Universidade Vigo

DETECCIÓN DE PEÓNS PARA PLATAFORMAS CON BAIXOS RECURSOS

Ismael Orge Fernández

Traballo de fin de grao
Escola de Enxeñaría de Telecomunicación
Grao en Enxeñaría de Tecnoloxías de Telecomunicación

Titor:
José Luis Alba Castro

2019

Universidade Vigo

Escola de Enxeñaría de Telecomunicación

Grao en Enxeñaría de Tecnoloxías de Telecomunicación

Mención: Son e Imaxe

Detección de peóns para plataformas con baixos recursos

Autor: Ismael Orge Fernández
Titor(es): José Luis Alba Castro
Curso: 2019-2020

1. Introdución

A detección de obxectos mediante a análise de imaxes reciviu unha gran atención nos últimos anos. A detección de peóns é un subtema moi popular nas últimas investigacións debido as súas diversas aplicacións, como pode ser o seguimento da súa traxectoria ou a predición desta mesma, extendéndonse a distintos campos: seguridade vial, videovixilancia, robótica... A pesar desta atención, o problema está lonxe de alcanzar un punto de saturación e os últimos estudos amosan avances significativos na fiabilidade das detección.

Nos últimos anos, realizáronse diferentes aproximacións ao problema: *HOG, SVM, Haar-like features, redes neurais...* Sendo as redes neurais (NN), en concreto as convolucionais (CNN), as máis investigadas na actualidade. Có auxe destas redes, probáronse novos métodos basados nelas e, aínda que ofrecen unha moi boa aproximación do problema, posúen unha gran desventaxa: a gran cantidade de información e operacións que hai que procesar e que leva á imposibilidade de poder utilizar equipos modestos (sen GPU dedicada por exemplo) para poder empregalas en tempo real.

Neste traballo preténdese precisamente evitar esta necesidade mediante o uso de sistemas de procesado de imaxe tradicional (evitando as redes ou sistemas máis pesados) e a información a priori referente ao problema. Isto é, basándose en que, de forma xeral, pódese dicir que un peón aparece de pé nas imaxes podéndose empregar o seu contorno para caracterízalo[1]. Tendo en conta que os detectores empregan grandes cantidades de características propias da imaxe para catalogala, isto permitirá utilizar características máis significativas en conxunto que se, por exemplo, se empregase un grupo de características xeradas aleatoriamente, como fan as redes.

En definitiva, o obxectivo é chegar a un detector que utilice un mínimo de características o máis signifactivas posíbeis para poder disminuir a carga computacional.

2. Obxectivos

Este proxecto enfócase principalmente na detección de peóns nunha posición ergueita nun entorno urbano mediante o emprego de *Informed Haar-like Features*, proposto no paper de S. Zhang [1]. Isto consiste en adestrar un algoritmo de aprendizaxe automática como clasificador cun determinado conxunto de datos. Neste caso estes datos foron imaxes de peóns.

Para realizar este adestramento foi necesario facer:

1. **Procesado dos datasets**. Os datasets empregados non teñen unha imaxe por peón cun tamaño determinado se non que teñen distintas imaxes con varios peóns nelas. Por iso

faise necesario un procesado dos dataset para separalos adecuademente e poder utilizalos no adestramento do detector, que necesita telos claramente separados e nun tamaño predeterminado.

2. **Xeración dos templates e cálculo das características**. Xerar templates rectangulares en base o xa comentado do contorno do peón e evitando a información redundante así como calcular as características das imaxes a empregar nos adestramentos.

3. **Selección e adestramento dos detectores**. É o punto máis crítico xa que supón precisar que modelos se van a empregar na detección e con qué parámetros, o que terá unha forte influencia no resultado final.

4. **Comparativa entre os distintos detectores adestrados**. Unha comparativa entre os detectores adestrados na que se mostran resultados como o tempo de procesado ou a precisión do clasificador.

3. Desenvolvemento

Nesta sección descríbese a metodoloxía seguida e as decisións técnicas tomadas para a implementación deste proxecto. Para elo, o proceso realizado desglósase nos seguintes puntos:

1. **Procesado dos datasets**. Como se empregaron os datasets e que cambios foron necesarios nos datos.

2. **Extracción da imaxe media de contornos dun peón**. Como se obtivo o mapa de contornos e qué se fixo logo con el.

3. **Xeración dos templates**. Qué son os templates e como se xeraron para este proxecto.

4. **Cálculo das características**. Cómo se obtiveron as características para alimentar os clasficadores e cómo son neste caso.

5. **Adestramento do clasificador** Qué tipo de adestramento se seguiu, e explicación do clasificador que se empregou nel.

Os puntos están ordeados para seguir o proceso de forma lóxica e conteñen as definicións dos termos necesarios para o seu entendemento.

Este proxecto está completamente feito en Python empregando os módulos scikit-learn[2] e CatBoost[3] para os clasificadores e os datasets de INRIA[4] e Caltech[5] para os adestramentos.

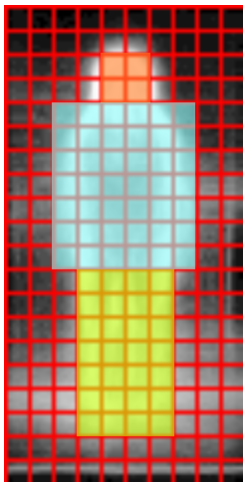
3.1. Procesado dos datasets

Ámbolos datasets (INRIA e Caltech) proporcionan un conxunto de imaxes nas que se poden encontrar peóns xunto cunhas anotacións nas que se localiza en cada imaxe a súa posición é tamaño.

Para realizar o adestramento foi necesario ter un conxunto con imaxes positivas (aquelas que conteñen peóns na imaxe) e outro con imaxes negativas (aquelas que non conteñen peóns). Non obstante a diferencia entre os formatos das anotacións e das imaxes fixo que houbese que tratar cada dataset por separado para extraer os datos que se ían utilizar.

No caso de Caltech, foi necesario extraer as imaxes das secuencias de video e as anotacións dun ficheiro con formato de Matlab a un JSON debido a que estes datos non era facilmente procesábeis en Python có formato orixinal. Tanto para a extracción coma para o formato utilizouse o extractor facilitado por Saito Shunta para este dataset[6]. En canto ás imaxes

(a) Mapa medio de contornos dun peón



(b) Mapa de contornos coas zonas de interese xa segmentadas

Figura 1: En (a) pódese observar o mapa obtido a partir do operador Canny mentres que en (b) vése a segmentación en cabeza(laranxa), torso(azul) e pernas(amarelo).

negativas, tivéronse que extraer aleatoriamente das imaxes principais evitando que contivesen algún fragmento de persoa, xa que este dataset non adxunta un set de negativos.

No caso de INRIA, o dataset xa viu coas imaxes separadas en positivas e negativas e coas anotacións nun formato .txt, polo que non foi necesario facer ningunha conversión. Ademais, este dataset, ao contrario que Caltech, si contén un set de imaxes negativas.

Unha vez obtidos os dous conxuntos de imaxes, recortáronse a un tamaño dado para adestrar logo o clasificador, tendo en conta que no caso dos positivos os peóns aparecen centrados na imaxe e cun certo marxe por cada lado.

Para os dous datasets as imaxes finais recortadas para o adestramento do clasificador foron de 60x120 píxels cómo propón S. Zhang[1][7] tanto para as positivas coma para as negativas. Desta maneira tivéronse dous sets de imaxes:

1. Set de adestramento Caltech: Que consistiu en 24596 imaxes de adestramento (4360 positivos + 20236 negativos) e 25275 imaxes de testeo (3119 positivos e 22156 negativos).

2. Set de adestramento INRIA: Que consistiu en 23435 imaxes de adestramento (3407 positivos + 20028 negativos) e 13091 imaxes de testeo (3069 positivos + 10022 negativos).

3.2. Extracción da imaxe media de contornos dun peón

Para obter información sobre a forma dos peóns, realizouse unha detección de bordes sobre tódalas imaxes positivas, facendo logo un mapa promedio destes bordes. Para a detección empregouse a función Canny[8] da libraría OpenCV[9].

A partir deste mapa, poidéronse etiquetar, delimitando as zonas dunha forma visual, tres partes diferenciadas: *cabeza, torso e pernas* (Fig. 1a).

Unha vez feito isto, dividiuse a imaxe nunha malla de celdas de 6x6 píxels (dimensións que presentan un mellor funcionamento [1]), clasificando cada celda manualmente como unhas das posíbeis opcións dependendo de cal destas zonas teña unha maior cobertura nesa celda: cabeza, torso, pernas ou fondo (Fig. 1b).

3.3. Xeración dos templates

Os clasificadores, como son AdaBoost e CatBoost, necesitan como entrada unha serie de características que se calculan a partir de información extraída da imaxe (cor, magnitude do gradiente...). Non obstante, non é necesario utilizar a completitude desta información, senón que se pode seleccionar certos patróns que se repitan ao longo desta, evitando así, en parte, información redundante e reducindo a cantidade de datos. Estes patróns son os que reciben o nome de *templates*.

Entón, para conseguir os *templates* tense que: coa malla de celdas xa debidamente etiquetada, xeráronse tódalas posíbeis combinacións de etiquetas en matrices dende dimensións 2x1 ata 4x4 e pasáronse ao longo da malla, escollendo como candidatas para o seguinte paso aquelas que coincidisen os seus valores no mapa (sen ter en conta matrices que teñan o mesmo valor en tódolos seus elementos). Estas matrices selecciondas son os *templates* e foron empregadas logo no cálculo das características.

Indo un pouco máis en profundidade na xeración destas matrices: Primeiro, definíronse un conxunto de tamaños tal que

$$S = \{(w, h) \mid w \leq w_m, h \leq h_m, w, h \in \mathbb{N}^+\}, \tag{1}$$

onde w e h indican ancho e alto en termos de celdas para un *template* e w_m e h_w o tamaño máximo destes *templates* que neste caso poden ir dende 2x1 ata 4x4 celdas.

A cada un dos *templates* presentes asígnaselle un conxunto de etiquetas L tal que para cada celda $c(i, j)$ existe unha etiqueta $L(i, j)$ que indica o compoñente presente nesa celda.

Logo, para cada par de dimensións S, deslízase unha xanela (*sliding window*) ao longo de toda a imaxe do mapa de contornos para xerar *templates* en diferentes posicións e con diferentes pesos. Para unha posición (x, y) o *template* xerado dependerá de cantas partes diferentes conteña. Cando sólo haia dúas xerarase un *template* binario e cando haia tres, un ternario (Fig. 2).

O conxunto de *templates* resultante sería

$$T = \{(x, y, s, W) \mid x, y \in \mathbb{N}, s \in S, W \in \mathbb{R}^2\}, \tag{2}$$

onde x e y indican a posición do *template* (en celdas), s as dimensións do template e W unha matriz de pesos determinada en base ao conxunto de etiquetas L (Fig. 2). Un exemplo de matriz W sería para un caso dunha matriz 3x1:

$$W = (-1, -1, 1) \tag{3}$$

onde poderíase corresponder a etiqueta -1 a torso e 1 a pernas.

Como neste caso pode haber *templates* ternarios (pode haber zonas nas que apareza torso, pernas e fondo ou cabeza, torso e fondo), o etiquetado tradicional dos pesos como $+1$ e -1 para empregar en W non é suficiente, polo que se optou por utilizar no caso dos ternarios o conxunto $\{-1, 0, +1\}$ (Fig. 2) mentres que nos binarios usouse $\{-1, +1\}$. O valor 0 no caso dos *templates* ternarios reservouse sempre para o fondo.

En total, e collendo unicamente aqueles *templates* que son distintos entre sí e non *templates* có mesmo valor en tódolos seus elementos, obtivéronse uns **1047 templates distintos** a partir do mapa de contornos. Estes *templates* son os que logo usáronse no seguinte punto para o cálculo de características que se empregaron como entrada dos clasificadores.

3.4. Cálculo das características

Unha vez obtidos os *templates*, para extraer as carecterísticas dunha imaxe foi necesario ter distintos tipos de información dela sobre os que empregar estes *templates*. En base a outras

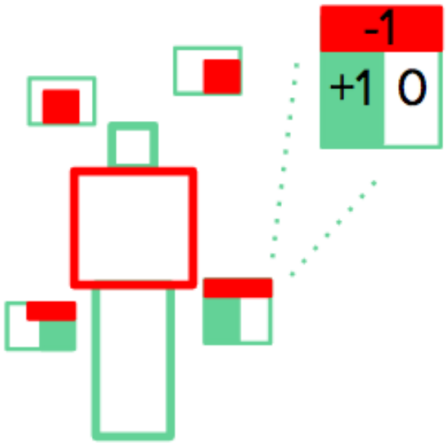


Figura 2: Exemplo de posíbeis templates xerados a partir da forma extraída do mapa de contornos

investigacións [10][1][11], HOG (Histogram of Oriented Gradients) é un dos máis informativos para este tipo de traballos así coma información sobre a cor presente na imaxe. Tamén é comunmente utilizada a información sobre magnitude do gradiente [7].

Para empregar tanto a información de cor coma a de gradiente, as características son multi-canal, isto é, para cada *template* temos unha característica por canal, onde cada un deles é un tipo distinto de información. Elíxense así 10 canais: **3 canais de cor no espazo LUV, 1 canal con información de magnitude de gradiente e 6 canais para HOG**. Desta forma, para cada template tivéronse 10 valores distintos, un por cada canal.

Entón, para calcular as características correspondentes a un *template* tense que:

Dado un *template* $t = (x, y, (w, h), W)$ poderíase calcular a matriz de pesos media como

$$W_{avg} = \frac{sgn(W)}{n_{add}} + \frac{sgn(-W)}{n_{sub}}, \tag{4}$$

onde n_{add} é a cantidade de pesos $+1$ presentes no *template* e n_{sub} a cantidade de -1 (de ahí a elección do conxunto $\{-1, +1\}$ para os templates binarios). Utilizando esta matriz W_{avg} poderíase calcular o valor dunha característica f para un *template* t e un canal k tal que

$$f(t, k) = \sum_{i=1}^{h} \sum_{j=1}^{w} \sigma(x + i, y + j, k) W_{avg}(i, j), \tag{5}$$

onde $\sigma(i, j, k)$ define a suma de todos os valores nas celdas $c(i, j)$ para o canal k.

Tendo en conta que por cada canal hai uns 1047 *templates*, por cada imaxe existen unhas $1047 \times 10 = 10470$ características.

Estas características son as que logo se empregaron como entrada para o adestramento do clasificador. Unha vez adestrado, a partir destas características, calculadas para unha imaxe, será capaz de determinar se existe un peón ou non.

3.5. Adestramento do clasificador

Para o adestramento empregouse o clasificador AdaBoost presente no módulo scikit-learn[2] de Python en ámbolos datasets (INRIA e Caltech). Ademais tamén utilizouse o clasificador

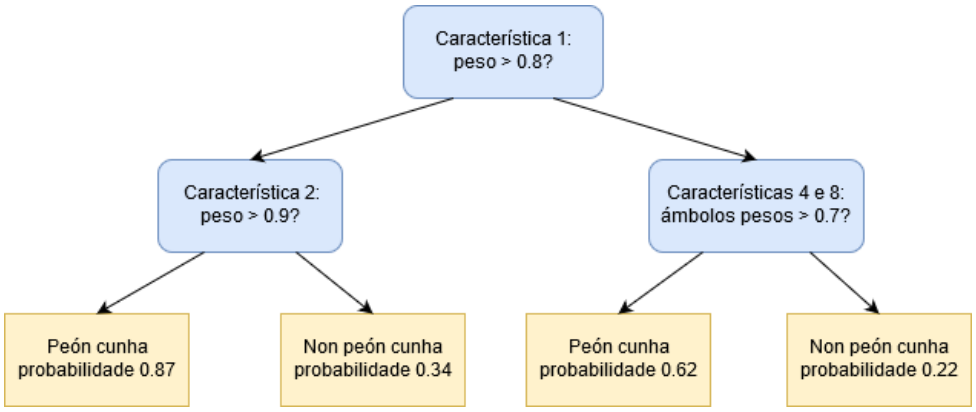


Figura 3: Exemplo dunha posíbel árbore de decisión resultante do adestramento do clasificador.

CatBoost[3] por dar bos resultados nas últimas investigacións e permitir realizar o adestramento en GPU [3].

Ámbolos clasificadores (AdaBoost e CatBoost) son meta-algoritmos de aprendizaxe automática que se empregan en conxunto con outros algoritmos para mellorar os resultados. A saída final é unha suma ponderada da combinación da saída dos outros algoritmos (*weak learners*). Para estes *weak learners* escolleuse neste caso o algoritmo de árbores de decisión. Tamén empregouse o parámetro *learning rate* que nestes clasificadores trátase dun valor que controla canto cambia o modelo en resposta ao erro estimado cada vez que se recalculan os pesos dos *weak leaerners*.

Para tódolos casos utilizouse un adestramento multironda consistente nun adestramento inicial do clasificador que logo é seguido dunha ronda de readestramento. A pesar do recomendado para os mellores resultados (3-4 rondas)[1][11], neste caso elexiuse unicamente de 2 rondas por mor do tempo (o caso de 4 rondas prolongaba o adestramento arredor dunha semana mentres que en 2 rondas quedouse en 2 días). Este foi levado a cabo nun PC cun procesador i7-6850K@3.6GHz, 32Gb de RAM e dúas NVIDIA 1080 GTX 12Gb GPU.

As rondas consistiron en:

1. Ronda inicial: O clasificador iniciouse cunhas 1024 árbores de decisión de profundidade 2 (exemplo na Fig. 3) e 0,001 de learning rate no caso de AdaBoost e 0,01 no caso de CatBoost. Para esta ronda empregouse todo o set de adestramento (sección 3.1).

2. Ronda de readestramento: Unha vez rematada a ronda inicial, obteuse un clasificador funcional que logo se mellorou mediante unha segunda ronda de adestramento. Para esta ronda empregáronse os chamados negativos fortes: aquelas imaxes que, sendo negativas, o clasificador identifica coma positivas. Este tipo de imaxes son máis significativas para o adestramento, xa que supoñen os casos onde o algoritmo adestrado erra na súa clasificación.

   Para obter estes negativos fortes implementouse un **detector** que segue os seguintes pasos:

   a) **Extracción de subimaxes**. A partir da imaxe principal que se lle pasa, o detector reescala a imaxe mediante un factor de escala tantas veces como poda ata que a imaxe sexa igual ou máis pequena que a xanela empregada no adestramento (60x120 píxels). A partir de cada unha destas imaxes reescaladas, extráense todas as subimaxes posíbeis de tamaño 60x120 empregando unha xanela deslizante para recortalas.

   b) **Cálculo dos canais**. Para cada subimaxe obtida, calcúlanse os 10 canais que se van a empregar para a extracción das características (3 canais de cor no espazo LUV, 1

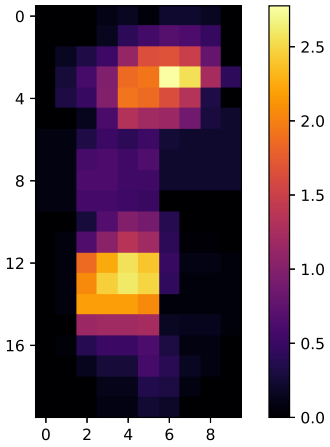


Figura 4: Mapa de pesos das características: Pódese observar como as características máis importantes atópanse principalmente na cabeza e no cambio de torso a pernas. Esta imaxe foi extraída empregando o clasificador CatBoost adestrado para o dataset INRIA e a cor varía en función do peso acumulado en cada celda.

canal con información de magnitude de gradiente e 6 canais para HOG).

c) **Cálculo das características**. Empregando os templates (sección 3.3) e os canais extraidos da subimaxe, o detector calcula as características.

d) **Clasificación da imaxe**. Coas caraterísticas calculadas, o detector alimenta ao clasificador e este devólvelle a probabilidade de que a imaxe sexa un peón (*probabilidade* > 0,5).

Polo tanto, aquelas subimaxes que deron como positivas en imaxes negativas, son as que se poideron clasificar como negativos fortes. Unha vez obtidos al menos uns 10000 destes negativos, engadíronse ao set de entranamento e volveuse a adestrar o clasificador. Para esta ronda, ademais, cambiouse o número de árbores a 2048 de profundidade 2[11].

Os resultados proxectados por cada dataset diverxen bastante entre si, xa que Catech é considerado un dataset moito máis esixente ca INRIA, que aínda sendo un dos máis coñecidos e empregados, é máis pequeno e menos desafiante.

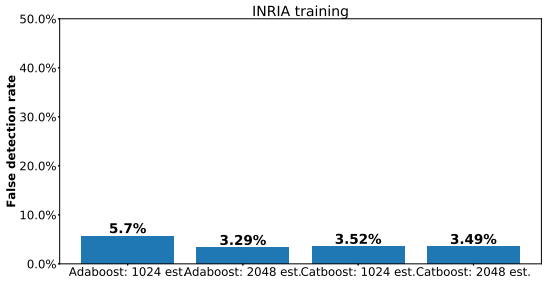
Para facer unha mostra daquelas características máis significativas, na Fig. 4 amósanse as 100 primeiras características que tiveron un maior peso(importancia) no adestramento.

4. Resultados

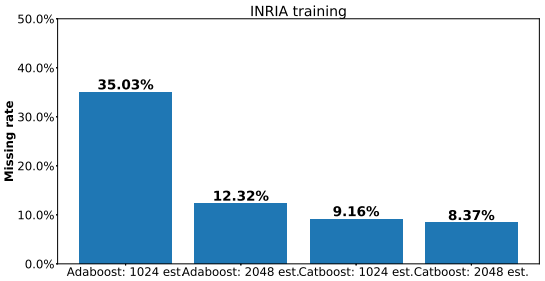
Unha vez adestrados os clasificadores, estes utilizáronse sobre as imaxes de testeo previamente saparadas das utilizadas para o adestramento (sección 3.1). Os resultados obtidos clasifícanse en tres tipos: peóns detectados correctamente, peóns non detectados e peóns que foron detectados incorrectamente (non aparecen na imaxe). Con estes resultados, pódense empregar as seguintes métricas para medir a efectividade dos clasificadores[12]:

1. Detection rate (DR): Porcentaxe de detecciónes correctas sobre tódolos peóns que poden ser detectados.

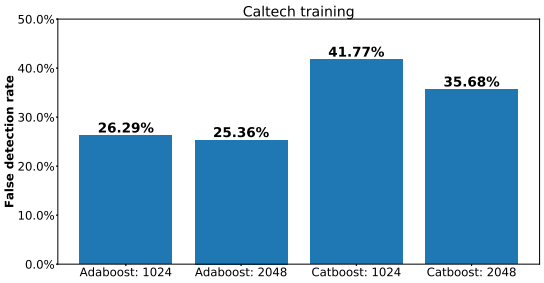
$$DR = \frac{DP}{NP} \times 100\,\%, \tag{6}$$



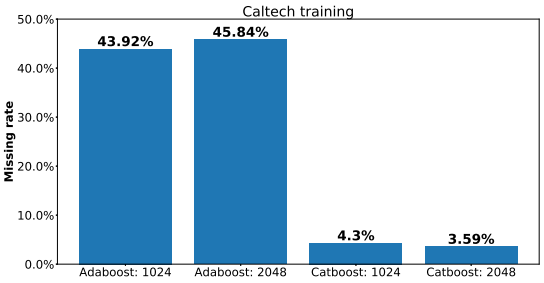
(a) False detection rate para os clasificadores AdaBoost e CatBoost.



(b) Missing rate para os clasificadores AdaBoost e CatBoost



(c) False detection rate para os clasificadores AdaBoost e CatBoost



(d) Missing rate para os clasificadores AdaBoost e CatBoost

Figura 5: Resultados obtidos no adestramento có dataset INRIA 5a 5b e có dataset Caltech 5c 5d. En ámbolos casos móstrase ademais a evolución entre a ronda inicial (barras etiquetadas con 1024) e a de readestramento (barras etiquetdas con 2048).

Onde DP é o número de peóns detectados correctamente e NP o número total de peóns detectábeis.

2. False detection rate (FR): Porcentaxe de falsas deteccións sobre tódolos peóns que poden ser detectados. Unha falsa detección considérase cando detecta un obxecto pero este non é un peón.

$$FR = \frac{FP}{NP} \times 100\,\% \tag{7}$$

Onde FP é o número de obxectos detectados erroneamente coma peóns.

3. Missing rate (MR): Porcentaxe de peóns que non foron detectados sobre tódolos posíbeis peóns detectábeis.

$$MR = \frac{NP - DP}{NP} \times 100\,\% \tag{8}$$

Os resultados obtidos para o set de testeo mostránse na Fig. 5. Nos dous casos (INRIA e Caltech) inclúese ademais do MR o FR para que se poida apreciar mellor a diferencia entre os resultados nos dous datasets. Os resultados correspondentes á ronda inicial son aqueles etiquetados con 1024 (nº de árbores de decisión) e os correspondentes á ronda de readestramento os de 2048. Según o dataset apréciase que:

1. No caso de INRIA os resultados para CatBoost son mellores que no caso de AdaBoost, chegando a baixar dun 12.32 % a un 8.37 % no MR. No caso do FR obsérvase que o seu valor é baixo (3.29 % como min.) pero para poder comparalo cos resultados do paper [1] sería todavía necesario baixalo a uns valores <1 %. Isto é porque, nunha imaxe, o

Factor de escala	Núm. reescalados	paso xanela deslizante	Tempo
1.33	6	60 px	21.35 s
1.09	6	60 px	41.48 s
1.33	10	60 px	22 s
1.09	10	60 px	49.18 s
1.33	3	6 px	1714.82 s
1.33	6	6 px	1997.82 s
1.09	6	6 px	3479.2 s

*Cadro 1: Tempo de inferencia para o clasificador **CatBoost** para unha imaxe de 1024×605 píxels en función do factor de escala para a imaxe, o número de reescalados e o tamaño do movemento en píxels da xanela deslizante ao percorrer a imaxe de entrada. Para facerse unha idea, no caso que máis tarda o clasificador está procesando un total de **65169 subimaxes**.*

Factor de escala	Núm. reescalados	paso xanela deslizante	Tempo
1.33	6	60 px	62.97 s
1.09	6	60 px	115.32 s
1.33	10	60 px	64.30 s
1.09	10	60 px	145.56 s
1.33	3	6 px	4633.17 s
1.33	6	6 px	5812.42 s
1.09	6	6 px	10156.83 s

*Cadro 2: Tempo de inferencia para o clasificador **AdaBoost** para unha imaxe de 1024×605 píxels en función do factor de escala para a imaxe, o número de reescalados e o tamaño do movemento en píxels da xanela deslizante ao percorrer a imaxe de entrada.*

clasificador explicado na sección 3.5 pode chegar a procesar máis de 60000 subimaxes e, con un FR de 3.29 %, só con que 1000 subimaxes sexan positivas isto supón ter ata 32.9 falsos positivos por imaxe (FPPI). Non obstante, para os valores obtidos en [1] fálase duns 0.1 FPPI, o que demostra o lonxe que están estes resultados do pretendido.

En canto á mellora nos resultados en MR de CatBoost respecto a AdaBoost, esta é coherente coa mellora demostrada polos seus creadores[3].

2. En Caltech os resultados xa son máis dispares, notándose a diferencia de esixencia entre os datasets. Neste caso problabemente o que máis condiciona os resultados é a falta de máis rondas de readestramento engadindo negativos fortes. Isto apréciase no elevado valor en tódolos casos do FR (baixando a un mínimo duns 25.36 %). No MR, a diferencia presente entre ámbolos clasificadores explícase con que no caso de CatBoost consegue reducilo a costa de aumentar considerabelmente o FR, é dicir, deixando pasar máis falsos positivos.

En canto ao tempo de inferencia do clasificador, empregouse o detector descrito na sección 3.5.

Nel, a imaxe orixinal é reescalada varias veces aplicando un factor de escala e empregando un NMS (Non Maximum Suppression) para en caso de aparecer varias deteccións que se solapen, que só conten como unha.

Nos cadros 1 e 2 amósase o tempo que tarda o detector en procesar toda unha imaxe de 1024 × 605 píxels empregando un único proceso (sen utilizar concurrencia), variando certos parámetros como son o factor de escala, número de reescalados e a cantidade de píxels que se move a xanela cada vez que procesa unha subimaxe.

No caso de utilizar concurrencia, no PC empregado para este proxecto os tempo podíanse

reducir nun factor de 1/12 (número de núcleos do PC). Non obstante, aínda usando isto, os tempos están moi lonxe de podelos utilizar para tempo real, tendo que reducilos a, polo menos, 50ms (20Hz ou 20fps) por imaxe.

No Anexo C móstranse algunhas deteccións obtidas co clasificador Catboost sobre imaxes de testeo.

5. Conclusións

A pretensión deste proxecto era atinxir un clasificador cuns resultados razoábeis que permitise prescindir de ter que utilizar redes neurais profundas e que poidese funcionar en sistemas con baixos recursos. Para facerse unha idea, coa rede ResNet152[13] de 152 capas na súa implementación en Python, para unha imaxe de 1024×605 píxels (como a empregada nos cadros 1 e 2) o tempo de inferencia é de 4.607s (0.22fps) en media na CPU. Isto fai obvio que non é factíbel empregala para unha aplicación en tempo real.

No caso dos clasificadores adestrados neste proxecto, o seu tempo de procesado é moito máis elevado polo que tampouco son válidos para ese tipo de aplicación.

Para poder empregar os clasificadores en tempo real faise necesario implementalo noutra lenguaxe, xa que Python pode chegar a ser 700 veces máis lento ca C++ e 70 veces máis que Matlab [14]. Ademais á vista dos MR, é necesario aumentar o número de negativos fortes nos adestramentos para coseguir reducir o FR e, por conseguinte, o MR. É o caso de [11] onde empregaron catro rondas de 10000 negativos fortes cada unha. Isto leva tamén á necesidade dunha gran cantidade de negativos, xa que extraer 10000 en cada ronda cada vez vólvese máis difícil ao funcionar moito mellor o clasificador.

No caso presente, chegando a usar un conxunto de imaxes negativas de máis de 3000 non foi abondo para acadar os 10000 negativos para a segunda ronda de readestramento (recordando que os negativos fortes son do tamaño co que se adestrou o clasificador, polo que nunha imaxe negativa pode haber máis de 20 negativos fortes normalmente).

Non obstante, aínda con estos resultados en canto a MR e FR os resultados con CatBoost son claramente moito mellores que no caso de AdaBoost, polo que poderíase acadar uns bos resultados levando a cabo as melloras citadas.

Ademais hoxe en día o dataset de INRIA consíderase insuficiente para conseguir bos resultados en case toda a documentación relacionada coa detección de peóns polo que tería máis sentido centrarse en Caltech, ou noutros datasets coma KITTI[15].

Cabe destacar tamén que se empregaron tódalas características para facer as deteccións, o que supón calcular uns 10470 valores por imaxe, cando se poderían escoller aquelas con un mellor resultado, limitando os cálculos a por exemplo unhas 1000 características.

Finalmente, engadindo certas técnicas habituais no procesado de imaxe, como é unha boa selección da zona de interese para a búsqueda de peóns ou facer a búsqueda unicamente nas escalas correspondentes coñecendo a distancia a que se encontran (facendo unha vista de paxaro), é posíbel acadar os resultados que se buscaban para este proxecto.

Referencias

[1] S. Zhang, C. Bauckhage, and A. B. Cremers, "Informed haar-like feature improve pedestrian detection," *CVPR*, 2014. [Online]. Available: https://zpascal.net/cvpr2014/Zhang_Informed_Haar-like_Features_2014_CVPR_paper.pdf

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python,"

Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011. [Online]. Available: http://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf

[3] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: unbiased boosting with categorical features," 2018. [Online]. Available: https://papers.nips.cc/paper/7898-catboost-unbiased-boosting-with-categorical-features.pdf

[4] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," 2005. [Online]. Available: https://hal.inria.fr/file/index/docid/548512/filename/hog_cvpr2005.pdf

[5] P. Dollár, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: A benchmark," *CVPR*, 2009. [Online]. Available: https://journals.sagepub.com/doi/pdf/10.1177/1729881417749949

[6] S. Shunta. (2015) Caltech pedestrian dataset converter. [Online]. Available: https://github.com/mitmul/caltech-pedestrian-dataset-converter

[7] S. A. Viesca and B. Garcia, "Informed haar-like features improve for pedestrian detection," 2016. [Online]. Available: https://web.stanford.edu/class/cs231a/prev_projects_2016/Alarcon_Garcia.pdf

[8] J. Canny, "A computational approach to edge detection," in *Readings in computer vision*. Elsevier, 1987, pp. 184–203. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.420.3300&rep=rep1&type=pdf

[9] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[10] P. Dollár, Z. Tu, P. Perona, and S. Belongie, "Integral channel features," 2009. [Online]. Available: https://authors.library.caltech.edu/60048/1/dollarBMVC09ChnFtrs.pdf

[11] S. Zhang, R. Benenson, and B. Schiele, "Filtered channel features for pedestrian detection," *CVPR*, vol. 1, no. 2, 2015. [Online]. Available: https://www.cv-foundation.org/openaccess/content_cvpr_2015/app/1B_070.pdf

[12] X. Zhang, H. Gao, C. Xue, J. Zhao, and Y. Liu, "Real-time vehicle detection and tracking using improved histogram of gradient features and kalman filters," *International Journal of Advanced Robotic Systems*, vol. 15, no. 1, 2018. [Online]. Available: https://journals.sagepub.com/doi/pdf/10.1177/1729881417749949

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[14] P. Ramachandran. A beginners guide to using python for performance computing. [Online]. Available: https://github.com/mitmul/caltech-pedestrian-dataset-converter

[15] A. Geiger, P.Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kittidataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2018. [Online]. Available: https://journals.sagepub.com/doi/pdf/10.1177/1729881417749949

[16] P. Viola, M. J. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," *International Journal of Computer Vision*, vol. 63, no. 2, pp. 153–161, 2005. [Online]. Available: https://link.springer.com/content/pdf/10.1007/s11263-005-6644-8.pdf

[17] S. Zhang, R. Benenson, M. Omran, J. Hosang, and B. Schiele, "Towards reaching human performance in pedestrian detection," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 973–986, 2017.

[18] P. Dollár, R. Appel, S. Belongie, and P. Perona, "Fast feature pyramids for object detection," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 8, pp. 1532–1545, 2014. [Online]. Available: https://authors.library.caltech.edu/49239/7/DollarPAMI14pyramids_0.pdf

[19] P. Dollár, R. Appel, and W. Kienzle, "Crosstalk cascades for frame-rate pedestrian detection," in *European Conference on Computer Vision*. Springer, 2012, pp. 645–659. [Online]. Available: https://authors.library.caltech.edu/59956/1/DollarECCV12crosstalkCascades.pdf

[20] R. Benenson, M. Mathias, R. Timofte, and L. Van Gool, "Pedestrian detection at 100 frames per second," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 2903–2910. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.408.651&rep=rep1&type=pdf

[21] R. Benenson, M. Mathias, T. Tuytelaars, and L. Van Gool, "Seeking the strongest rigid detector," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 3666–3673. [Online]. Available: http://openaccess.thecvf.com/content_cvpr_2013/papers/Benenson_Seeking_the_Strongest_2013_CVPR_paper.pdf

[22] A. Daniel Costea and S.Ñedevschi, "Semantic channels for fast pedestrian detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2360–2368. [Online]. Available: http://openaccess.thecvf.com/content_cvpr_2016/papers/Costea_Semantic_Channels_for_CVPR_2016_paper.pdf

[23] W.Ñam, P. Dollár, and J. H. Han, "Local decorrelation for improved pedestrian detection," in *Advances in Neural Information Processing Systems*, 2014, pp. 424–432.

[24] S. Paisitkriangkrai, C. Shen, and A. Van Den Hengel, "Strengthening the effectiveness of pedestrian detection with spatially pooled features," in *European conference on computer vision*. Springer, 2014, pp. 546–561.

[25] S. Zhang, C. Bauckhage, D. A. Klein, and A. B. Cremers, "Exploring human vision driven features for pedestrian detection," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 10, pp. 1709–1720, 2015.

[26] R. Benenson, M. Omran, J. Hosang, and B. Schiele, "Ten years of pedestrian detection, what have we learned?" in *European Conference on Computer Vision*. Springer, 2014, pp. 613–627. [Online]. Available: https://arxiv.org/pdf/1411.4304.pdf

[27] S. Zhang, R. Benenson, M. Omran, J. Hosang, and B. Schiele, "How far are we from solving pedestrian detection?" in *Proceedings of the iEEE conference on computer vision and pattern recognition*, 2016, pp. 1259–1267. [Online]. Available: https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Zhang_How_Far_Are_CVPR_2016_paper.pdf

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[29] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. LeCun, "Pedestrian detection with unsupervised multi-stage feature learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 3626–3633. [Online]. Available: http://openaccess.thecvf.com/content_cvpr_2013/papers/Sermanet_Pedestrian_Detection_with_2013_CVPR_paper.pdf

[30] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 9, pp. 1627–1645, 2009.

[31] A. Angelova, A. Krizhevsky, V. Vanhoucke, A. Ogale, and D. Ferguson, "Real-time pedestrian detection with deep network cascades," 2015.

[32] J. Hosang, M. Omran, R. Benenson, and B. Schiele, "Taking a deeper look at pedestrians," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4073–4082.

[33] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[34] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[35] A. Ess, B. Leibe, K. Schindler, and L. Van Gool, "A mobile vision system for robust multi-person tracking," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2008, pp. 1–8.

[36] C. Wojek, S. Walk, and B. Schiele, "Multi-cue onboard pedestrian detection," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, pp. 794–801.

[37] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997. [Online]. Available: http://www.face-rec.org/algorithms/Boosting-Ensemble/decision-theoretic_generalization.pdf

Anexos

A. Estado da arte

A detección de peóns é un problema amplamente investigado nos últimos 16 anos, sobre todo debido a tódalas súas posíbeis implicacións en distintos ámbitos como son a seguridade, ou a automoción. Dende os seus inicios no 2005 cando o primeiro detector por Viola e Jones empezou a funcionar[16] ata a actualidade, fixéronse grandes avances. Moitos dos últimos papers e investigacións introducen novos métodos que presentan mellores resultados. Hoxe en día pódese dicir que os métodos máis empregados e investigados clasifícanse en dous tipos[17]: ICF (Integral Channel Feature detector) e CNN(Convolutional Neural Network) para detección de peóns.

- **Detectores ICF**. Os detectores ICF foron propostos por [18][10], onde se demostrou que a orientación dos gradientes e as características de cor dunha imaxe (HOG+LUV), empregadas para mellorar árbores de decisión, eran máis efectivas e superaban facilmente os detectores previos. Para reducir os costes computacionais, algúns traballos propuxeron estudar unha aproximación ao cálculo das características mediante diferentes escalas e xanelas veciñas[19]. Para máis velocidade incluso propúxose un adestramento con varios modelos a diferentes escalas para evitar ter que reescalar a imaxe en tempo de procesado[20].

  Non obstante, a maior parte dos esforzos foron dirixidos a obter os detectores con mellor rendemento[21][22][23][24][1][25][11], onde se atopou que o que máis lle afecta é o tipo de filtros empregado. Algún exemplo dos anteriormente citados poden ser: *SquareChntrs*[26] que utiliza filtros cuadráticos promediados; *InformedHaar*[1] que básase en utilizar información da forma do corpo do peón; *LDCF*[23] que emprega autovectores que proveñen de análises linais discriminativas sobre imaxes naturais; *RotatedFilters* que é unha versión simplificada dos *LDCF*[27]; *Checkerboards*[11] que utiliza un conxunto de filtros que inclúen detectores de gradiente vertical e horizontal e patróns tipo xadrez. Pódense apreciar algúns exemplos de posíbeis filtros na Fig. 6.

  Como resumo, os detectores ICF acadan uns resultados moi competitivos cunha complexidade computacional baixa; non obstante, hay que ter en conta tamén as súas desventaxes. A creación manual de características deseñadas intuitivamente basadas en, por exemplo, a forma do corpo, limitan o funcionamento do detector facendo máis posíbel a confusión con obxectos que se parezan na súa forma. Ademais, as árbores de decisións son máis rápidas a hora de testear, pero son máis sensíbeis ao ruido nas anotacións durante o adestramento.

- **Detectores basados en CNN**. As redes neurais convolucionais (CNN) acadaron moi bos resultados tanto en clasificación coma en detección en datasets coma ImageNet[28], Pascal ou MS COCO. Non foi ata estes últimos anos que empezáronse a aplicar tamén en detección de peóns.

  A maior partes das CNN confían en outros métodos para a extracción das características, excepto no caso de *ConvNet*[29], que emprega directamente a rede sobre a imaxe mediante unha xanela deslizante. *DPM*[30] e *HOG/linSVM*[4] foron os primeiros métodos en xurdir, mentres os detectores ICF íanse facendo cada vez máis populares. Posteriormente descubriuse que os ICF poden ser utilizados como candidatos moi eficaces para mellorar a precisión das CNN[31][32].

Dataset	Captura da imaxe	Imaxe en cor	Nº de imaxes	Nº de peóns
Caltech-USA	móbil	Si	122k	155k
KITTI	móbil	Si	7518	-
INRIA	estática	Si	288	566
ETH	móbil	Si	1804	12k
TUD-Brussels	móbil	Si	508	1498
Daimler	móbil	Non	21.8k	56.5k

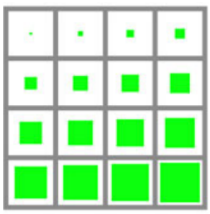
Cadro 3: Comparación en canto a volume de datos entre distintos datasets (unicamente en referencia aos sets de testeo).

Recentemente, os detectores basados en CNN están tendendo a utilizar redes profundas (deep learning) como son *AlexNet*[28], *GoogLeNet*[33] e *VGG*[34]. Estas redes son capaces de facer deteccións de alta fiabilidade sen ter que depender de métodos externos. Non obstante este aumento da calidade e fiabilidade das deteccións ten un prezo: o aumento do custo computacional e a necesidade dunha mayor cantidade de datos para o adestramento. Ademais, xeralmente este tipo de redes teñen problemas para producir deteccións ben aliñadas, especialmente para obxectos pequenos.

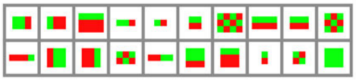
En canto aos sets de imaxes utilizados para a evaluación e adestramento dos clasificadores, xurdiron varios diferentes ao longo da última década. INRIA[4], ETH[35] e TUD-Brussels[36] foron os primeiros en aparecer e en tentar xuntar un número de imaxes significativo para facer investigacións referentes á detección de peóns. Hoxe en día Catech-USA[5] e KITTI[15] convertéronse nos máis populares debido a súa gran cantidade de información e a que son máis desafiantes que os anteriormente citados. Ambos datasets consisten en imaxes grabadas dende un coche a través dunha gran cidade, sendo para cada caso:

- Caltech-USA. Este dateset consiste en 2.5 horas de vídeos grabados a 30Hz dende un vehículo a través das rúas de Los Angeles, USA. Nos vídeos están anotados un total de 350k obxectos dos cales en torno a 2300 son peóns. O seu set de testeo consiste nun conxunto de 4024 frames. Habitualmente, para extraer as imaxes deste dataset utilízase unha de cada 30 imaxes do vídeo para evitar redundancia.
- KITTI. O dataset de KITTI foi capturado tamén a bordo dun vehículo que conducía a través da vila de Karlsruhe, Alemania, recollendo tanto zonas ruraris coma de autovía. Debido a que o set de testeo de KITTI non é público, normalmente sepárase o set de adestramento en dúas partes, empregando unha para testeo e outra para o adestramento propiamente dito.

Para a comparación, pódese ver o cadro 3, onde se amosa que Caltech-USA e KITTI sobrepasan por moito en termos de volume a outros datasets.



(a) Exemplo de filtros SquaresChntrs[26]



(b) Exemplo de filtros Chekerboards[11]

Figura 6: Comparación entre filtros de dous detectores ICF.

B. Tecnoloxías empregadas

1. **AdaBoost (*Adaptive Boosting)***[37]. É un meta-algoritmo de aprendizaxe automática, proposto por Yoav Freund e Robert Schapire, que se emprega en conxunto con outros algoritmos para mellorar os resultados finais.

   O termo *boosting* fai referencia a un tipo de algoritmos onde a finalidade é encontrar unha hipótese forte a partir de empregar hipóteses simples e febles. *Adaptive* fai referencia a que propón adestrar unha serie de clasificadores débiles de maneira iterativa, de forma que cada novo clasificador (*weak learner*) só se enfoque nos datos que foron erroneamente clasificados polo seu predecesor, conseguindo que o algoritmo final se adapte e obteña mellores resultados.

2. **CatBoost**[3]. CatBoost é un algortimo *open source* deseñado para *gradient boosting* empregando árbores de decisión. Foi creado por investigadores de Yandex e é empregada por outras empresas como Cloudflare ou o CERN. Algunhas das súas principais ventaxes son a súa mellora, respecto a outros algoritmos similares, na predicción de clases e o soporte para adestramento multi-GPU.

3. **Dataset Caltech-USA**[5]. Este dateset consiste en 2.5 horas de vídeos grabados a 30Hz dende un vehículo a través das rúas de Los Angeles, USA. Nos vídeos están anotados un total de 350k obxectos dos cales en torno a 2300 son peóns. O seu set de testeo consiste nun conxunto de 4024 frames. Habitualmente, para extraer as imaxes deste dataset utilízanse unha de cada 30 imaxes do vídeo para evitar redundancia.

4. **Dataset INRIA**[4]. INRIA é un dos datasets de peóns máis antigos e comparativamente cós actuais, é un dos máis reducidos. Consiste nunha serie de imaxes nas que aparecen peóns en distintos contextos, non só en cidade se non tamén en parques, nunha montaña, na praia, etc. Xeralemente, antes de que aparacesen os datasets máis recentes empregábase para o adestramento dos clasificadores polas súas anotacións das posicións dos peóns, que eran de gran calidade.

5. **OpenCV**[9]. OpenCV é unha libraría *open source* centrada na visión artificial orixinalmente desenvolta por Intel. Actualmente (outubro de 2019) encóntrase na súa versión 4.1.2 e abarca máis de 2500 algoritmos de procesado de imaxe optimizados para o seu uso baixo unha licenza BSD. É unha libraría popular e de prestixio no seu campo (é empregada por Google, Intel, IBM, Honda...) e está dispoñíbel para tódalas plataformas (Windows, Linux, Android e Mac OS) e ten interfaces dispoñíbeis en Python, Java, C++ e MATLAB.

6. **Scikit-learn**[2]. É unha libraría de *machine learning* creada en 2007 para a lenguaxe de programación Python. Inclúe algoritmos de clasificiación, regresión e análise de grupos entre os cales están: *Random Forest, AdaBoost, K-means e SVM*. Esta deseñada para traballar con outras librarías de Python como NumPy e SciPy. Está maiormente escrita en Python, aínda que algunhas partes escribíronse en Cython para mellorar o rendemento.

C. Deteccións en imaxes

Figura 7: Deteccións obtidas empregando o clasificador Catboost sobre algunhas imaxes do set de testeo de INRIA. Recadrado en verde aparece aquelo que o clasificador detecta coma peón.