] \right) = \frac{p\left( A_i(t) | B[k] \right)}{\sum_{j=1}^{N} p\left( A_j(t) | B[k] \right)} \), where \( N = \) num. of particles
- \( p\left( A_i(t) | B[k] \right) \approx \exp\left\{ -\text{dist}(A_i(t), B[k]) \right\} \)

distance between affine subspaces
Define the **distance** between two affine subspaces as:

\[
dist(\mathcal{A}_i, \mathcal{A}_j) = \alpha \hat{d}_o (\mu_i, \mu_j) + (1 - \alpha) \hat{d}_g (U_i, U_j)
\]

- \(\hat{d}_o (\mu_i, \mu_j)\) = normalised Euclidean distance between means
- \(\hat{d}_g (U_i, U_j)\) = normalised geodesic distance between bases

**Grassmann manifolds:**

- space of all \(n\)-dimensional linear subspaces of \(\mathbb{R}^D\) for \(0 < n < D\)
- a point on Grassmann manifold \(\mathcal{G}_{D,n}\) is a \(D \times n\) matrix

Geodesic distance between subspaces \(U_i\) and \(U_j\) is:

\[
dg (U_i, U_j) = \| [\theta_1, \theta_2, \cdots, \theta_n] \|
\]

- \([\theta_1, \theta_2, \cdots, \theta_n]\) = vector of principal angles
- \(\theta_1 = \) smallest angle btwn. all pairs of unit vectors in \(U_i\) and \(U_j\)
- principal angles are computed via SVD of \(U_i^T U_j\)
- each image set is represented as a point on a Grassmann manifold
- explicitly takes into account previously tracked frames
Computational Complexity

- Generation of new affine subspace:
  - patch size: $H_1 \times H_2$
  - represent patch as vector: $D = H_1 \times H_2$
  - use patches from $P$ frames
  - $\therefore$ SVD of $D \times P$ matrix
  - $D \gg P$
  - using optimised thin SVD\textsuperscript{[28]}: $O(Dn^2)$ operations
  - $n = \text{number of basis vectors}$

- To keep computational requirements relatively low:
  - patch size: $32 \times 32$
  - number of frames: 5
  - number of basis vectors: 3

Comparative Evaluation

- Evaluation on 8 commonly used videos in the literature
- Compared against recent tracking algorithms:
  - Tracking-Learning-Detection (TLD)\textsuperscript{[29]}
  - Multiple Instance Learning (MILTrack) \textsuperscript{[30]}
  - Sparse Collaborative Model (SCM) \textsuperscript{[31]}
- Qualitative and quantitative evaluation

\textsuperscript{29} Z. Kalal et al.: *Tracking-learning-detection*. In: *IEEE PAMI* 34.7 (2012).
\textsuperscript{30} B. Babenko et al.: *Robust object tracking with online multiple instance learning*. In: *IEEE PAMI* 33.8 (2011).
The table compares the proposed method with TLD (PAMI 2012), MILTrack (PAMI 2011), and SCM (CVPR 2012) in terms of object tracking performance. The diagrams illustrate the tracking results at different frames, highlighting the accuracy and effectiveness of each method.
Quantitative Results

- Used two measures:

  1. **centre location error**: distance between the centre of the bounding box and the ground truth object position

  2. **precision**: percentage of frames where the estimated object location is within a pre-defined distance to ground truth

---

**average error**
(lower = better)

**average precision**
(higher = better)
Future Work

- Affected by motion blurring (rapid motion or pose variations)

- Better update scheme by measuring the effectiveness of new affine subspace before adding it to the bag

- Allow bag size and update rate to be dynamic, possibly dependent on tracking difficulty
Part 4: Related Work on Surveillance Technologies

Overview of our papers on:

- face recognition in realistic scenarios
- shadow removal for improved object detection and tracking
- estimation of true background in cluttered surveillance videos
- face selection for improved recognition in surveillance videos
Y. Wong, M. Harandi, C. Sanderson.
On Robust Face Recognition via Sparse Coding: The Good, The Bad and The Ugly.

- official version: http://dx.doi.org/10.1049/iet-bmt.2013.0033
- arXiv pre-print: http://arxiv.org/pdf/1303.1624v1

Summary:

- Shows that most face recognition systems based on sparse coding:
  - rely on flawed assumptions
  - are inapplicable to realistic scenarios: open-set identification and misalignment (imperfect face detection / localisation)
- Proposes sparse coding on patch-based face representation
  - results in a robust face descriptor
  - robust to face misalignment & environmental variations
  - readily applicable to open set identification and verification
A. Sanin, C. Sanderson, B.C. Lovell.


- official version: http://dx.doi.org/10.1016/j.patcog.2011.10.001
- arXiv pre-print: http://arxiv.org/pdf/1304.1233v1
- C++ source code: http://arma.sourceforge.net/shadows/

Summary:

- Shadow removal is a critical step for improving object detection and object tracking
- Places shadow detection algorithms in a feature-based taxonomy: chromacity, physical, geometry and textures
- Quantitatively compares recent algorithms in terms of shadow detection and discrimination rates, colour desaturation
- Small-region texture based method is especially robust

A Low-Complexity Algorithm for Static Background Estimation from Cluttered Image Sequences in Surveillance Contexts.

Image and Video Processing, 2011.

- official version: http://dx.doi.org/10.1155/2011/164956
- arXiv pre-print: http://arxiv.org/pdf/1303.2465v1
- C++ source code: http://arma.sourceforge.net/background_est/

Summary:

- True background model is unavailable in many practical circumstances: surveillance videos cluttered with foreground objects
- Propose a sequential technique for estimation of static backgrounds
- Background is reconstructed through a Markov Random Field framework
- Image sequences are analysed on a block-by-block basis; clique potentials are computed based on the combined frequency response of the candidate block and its neighbourhood

Patch-based Probabilistic Image Quality Assessment for Face Selection and Improved Video-based Face Recognition.

- official version: http://dx.doi.org/10.1109/CVPRW.2011.5981881
- arXiv pre-print: http://arxiv.org/pdf/1304.0869v2
- surveillance database: http://arma.sourceforge.net/chokepoint/

Summary:

- In face recognition from surveillance videos, face images are captured over multiple frames in uncontrolled conditions.
- Using all face images (including poor quality images) can degrade face recognition performance!
- Current face selection techniques are incapable of simultaneously handling all relevant environmental factors.
- Propose an efficient patch-based face image quality assessment algorithm which quantifies similarity of face images to a probabilistic face model, representing an “ideal” face.
Part 5: Rethinking Approaches to Computer Vision Research
Adapt the main lessons learned\textsuperscript{[32]} from big data:

- more data provides more depth
- some correlations or trends are only visible in large datasets
- don't sample, use all data: sampling throws out information!
- sampling is a leftover from a bygone age: when we had lack of storage & processing power

Implications for computer vision algorithms:

- trade-off between amount of data that can be processed and algorithm complexity
- better to make a fast & “imprecise” algorithm that can go through a lot of data, instead of a slow & “precise” algorithm
- design algorithms from the start to be scalable: parallelisable and able to process chunks of data at a time

Algorithms are currently implemented to run on CPUs:

- stem from **Von Neumann architecture** (1945)
- read instruction, read data, process data, store data, ...
- good for fast processing of spreadsheets
- **inefficient** for computer vision: slow and uses **lots of energy**

Organic brain:

- NOT Von Neumann architecture
- data is encoded and processed in terms of **spikes** (eg. rate of spikes)
- massively parallel execution
- easily deals with incomplete data
- **energy efficient**
CPU →

vs

Brain →

∴ CPUs use lots of energy, get hot, and don’t accomplish much ...
TrueNorth: new computer architecture from IBM Research[33]:

- rough approximation of the organic brain
- NOT simply a hardware implementation of ANNs
- implements interconnected modules of spiking neurons
- implemented using existing CMOS hardware building blocks
- 4096 cores, 1 million neurons, 5.4 billion transistors
- each core has memory ("synapses"), processors ("neurons"), and communication ("axons")

---

Fundamentally different programming approach:
- composing networks of neurosynaptic cores

Successfully implemented computer vision algorithms\[^{34}\] :
- digit recognition, collision avoidance, optical flow, eye detection, ...
- 400 billion synaptic operations per second (SOPS) per watt
- most efficient supercomputer: 4.5 billion FLOPS per watt
- uses less energy: 176,000 times more efficient than a modern CPU running the same brain-like workload

Implications:

- a paradigm shift is on the horizon
- nature of computer vision research will need to adapt to make use of the new architecture

How deep does the rabbit hole go?

- the organic brain already contains excellent vision algorithms, thanks to a few billion years of evolution
- is the code used by the organic brain similar to the code used by TrueNorth?
- if so, can we reverse engineer the pre-existing algorithms in the brain?
- re-implement the reverse engineered algorithms on the TrueNorth architecture?
Questions? Comments?
e-mail: conradsand [at] ieee [dot] org

More papers on computer vision & machine learning:
http://conradsanderson.id.au/papers.html