Interfețe Vizuale Om-Mașină
IVOM

Cosmin Țoca	
Mail.	cosmin.toca@gmail.com

IVOM

Pedestrian Detection – Aggregated Channel Features

Overview

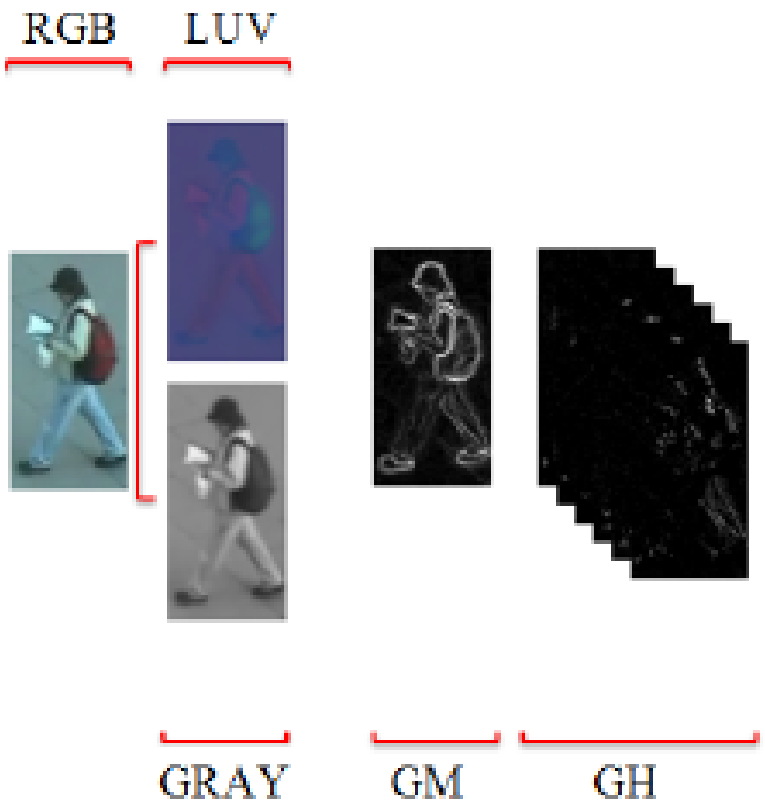
1. Aggregated Channel Features – Descriptors
2. Week Classifiers – Regression Trees
3. Boosted Trees (Modified Discrete AdaBoost) – Learning Algorithm

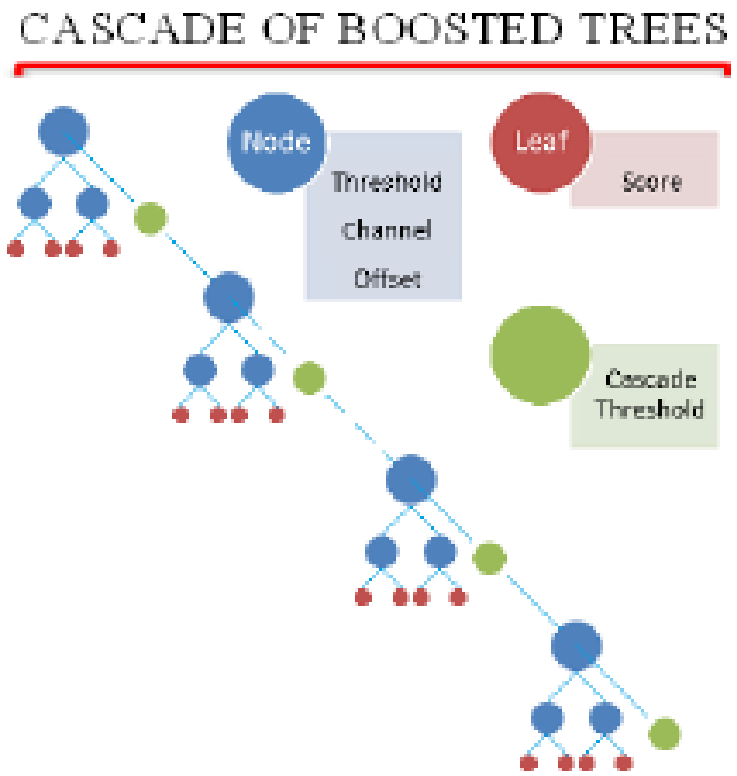
Intro

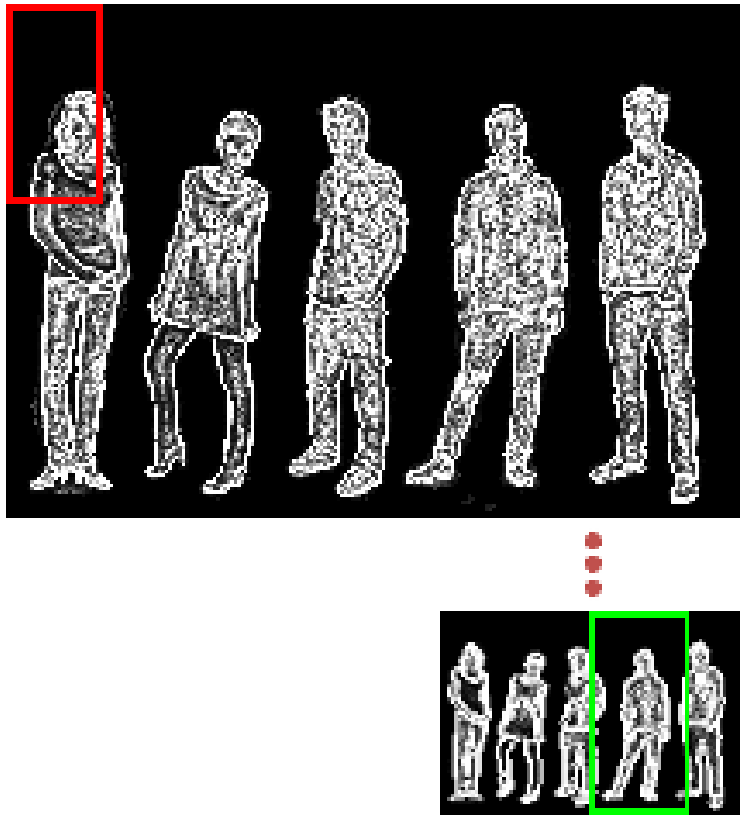


Tasks

Sliding Window Object Detection (Pedestrian Detection)

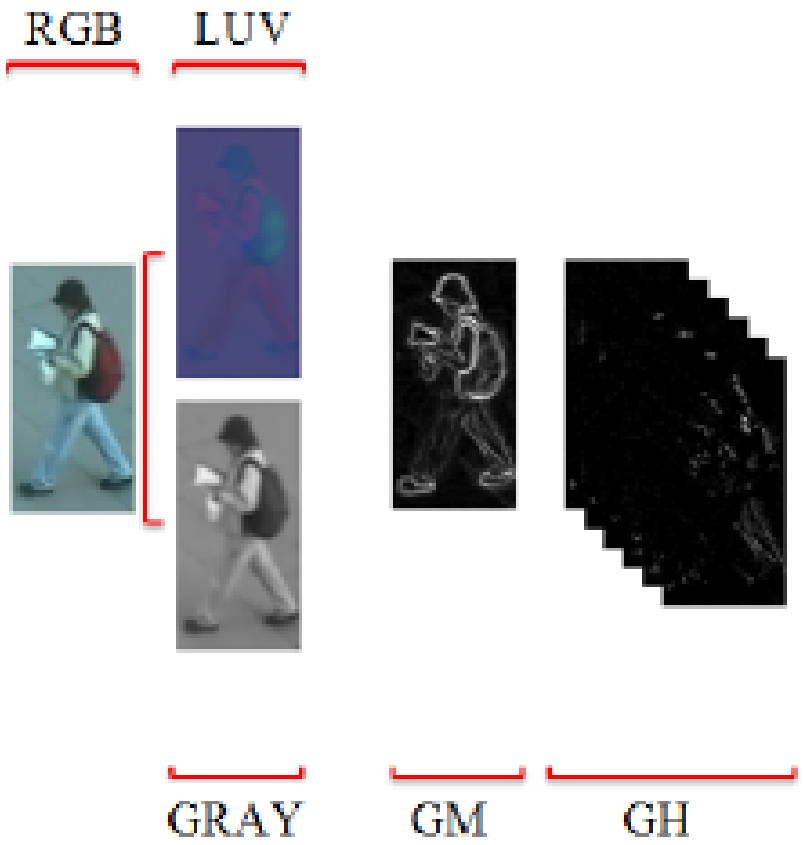






Aggregated Channel Features

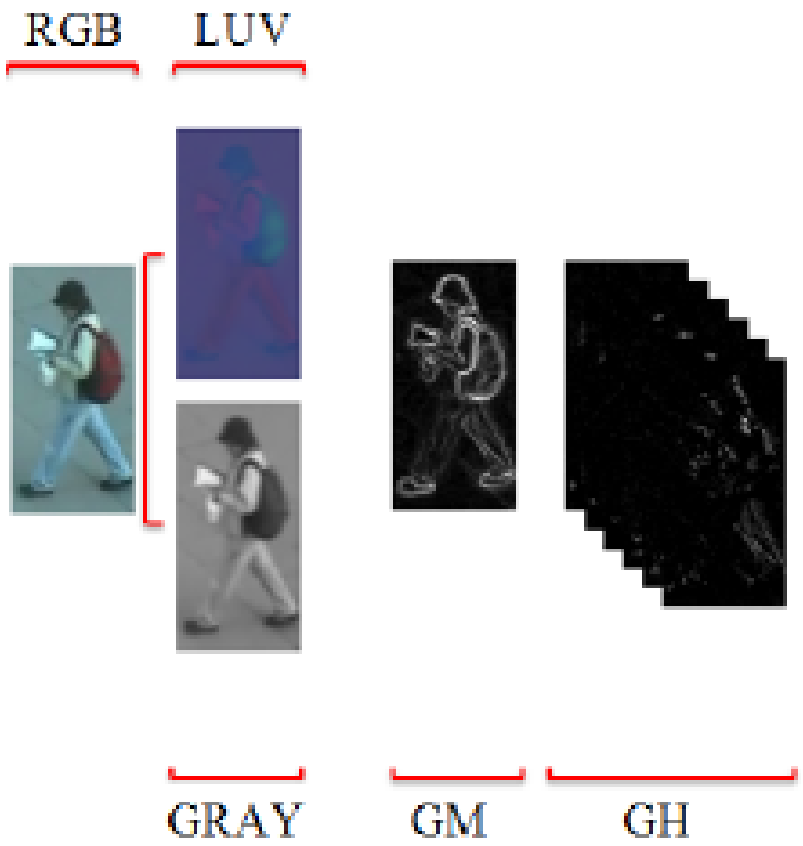
- LUV – is computed by converting the input RGB Image into LUV Color Space



Aggregated Channel Features

- LUV – is computed by converting the input RGB Image into LUV Color Space
- GM (Gradient Magnitude) – is computed by convolution with a discrete derivative mask in both horizontal and vertical directions, and captures undirected edge strength

$$\mathcal{M}(\gamma_\xi) = \sqrt{\frac{\partial}{\partial x}\mathcal{N}_\xi^2 + \frac{\partial}{\partial y}\mathcal{N}_\xi^2}.$$



Aggregated Channel Features

- LUV – is computed by converting the input RGB Image into LUV Color Space
- GM (Gradient Magnitude) – is computed by convolution with a discrete derivative mask in both horizontal and vertical directions, and captures undirected edge strength

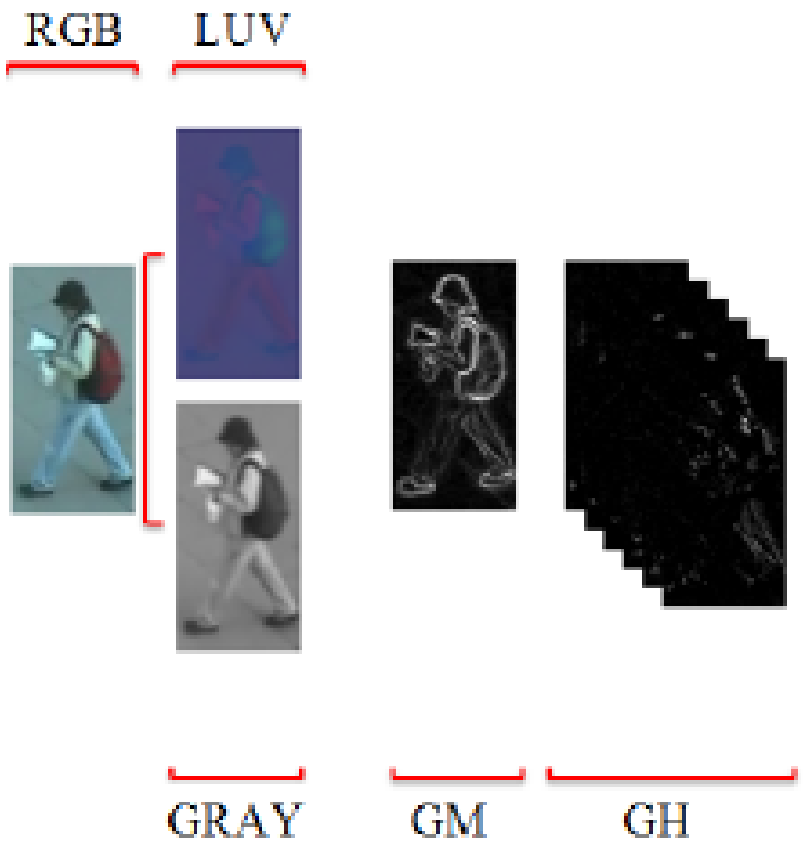
$$\mathcal{M}(\gamma_\xi) = \sqrt{\frac{\partial}{\partial x}\mathcal{N}_\xi^2 + \frac{\partial}{\partial y}\mathcal{N}_\xi^2}.$$

- GH (Gradient Histograms) – are computed by assigning a weighted vote based on the values found in the gradient computation

$$\mathcal{O}(\xi) = \arctan\left(\frac{\partial}{\partial y}\mathcal{N}_\xi / \frac{\partial}{\partial x}\mathcal{N}_\xi\right).$$

$$\mathcal{H}_\theta(\xi) = \delta\Big(\mathcal{O}(\xi)\Big) \cdot \mathcal{M}(\xi)$$

$$\delta\Big(\mathcal{O}(\xi)\Big) = \begin{cases} 1, & \mathcal{O}(\xi) = \theta \\ 0, & \mathcal{O}(\xi) \neq \theta \end{cases}$$



Boosted Trees – Learning Algorithm

ACF Training	
Initialization	*Initialize positive and negative training samples and consider k = 4 levels of cascade length* L=[32, 128, 512, 2048]
Cascade Training - Loop over L=[32, 128, 512, 2048] -	*Train a cascade according to boosting scheme "Cascade Training" considering* L(k) *cascade length*
Bootstrapping	*Evaluate cascade over a larger set of negative samples and improve the negative dataset with hard negatives*
Finish	Output final cascade.

Boosted Trees – Learning Algorithm

Cascade Training	
Initialize the weights	Given the set of positive $\{\boldsymbol{x}_i, 1\}_{i=1}^{Np}$ and negative $\{\boldsymbol{x}_i, -1\}_{i=Np+1}^{Nn+Np}$ sample sets $\boldsymbol{x}_i \in \mathbb{R}^d$ Initialize positive and negative weights $w_p(i) = \frac{1}{2Np}\ i = 1, \dots, Np$ $w_n(i) = \frac{1}{2Nn}\ i = Np+1, \dots, Np+Nn$
Decision Trees Training - Loop over L(k) -	Train regression trees according to algorithm "Decision Trees Training". The learning procedure returns a tree classifier $h_t(\boldsymbol{x})$ and also a training error ε_{tree}. Calculate $\alpha_t = max\ (-3, min\left(3, \frac{1}{2} log \frac{1-\varepsilon_{tree}}{\varepsilon_{tree}}\right))$ For all training samples $\{\boldsymbol{x}_i, y_i\}$ calculate score $S_i = \sum_{j=1}^{t} \alpha_j h_j(\boldsymbol{x})$. Update weights $w_{p,n}(i) = e^{\frac{-y_i S_i}{2Np,n}}$ and normalize all $w(i) = \frac{w(i)}{\sum_{i=1}^{Np+Nn} w(i)}$
Finish	Output final classifier as $H(\boldsymbol{x}) = sign(\sum_{j=1}^{L(k)} \alpha_j h_j(\boldsymbol{x}))$

Boosted Trees – Learning Algorithm

Decision Trees Training	
for m=1 to 2^k	
Create a tree Node N(k,m) with the set $S(k,m)$ of samples and their associated weights	
if k = NLevels+1 mark it as a leaf	Calculate leaf prior error $\varepsilon_{priorl} = \frac{\sum w_{pl}}{\sum w_{pl} + \sum w_{nl}}$ Calculate leaf score $Sl = \frac{1}{2} log \frac{1-\varepsilon_{priorl}}{\varepsilon_{priorl}}$ – the leaf label $Ll = sign(Sl)$ Calculate leaf weight as the sum of the sample weights that reach the leaf $Wl = \sum w_{pl}$ Calculate leaf error as $\varepsilon_l = \min(\varepsilon_{priorl}, 1 - \varepsilon_{priorl})$
else"Train a Stump" with sample subset S(k,m) and their associated weights	
	for k=0 to NrLevels+1 If N(k,m) is a leaf output the leaf label Ll break end if $f(k,m) < T(k,m)$ m=$2^m - 1$ else set m=2^m
end	Output tree structure and ε_{tree}

Boosted Trees – Learning Algorithm

Stump Training	
Given the sample set $S(k,m) \subset S(0,1)$containing $\{\boldsymbol{x}_i, y_i\}$ with their associated sample weights $w(i)$, $\boldsymbol{x}_i \in \mathbb{R}^d$	Consider the feature vector $\boldsymbol{F}_i \in \mathbb{R}^d$, $\boldsymbol{F}_i = [f_1, \dots, f_i, \dots, f_d]^T$
Given the weighted probability distribution (PDF) of elements f_i over all set of positive samples i:	$wPDFp[k,j] = \sum_{i=1}^{Np} w_p(i)(f_i == k)$, k = 0..255 over positive samples $wPDFn[k,j] = \sum_{i=1}^{Nn} w_n(i)(f_{ji} == k)$ over negative samples the cumulative weighted distributions (CDF) can be calculated as: $$CDFp,n[k,j] = \sum_{l=1}^{k} wPDFp,n[l,j]$$
	The composite feature that best separates the positive and negative data is calculated as $$b = \underset{j}{\operatorname{argmax}} \left(\max_k (\lvert CDFp[k,j] - CDFn[k,j] \rvert) \right)$$
	The decision rule of the stump will be applied on the b feature as $$T = \underset{k}{\operatorname{argmax}} (\lvert CDFn[k,b] - CDFp[k,b] \rvert)$$ if $f_{bi} < T$ then label the left leaf is $sign(CDFp[T,b] - CDFn[T,b])$. The samples will be treated as inputs for the node N(k+1,$2^m - 1$) else label the right leaf is $-sign(CDFp[T,b] - CDFn[T,b])$. The samples will be treated as inputs for the node N(k+1,2^m).

References

[1] Piotr Dollar Matlab Toolbox
[2] P. Dollár, R. Appel, S. Belongie and P. Perona, “Fast Feature Pyramids for Object Detection”
[3] Woonhyun Nam, Piotr Dollar, Joon Hee Han, “Local Decorrelation for Improved Pedestrian Detection”

Q&A