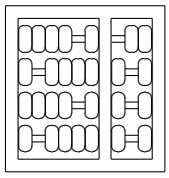


INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

Um Estudo sobre Aproximação de Altura de Pessoas em Vídeos de Segurança

Fábio Sartorato *Helio Pedrini*

Relatório Técnico - IC-PFG-16-06 - Projeto Final de Graduação

December - 2016 - Dezembro

The contents of this report are the sole responsibility of the authors.
O conteúdo do presente relatório é de única responsabilidade dos autores.

Um Estudo sobre Aproximação de Altura de Pessoas em Vídeos de Segurança

Fábio Sartorato* Helio Pedrini†

Dezembro de 2016

Resumo

Estimar a altura de pessoas é uma atividade importante na área de vigilância e segurança, facilitando a identificação de um indivíduo em aplicações tais como biometria e ciência forense. As imagens geradas pelos sistemas de vigilância normalmente apresentam baixa qualidade para reconhecimento facial, de modo que outras características podem ser utilizadas para reconhecer indíviduos. Este trabalho investiga o problema de aproximação da altura de pessoas em vídeos capturados por câmeras de segurança. Diversos métodos da literatura são analisados e uma abordagem baseada em regressão não-linear para calibração da câmera é selecionada para implementação devido aos resultados satisfatórios apresentados.

1 Introdução

Câmeras de vigilância e segurança têm desempenhado um papel importante em tarefas de identificação de pessoas suspeitas de cometerem crimes. Entretanto, por limitações inerentes aos próprios equipamentos, os vídeos que estas câmeras oferecem são, muitas vezes, de baixa qualidade, impossibilitando o reconhecimento facial de indivíduos. Dessa maneira, torna-se necessário buscar outras abordagens para identificação que não requerem imagens de alta definição como, por exemplo, a altura de um indivíduo presente no vídeo.

Muitos autores, tais como Creminisi et al. [1], propõem a utilização de pontos de fuga para calibrar a câmera de segurança. Outra estratégia muito diferente é proposta por Li et al. [2], a qual utiliza uma regressão não-linear para definir alguns parâmetros associados diretamente à câmera de segurança, calibrando-a e, assim, podendo aproximar alturas de elementos do vídeo.

Este trabalho estuda o problema de aproximação da altura de pessoas em sequências de vídeos adquiridas por câmeras de segurança. Algumas técnicas da literatura são analisadas e um método baseado em regressão não-linear para calibração da câmera é investigada mais detalhadamente e selecionada para implementação.

*Instituto de Computação, Universidade Estadual de Campinas, 13083-852, Campinas, SP
†Instituto de Computação, Universidade Estadual de Campinas, 13083-852, Campinas, SP

Este relatório está organizado da seguinte maneira. Na seção 2, duas abordagens da literatura são analisadas. Na seção 3, detalhes da implementação fornecida pelos autores do trabalho de Li et al. [2] são apresentados, bem como uma implementação simples para identificar pessoas caminhando em uma cena de vídeo realizada em linguagem Python e pacote OpenCV. Finalmente, comentários finais sobre possíveis refinamentos do método são discutidos na seção 4.

2 Trabalhos Relacionados

Esta seção descreve os principais aspectos de dois métodos da literatura para estimação de alturas em imagens.

2.1 Método de Li et al. [2]

Hartley e Zisserman [3] propuseram o modelo de câmera, expresso como:

$$\mathbf{P} = \mathbf{K} \cdot \mathbf{R} \cdot [\mathbf{I} \,|\, \mathbf{t}] \tag{1}$$

em que $\mathbf{R}$ é a matrix de rotação da câmera $\mathbf{P}$ e $\mathbf{t}$ o vetor de translação.

Li et al. [2] buscaram simplificar esse modelo assumindo uma câmera fixa, que não translada nos eixos X e Z (ou seja, pode ter altura variável) e sobre rotação apenas no eixo X. Dessa maneira, a equação 1 torna-se:

$$\mathbf{P} = \mathbf{K} \cdot \mathbf{R_x} \cdot [\mathbf{I} \,|\, \mathbf{c_y}] \tag{2}$$

em que $\mathbf{R_x}$ é a matriz de rotação no eixo X, que representa a inclinação da câmera, enquanto $\mathbf{c_y}$ o vetor de translação pelo eixo Y.

A partir de apenas três parâmetros, podemos mapear uma coordenada do mundo real para a da imagem em questão, a partir da definição de $\mathbf{P}$:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathbf{P} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3}$$

O parâmetro P deve ser expandido para se definir totalmente esse mapeamento. Da equação 2 e, assumindo a câmera como nosso ponto inicial, tem-se:

$$\begin{bmatrix} 0 & 0 \end{bmatrix}^T \tag{4}$$

tal que:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & f \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\operatorname{sen}\theta \\ 0 & \operatorname{sen}\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -c \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{5}$$

As variáveis f, θ e c são os três parâmetros usados para calibrar a câmera, sendo eles a distância focal, a inclinação da câmera e a altura da câmera, respectivamente.

Resolvendo essa operação matricial, temos:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} fX \\ f\cos\theta \cdot Y - f\,\mathrm{sen}\,\theta \cdot Z - fc\cos\theta \\ \mathrm{sen}\,\theta \cdot Y + \cos\theta \cdot Z - c\,\mathrm{sen}\,\theta \end{bmatrix} \tag{6}$$

Como uma pessoa caminhando em um cena de vídeo está disposta em orientação vertical com relação ao chão, o valor associado à coordenada x do plano real não é tão relevante quanto a coordenada y. Dessa maneira, a partir da equação 5 e considerando $\cos\theta \neq 0$, podemos representar y em coordenadas cartesianas da seguinte forma:

$$y = \frac{fY - f\tan\theta \cdot Z - fc}{\tan\theta \cdot Y + Z - c\tan\theta} \tag{7}$$

Contudo, temos dois tipos de alturas em questão. O processamento da imagem se dá encontrando dois pontos, a coordenada do pé à cabeça no eixo Y e no eixo Z. Logo, podemos dividir a equação 6 em duas, uma referente à variável associada ao pé (y_f) e outra referente à variável associada à cabeça (y_h). Entretanto, para medir as coordenadas do par pé-cabeça de Z é necessário conhecer maiores detalhes na imagem. Para simplificar, podemos eliminar qualquer variável associada a Z e analisar apenas a equação em função de coordenadas do eixo Y.

$$\mathrm{y_h} = \frac{f\left(-c\tan^2\theta + \mathrm{Y_h} - c\right)\mathrm{y_f} + f^2 tg\theta \cdot \mathrm{Y_h}}{\tan\theta \cdot \mathrm{Y_h y_f} + f\left(\tan^2\theta \cdot \mathrm{Y_h} - c\,tg^2\theta - c\right)} + \epsilon \tag{8}$$

em que ϵ é o erro produzido pelos parâmetros de calibração.

Espera-se que a equação 7 dê uma aproximação para y_h, definida pela função de aproximação $\hat{y}_h$, que possui dois argumentos: y_f e Y_h:

$$\hat{\mathrm{y}}_\mathrm{h}\left(\mathrm{y_f}, \mathrm{Y_h}\right) = \frac{f\left(-c\tan^2\theta + \mathrm{Y_h} - c\right)\mathrm{y_f} + f^2\tan\theta \cdot \mathrm{Y_h}}{\tan\theta \cdot \mathrm{Y_h y_f} + f\left(\tan^2\theta \cdot \mathrm{Y_h} - c\tan^2\theta - c\right)} \tag{9}$$

Como a equação 8 representa uma forma não-linear, utiliza-se uma regressão não-linear para encontrar os melhores valores para os parâmetros.

$$\begin{bmatrix} \hat{f} \\ \hat{\theta} \\ \hat{c} \end{bmatrix} = \underset{f,\theta,c}{\arg\min} \sum_{i=1}^{N} \left(\hat{\mathrm{y}}_{\mathrm{h}i} - \mathrm{y}_{\mathrm{h}i}\right)^2. \tag{10}$$

Os parâmetros θ_0 e c_0 podem ser aproximados visualmente, enquanto o parâmetro f_0 é definido de 0,5 a 1,5 vezes a altura da imagem (em centímetros), como dito por Li et al. [2].

Finalmente, a altura de uma pessoa pode ser estimada por um par de pontos pé-cabeça diretamente da imagem, sendo essa altura definida por:

$$\hat{\mathrm{Y}}\left(\mathrm{y_f}, \mathrm{y_h}\right) = \frac{-\hat{f}\hat{c}\left(tg^2\hat{\theta} + 1\right) \cdot \left(\mathrm{y_f} - \mathrm{y_h}\right)}{tg\hat{\theta} \cdot \mathrm{y_f y_h} - \hat{f}\mathrm{y_f} + \hat{f}tg^2\hat{\theta}\mathrm{y_h} - \hat{f}^2 tg\hat{\theta}}. \tag{11}$$

Utilizando os parâmetros encontrados pela regressão não-linear.

2.2 Método de Criminisi et al. [1]

Enquanto o método de Li et al. [2] empregam regressões não-lineares para estimar a altura de uma pessoa em vídeo, Criminisi et al. [1] usam basicamente comparações geométricas entre a imagem e um plano de referência.

Criminisi et al. [1] definem que a altura pode ser estimada em dois passos: (i) calibração mínima; (ii) medição da altura. Em (i), são definidos os pontos de fuga e as linhas de fuga da imagem, bem como os referenciais. Em (ii), a altura é computada utilizando dois pontos da imagem.

Apesar de ser matematicamente menos complexo, o maior problema com o método de Criminisi et al. [1] está na calibração da câmera, pois para definir os pontos de fuga, bem como as linhas verticais de referência da imagem, é necessário que o vídeo tenha uma qualidade relativamente boa, já que, como mencionado por Li et al. [2], a definição desses pontos é muito suscetível a ruído. Em câmeras de vigilância, limitações dos equipamentos normalmente contribuem para o aumento de ruídos nos quadros do vídeo, tornando a calibração deste método em questão não tão trivial.

3 Implementação

Junto com o artigo de Li et al. [2], foi disponibilizado um código em MATLAB que implementa o algoritmo proposto nesse artigo. Esse código possui licença *GNU General Public License version 2*, podendo então ser alterado e distribuído livremente. A compreensão do método ficou mais simples a partir do estudo desse código.

3.1 Calibração em MATLAB

A partir do arquivo *CalibrationDemo.m*, podemos explorar o funcionamento do código.

Implementação 1: CalibrationDemo.m (Extraído de Li et al. [2].)

```
clear all

width = 1280;
height = 720;

ftc0 = [1,-30,-3];

filename = 'CC-001-01-20131107-01.txt';

% k = ku1 ku2 kd1 kd2
k = [0.6322 0.5731 -0.6 0.4]; % cam01 cam02
% k = [0 0 0 0] % cam03
% k = [0.1890 0.1996 -0.2 0]; % cam05

% most precise, k is needed
ftc = CalibrateFromFileXYd(filename, width, height,ftc0, k)

```

```
18 % suitble for many cases
19 ftc = CalibrateFromFileXY(filename, width, height,ftc0)
20
21 % use only Y coordinates, only for height estimation
22 ftc = CalibrateFromFileY(filename, width, height,ftc0)
```

Inicialmente, as dimensões da imagem são definidas, bem como a distância focal, inclinação e altura da câmera inicial, como descrito no artigo. No arquivo *CC-001-01-20131107-01.txt* estão definidos os dados encontrados após a identificação das pessoas caminhando.

Em seguida, temos três definições diferentes para o cálculo dos parâmetros de calibração, cada uma delas mais robusta do que a próxima. A robustez está no algoritmo usado para resolver a regressão não-linear. O pacote MATLAB já possui algumas implementações disponíveis. Neste trabalho, vamos focar na mais simples, que utiliza apenas as coordenadas no eixo Y (linha 22).

Implementação 2: CalibreateFromFileY.m (Extraído de Li et al. [2].)

```
1
2  function ftc = CalibrateFromFileY(filename,imageWidth,imageHeight, ftc0)
3
4  options = optimset('Algorithm','trust-region-reflective',...
5                     'MaxFunEvals', 1e5, 'MaxIter', 1e4);
6
7  rawPoints = dlmread(filename,'\t',1);
8
9  yf = 0.5 - rawPoints(:,5)/imageHeight;
10 yh = 0.5 - rawPoints(:,3)/imageHeight;
11
12 f2h = @(x,xdata)FootToHeadY(x,xdata,rawPoints(1,6)/100.);
13
14 [ftc,r,J,cov,mse] = nlinfit(yf,yh,f2h,ftc0);
```

No pré-processamento para a regressão não-linear são atribuídos os valores encontrados dos pontos de pé e cabeça referentes ao eixo Y. Neste módulo. utiliza-se o método mais simples, já implementado em MATLAB, para resolver uma regressão não-linear (*nlinfit(X,Y,modelfun,beta0,options)*).

Da documentação do MATLAB, *beta = nlinfit(X,Y,modelfun,beta0)* retorna um vetor de coeficientes estimados para uma regressão não-linear das respostas em Y nos preditores em X utilizando o modelo especificado por *modelfun*. Os coeficientes são estimados por meio do método de mínimos quadrados iterativo com valores iniciais especificados por beta0.

Assim, define-se uma função anônima *f2h* que representa o modelo a ser usado na regressão. Analisando esse modelo *FootToHeadY*, verificamos que ele implementa a equação 8.

Implementação 3: FootToHeadY.m (Extraído de Li et al. [2].)

```
1 function yh = FootToHeadY(x,xdata,h)
2
```

```
f=x(1);
theta=x(2);
c=x(3);
yf=xdata;
tt = tand(theta);

yh=(f*(tt^2*c+h+c).*yf+f^2*tt*h)./ (tt*h.*yf+f*(tt^2*h+tt^2*c+c));
```

Os parâmetros a serem encontrados estão contidos na variável x, sobre os dados de *xdata*.

Após a calibração, basta chamar a função *PointsToHeightY*, que implementa a equação 10 utilizando os parâmetros encontrados na regressão e os dados de par pé-cabeça no eixo Y, colhidos anteriormente.

Implementação 4: PointsToHeightY.m (Extraído de Li et al. [2].)

```
function h = PointsToHeightY(x,xdata)

f=x(1);
theta=x(2);
c=x(3);

yf=xdata(:,1);
yh=xdata(:,2);

tt=tand(theta);

h = (f*tt^2*c+f*c).*(yf-yh)./(tt*yf.*yh+f*tt^2*yh-f*yf-f^2*tt);
```

3.2 Detecção na Linguagem Python e no Pacote OpenCV

Apesar do método analisado calibrar com eficiência uma câmera, ainda é necessário desenvolver um método de identificação de pessoas caminhando para assim gerar os arquivos de dados de entrada para a regressão não-linear e, futuramente, para a aproximação da altura. Desse modo, é necessário um meio eficiente de detecção de pessoas caminhando para, durante um vídeo, definir os pontos de pé-cabeça para calibrar a câmera.

Rosebrock [4, 5, 6] desenvolveu ferramentas de busca por imagens e, atualmente, mantém um *blog* com dicas e tutoriais relacionados à visão computacional, aprendizado de máquina e processamento de imagens. Nesse blog, pode-se encontrar dois tutoriais interessantes para este projeto: *Basic Motion Detection and Tracking with Python and OpenCV* e *Pedestrian Detection OpenCV*.

No primeiro, Rosebrock ensina a implementar um método simples e leve para identificar objetos se movendo num vídeo. Ao iniciar o programa, o primeiro quadro de vídeo é tido como o quadro base. Então, inicia-se a iteração sobre todos os quadros seguintes. A cada iteração, calcula-se a diferença entre o quadro base e o quadro atual para, assim, verificar as alterações ocorridas. Se uma área grande sofreu mudanças, então ela pode ser considerada

um objeto em movimento e um retângulo é circunscrito nessa área.

Implementação 5: Basic motion detection (Extraído de Rosebrock [4, 5, 6]).

```
# resize the frame, convert it to grayscale, and blur it
frame = imutils.resize(frame, width=500)
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
gray = cv2.GaussianBlur(gray, (21, 21), 0)

# if the first frame is None, initialize it
if firstFrame is None:
firstFrame = gray
continue

# compute the absolute difference between the current frame and
# first frame
frameDelta = cv2.absdiff(firstFrame, gray)
thresh = cv2.threshold(frameDelta, 25, 255, cv2.THRESH_BINARY)[1]

# dilate the thresholded image to fill in holes, then find contours
# on thresholded image
thresh = cv2.dilate(thresh, None, iterations=2)
(cnts, _) = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

# loop over the contours
for c in cnts:
# if the contour is too small, ignore it
if cv2.contourArea(c) < args["min_area"]:
continue

# compute the bounding box for the contour, draw it on the frame,
# and update the text
(x, y, w, h) = cv2.boundingRect(c)
#print h
cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 1)
text = "Occupied"
```

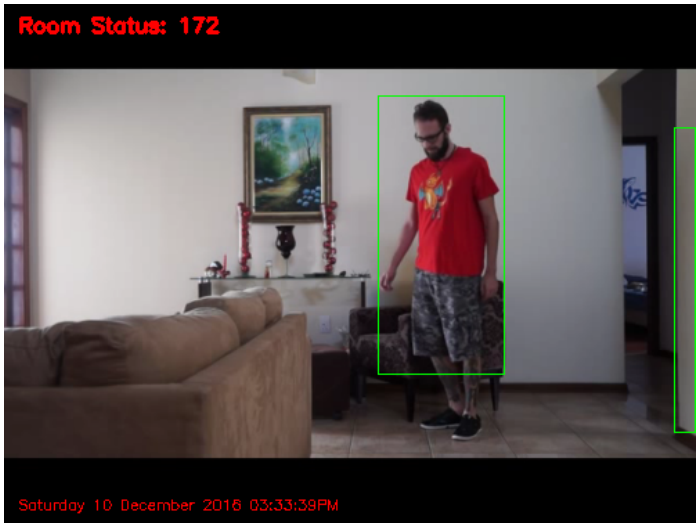
Entretanto, esta implementação se mostrou muito falha, pois é muito dependente do primeiro quadro e muito sensível à luz e sombras, como é possível ver na Figura 1.

Note como, no momento deste quadro (quadro 120), outro retângulo foi desenhado, indicando que existem objetos em movimento, contudo, claramente podemos ver que alguns deles estão parados durante o vídeo inteiro (nesse caso, uma parede). Desse modo, uma análise foi necessária para se encontrar outra maneira de solucionar o problema da identificação de pessoas caminhando.

No segundo tutorial de Rosebroc, o autor descreve um método para identificação de pedestres baseada no modelo pré-treinado do descritor Histograma de Gradientes Orientados (*Histogram of Oriented Gradients*) (HOG) [7] colmbinado com o classificador SVM linear.

(a) Diferença entre quadro atual e quadro inicial binarizado.

(b) Diferença absoluta entre quadro atual e quadro inicial.



(c) Detecção de movimento no instante do quadro atual.

Figura 1: Resultados do detector de movimentos.

Implementação 6: HOG Initializer (Adaptado de Rosebrock [4, 5, 6]).

```
# initialize the HOG descriptor/person detector
hog = cv2.HOGDescriptor()
hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())
```

A inicialização do descritor se dá em apenas duas linhas, como pode ser visto na Implementação 6. Em seguida se inicia um laço por todos os quadros do vídeo. Esse laço está descrito na Implementação 7.

Implementação 7: HOG in action (Adaptado de Rosebrock [4, 5, 6]).

```
# detect people in the image
(rects, weights) = hog.detectMultiScale(image, winStride=(4, 4),padding=(0,
    0), scale=1.05)

# draw the original bounding boxes

for (x, y, w, h) in rects:
   cv2.rectangle(orig, (x, y), (x + w, y + h), (0, 0, 255), 2)

# apply non-maxima suppression to the bounding boxes using a
# fairly large overlap threshold to try to maintain overlapping
# boxes that are still people
rects = np.array([[x, y, x + w, y + h] for (x, y, w, h) in rects])
pick = non_max_suppression(rects, probs=None, overlapThresh=0.65)

# draw the final bounding boxes
for (xA, yA, xB, yB) in pick:
   cv2.rectangle(image, (xA, yA), (xB, yB), (0, 255, 0), 2)
```

Na linha 2, a identificação das pessoas é realizada. O quão bem esse descritor funcionará depende de como os parâmetros do módulo *detectMultiScale()* são atribuídos. Rosebroc apresentou outro tutorial que explica cada um desses parâmetros e, a partir dessa descrição, foi possível encontrar bons resultados para a identificação de pessoas em movimento, os quais podem ser vistos como exemplo na Figura 2.

Na linha 13, um método de supressão não-máxima (*non-maximum suppression*) ajuda a identificar se dois retângulos que se sobrepõem pertencem ao mesmo objeto. Na Figura 2, não houve retângulos sobrepostos, então a imagem da Figura 2(a) e (b) tem o mesmo valor quanto à identificação de pessoas.

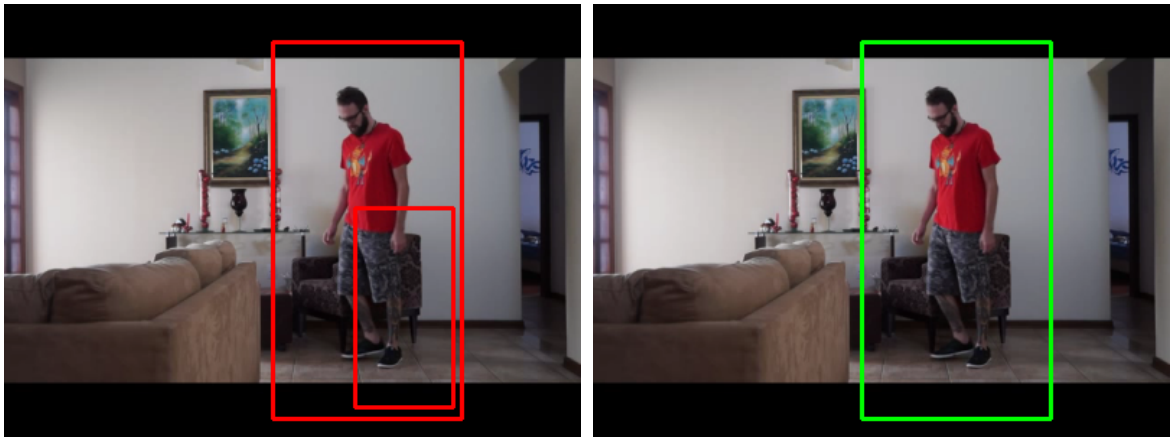
Entretanto, na Figura 3, podemos ver um exemplo de como é importante utilizar esse último filtro, encontrado no quadro 176.

A ideia é utilizar esse descritor para, em determinados quadros, marcar as coordenadas de onde seriam os pés e a cabeça da pessoa, para assim termos os dados de entrada para a calibração e para o cálculo da altura. Contudo, esse método se apresentou com um grande erro associado, pois, por melhor que seja a identificação de pessoas, o retângulo associado a cada pessoa possui uma borda grande o suficiente para ser considerada, provavelmente por conta dos efeitos de sombra e reflexo nas imagens do vídeo. Note como no exemplo da

(a) Detecção de pessoas antes da supressão não-máxima.

(b) Detecção de pessoas após supressão não-máxima.

Figura 2: Identificador de pessoas.



(a) Detecção de pessoas antes da supressão não máxima.

(b) Detecção de pessoas após supressão não máxima.

Figura 3: Funcionamento da supressão não máxima.

Figura 2 a reta de baixo do retângulo está consideravelmente longe do pé do indivíduo.

4 Comentários Finais

Apesar do código disponibilizado por Li et al. [2] ter licença *GNU General Public License version 2*, ele está implementado no pacote MATLAB, cuja licença é proprietária. Um aspecto de interesse deste projeto foi portar o código para uma linguagem a qual fosse mais fácil o acesso ao executável.

O principal ponto do código portado foi a regressão não-linear para a etapa de calibração. Dessa maneira, buscou-se uma linguagem que proporcionasse meios de resolver uma regressão desse tipo. As bibliotecas Scipy e Numpy da linguagem Python fornecem recursos para realizar uma regressão não-linear pelo método de mínimos quadrados, utilizando o método *scipy.optimize.curve_fit*, exatamente como descrito no trabalho de Li et al. [2].

Da documentação da Scipy, temos:

```
scipy.optimize.curve_fit
...
Use non-linear least squares to fit a function, f, to data.
Assumes ydata = f(xdata, *params) + eps
```

Experimentalmente, obtivemos os mesmos resultados para o mesmo conjunto de dados quando executados na implementação em MATLAB, são eles:

ftc = [0.76069236 -38.60776131 -2.70211307]

Implementação 8: Non-linear regression in Python (Adaptado de Rosebrock [4, 5, 6]).

```
f = 1.0       #Initial values for focal length,
theta = -30.0 #tilt angle and camera height.
c = -3.0

# Definition of the function to be fitted
f2h = lambda f, theta, c, xdata:FootToHeadY(f, theta, c, xdata)

# Non-linear regression.
ftc, popt = curve_fit(f2h, yf,yh,p0=(f,-theta,c))

print ftc
```

Implementação 9: Definition of the function to be fitted (Adaptado de Rosebrock [4, 5, 6]).

```
def FootToHeadY(xdata, f, theta, c):

   h = 1.7
   yf = xdata
   tt = math.tan(math.radians(theta))

```

Tabela 1: Dados de teste.

Quadro	Head.X	Head.Y	Feet.X	Feet.Y	Height
3214	658	3	664	114	170
3220	647	6	651	126	170
3227	629	15	637	142	170
3235	606	23	617	161	170
3241	584	35	601	178	170
3249	554	44	568	201	170
3255	520	60	543	228	170
3263	481	75	499	255	170
3270	437	94	468	288	170
3278	385	122	417	334	170
3284	334	148	384	373	170
3292	257	186	309	435	170
3299	176	237	258	500	170
3307	77	312	177	582	170

```
7     yh = ( f * ((tt**2) * c+h+c) * yf+(f**2) * tt*h) /
8          (tt*h * yf+f * ((tt**2) * h+(tt**2) *c+c))
9
10    return yh
```

Pode se observar que, na Implementação 9, as linhas 7 e 8 representam a Equação 8 descrita na Seção 2.

A tabela 1 contém os dados de entrada para testes referentes ao quadro final do vídeo disponível no *Github* do projeto de Li et al. [2]. O arquivo apresenta todos os pares de pontos pé-cabeça, tanto no eixo X quanto no eixo Y. Entretanto, para essa regressão, utilizamos apenas as colunas 3 e 5, bem como a estimativa da altura da pessoa usada na calibração (coluna 7).

Quanto à eficiência, ambos os códigos são eficientes com relação à calibração. Utilizando-se o método *tic* e *toc* do MATLAB, obtivemos um tempo de execução de 0.094154 segundos, enquanto que, utilizando a biblioteca *time* de Python, obtivemos um tempo de execução menor, de 0.0030000 segundos. O refinamento deve ser feito agora no módulo que realiza a identificação de pessoas e calcula os pares de pontos pé-cabeça.

Portanto, podemos concluir que o método descrito por Li et al. [2] é muito eficiente para calibrar uma câmera, mas ainda é necessário investigar outras estratégias para o cálculo dos pares de pontos pé-cabeça.

Referências

[1] Criminisi, A., Zisserman, A., Van Gool, L.J., Bramble, S.K., Compton, D.: New Approach to Obtain Height Measurements from Video. In: Enabling Technologies for

Law Enforcement and Security, International Society for Optics and Photonics (1999) 227–238

[2] Li, S., Nguyen, V.H., Ma, M., Jin, C.B., Do, T.D., Kim, H.: A Simplified Nonlinear Regression Method for Human Height Estimation in Video Surveillance. EURASIP Journal on Image and Video Processing **2015**(1) (2015) 1–9

[3] Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press (2003)

[4] Rosebroc, A.: Basic motion detection and tracking with Python and OpenCV (Acesso em 30 de novembro de 2016) `http://www.pyimagesearch.com/2015/05/25/basic-motion-detection-and-tracking-with-python-and-opencv`.

[5] Rosebroc, A.: Pedestrian Detection OpenCV (Acesso em 30 de novembro de 2016) `http://www.pyimagesearch.com/2015/11/09/pedestrian-detection-opencv`.

[6] Rosebroc, A.: Pedestrian Detection OpenCV (Acesso em 30 de novembro de 2016) `http://www.pyimagesearch.com/2015/11/16/hog-detectmultiscale-parameters-explained/`.

[7] Dalal, N., Triggs, B.: Histograms of Oriented Gradients for Human Detection. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Volume 1., IEEE (2005) 886–893