Background subtractors overview

Made by Viktor Kosikov, MIPT

January 2023

Outline

1. Problem statement
    a. Problem itself
    b. Adjacent tasks
2. Methods
    a. Modelling stages
    b. Naive subtraction
    c. MOG
    d. MOG2
    e. KNN
    f. LSBP
    g. CNT
    h. Overview of DL methods
3. Comparison
    a. Comparison papers
    b. Manual inference comparison
4. Case studies/Usage

Problem statement

Task overview (part 1)

Background subtraction (BS/BGS) - is a common and widely used technique for generating a foreground mask (namely, a binary image containing the pixels belonging to moving objects in the scene) by using static cameras.

Task overview (part 2)

Main presumptions:

- Static (or mostly static camera)
- Minimal training

One more example

Adjacent tasks

Main adjacent task - Moving Camera Background Modelling

Key difference:

- Static camera presumption is broken

Datasets: Davis 2017, SBI, SBM-RGBD etc. (more here)

Noticeable paper: https://arxiv.org/pdf/2209.07923v1.pdf

Cool results to check: https://github.com/bgu-cs-vil/deepmcbm

Methods

Modelling stages

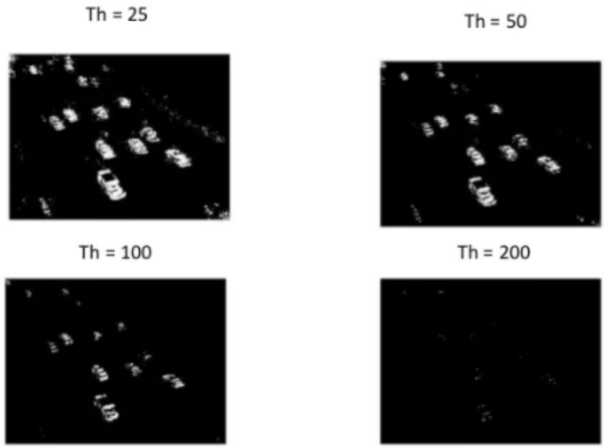
Overall in the production stages of background modelling are the following:

- Background initialization
- Background modelling
- Background maintenance
- Foreground detection

Naive subtraction (part 1)

Naive subtraction is as naive as the following:

$$|Frame_i - Frame_{i-1}| > Threshold$$



girafe ai

Naive subtraction (part 2)

Main problems:

- Too dependant on the threshold
- Reacts badly to lighting changes
- Reacts badly to shades
- Reacts badly to small camera movements

MOG

Paper: An Improved Adaptive Background Mixture Model for Realtime Tracking with Shadow Detection (pdf)

Year: 2002

Authors: P. KaewTraKulPong and R. Bowden

OpenCV implementations: CPU algorithm, CUDA algorithm

MOG (Algorithm outline, part 1)

1. Each pixel is modelled by a mixture of K gaussian distributions
2. Different Gaussians are assumed to represent different colours
3. The weight parameters of the mixture represent the time proportions that those colours stay in the scene
4. The background components are determined by assuming that the background contains B highest probable colours

MOG (Algorithm outline, part 2)

5. The probable background colours are the ones which stay longer and more static. Static single-colour objects trend to form tight clusters in the colour space while moving ones form widen clusters due to different reflecting surfaces during the movement

6. Every new pixel value is checked against existing model components in order of fitness (**new**)

7. New shadow detection heuristic (**new**)

MOG (Important math)

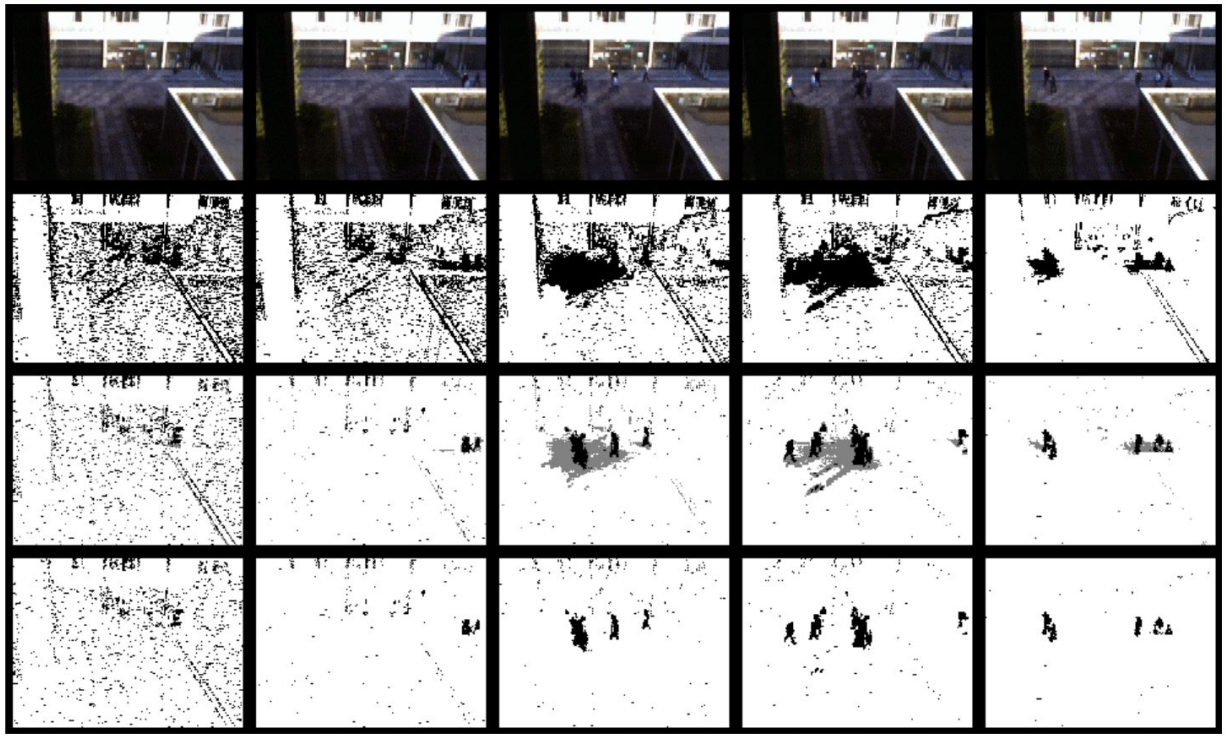
Each pixel in the scene is modelled by a mixture of K Gaussian distributions. The probability that a certain pixel has a value of $\mathbf{x}_N$ at time N can be written as

$$p(\mathbf{x}_N) = \sum_{j=1}^{K} w_j \eta(\mathbf{x}_N; \boldsymbol{\theta}_j)$$

w_k / σ_k fitness value

girafe ai

MOG (pictures)



Figrue 1: The top row displays the original sequence at frames 15, 105, 235, 290 and 1200 respectively. The second row shows the results from Grimson et al's. The last two rows are the results of our proposed method with and without moving shadows displayed in the images. The shadows are shown in grey.

girafe ai

MOG2

Papers:

- Efficient adaptive density estimation per image pixel for the task of background subtraction (2006, Zoran Zivkovic and Ferdinand van der Heijden, link)
- Improved adaptive gaussian mixture model for background subtraction (2004. Zoran Zivkovic, link)

OpenCV implementations: link

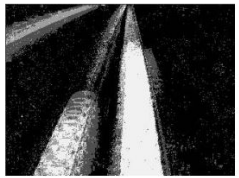
MOG2 (Algorithm outline)

There is 1 the most important feature - number K is chosen algorithmically

Math here is pretty hard (oh, these pre-NN times)

Basically before it was fixed number K, then it is chosen based on Gaussian probability density

MOG2 (Pictures)

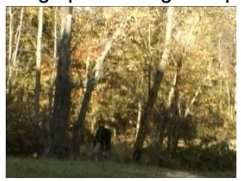


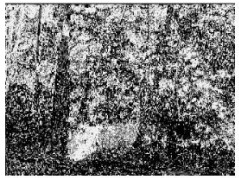
'traffic' sequence
average processing time per frame Old: 19.1ms New: 13.0ms

selected number of modes M

'lab' sequence
average processing time per frame Old: 19.3ms New: 15.9ms

selected number of modes M





'trees' sequence
average processing time per frame Old: 19.7ms New: 19.3ms

selected number of modes M

girafe ai

KNN

Paper: Efficient adaptive density estimation per image pixel for the task of background subtraction (2006, Zoran Zivkovic and Ferdinand van der Heijden, link)

Official OpenCV implementation: link

It is usually very efficient in case foreground image size is very small.

KNN (algorithm outline)

In the paper it's called "balloon variable kernel density estimation"

$$\hat{p}_{\text{non-parametric}}(\vec{x}|\mathcal{X}_T, \text{BG}) \approx \frac{1}{TV} \sum_{m=t-T}^{t} b^{(m)} \mathcal{K}\left(\frac{\|\vec{x}^{(m)} - \vec{x}\|}{D}\right).$$

K - kernel,

D is dynamically chosen set

CNT

Paper: There is no actual paper!

Algorithm was provided by Sagi Zeevi in his github repo (https://github.com/sagi-z/BackgroundSubtractorCNT)

But it is also implemented in OpenCV: link

Pros: **super** fast

CNT (algorithm outline)

It is basically a counter of frames any pixel is in the image.

That’s pretty much it.

DL-based methods

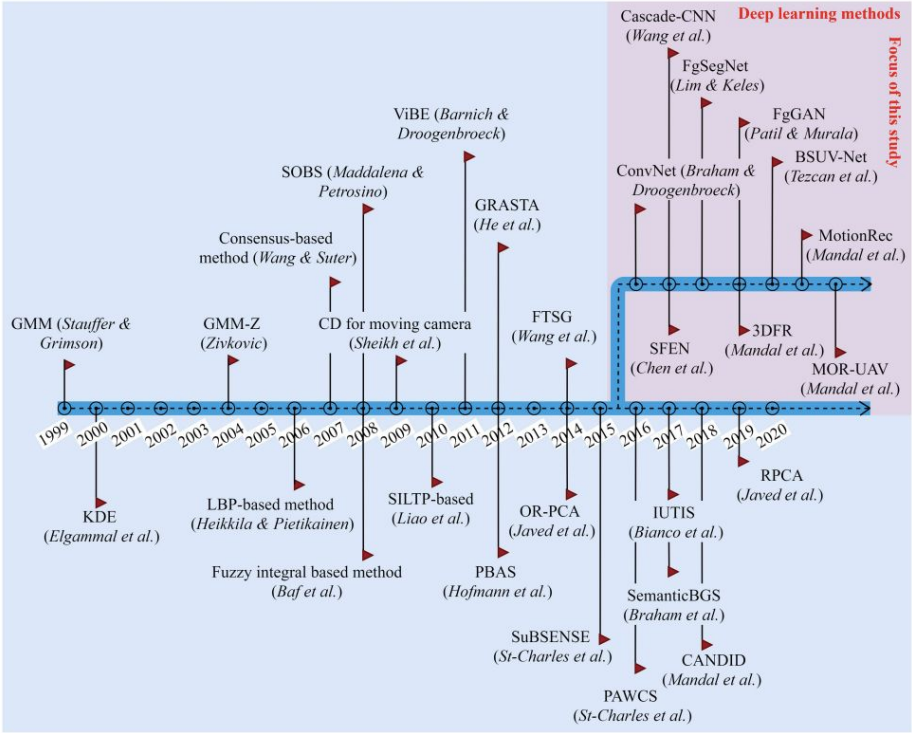
LInk to a great paper - https://arxiv.org/pdf/2105.01342.pdf

TL;DR - basically it is a lot of different versions of CNN

Main issue - need for labelled dataset

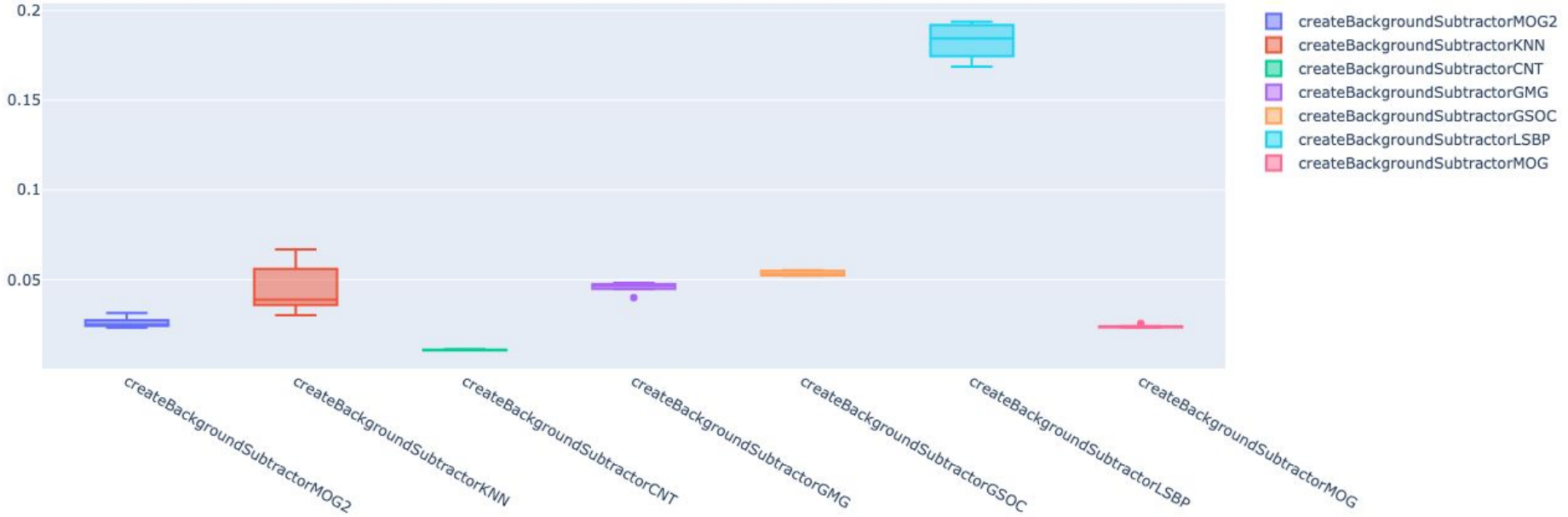
Comparison

History of methods



girafe ai

My code for inference comparison



Some metrics for comparison

Here are the results for basic metrics for algorithms on different frames:

Methods	Precision	Recall	Specificity	PWC	F-Measure
MOG	0.9499	0.9054	0.9990	0.2880	0.9271
MOG2	0.0839	0.9833	0.7784	21.7460	0.1547
GMG	0.6131	0.9869	0.9871	1.2863	0.7564
KNN	0.7549	0.9693	0.9935	0.6988	0.8488

Methods	Precision	Recall	Specificity	PWC	F-Measure
MOG	0.3052	0.5934	0.9493	6.3604	0.4031
MOG2	0.0720	0.8890	0.5700	41.8426	0.1333
GMG	0.4366	0.8940	0.9567	4.5586	0.5867
KNN	0.0794	0.8486	0.6306	36.1553	0.1452

Metrics explanation

$$Precision = \frac{TP}{TP + FP}, Recall = \frac{TP}{TP + FN}$$

$$Specificity = \frac{TN}{TN + FP}, FalseNegativeRate = \frac{fN}{TP + FN}$$

$$FalsePositiveRate = \frac{FP}{FP + TN},$$

$$F - measure = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

$$PWC = 100 \times \frac{FN + FP}{TP + FN + FP + TN}$$

Comparison of pictures

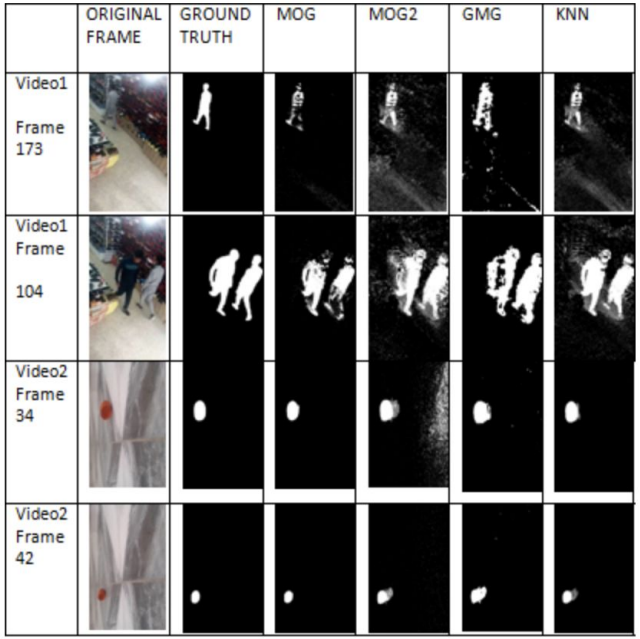


Fig. 1. Comparison the motions detected by different BS algorithms.

girafe ai

Case studies

Paper with use cases

There is a great paper with an overview of different methods across all possible use cases and industries:

Background Subtraction in Real Applications: Challenges, Current Models and Future Directions (Thierry Bouwmans, Belmar Garcia-Garcia), link

Most interesting insights

1. Most of the tasks are solved with basic MOG (somtetimes KNN), it may be because article are outdated
2. Honeybees are observed with KNN (cause they are small, just like we discussed)
3. Camera jitters is still a challenge for such methods, even with the current state of DL models

Thank you for attention

Q&A