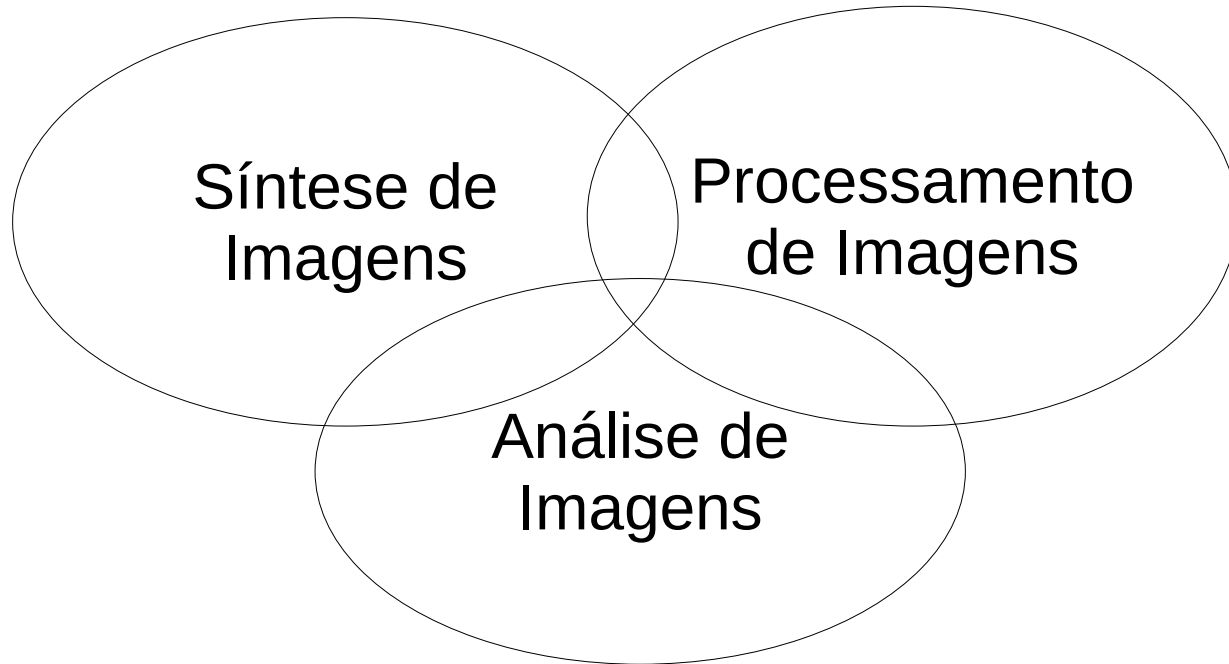


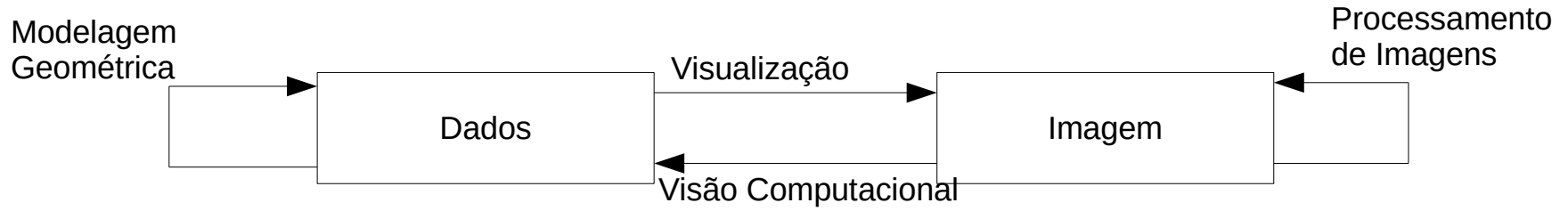
Computação Gráfica

Introdução

- 3 grandes áreas



Introdução

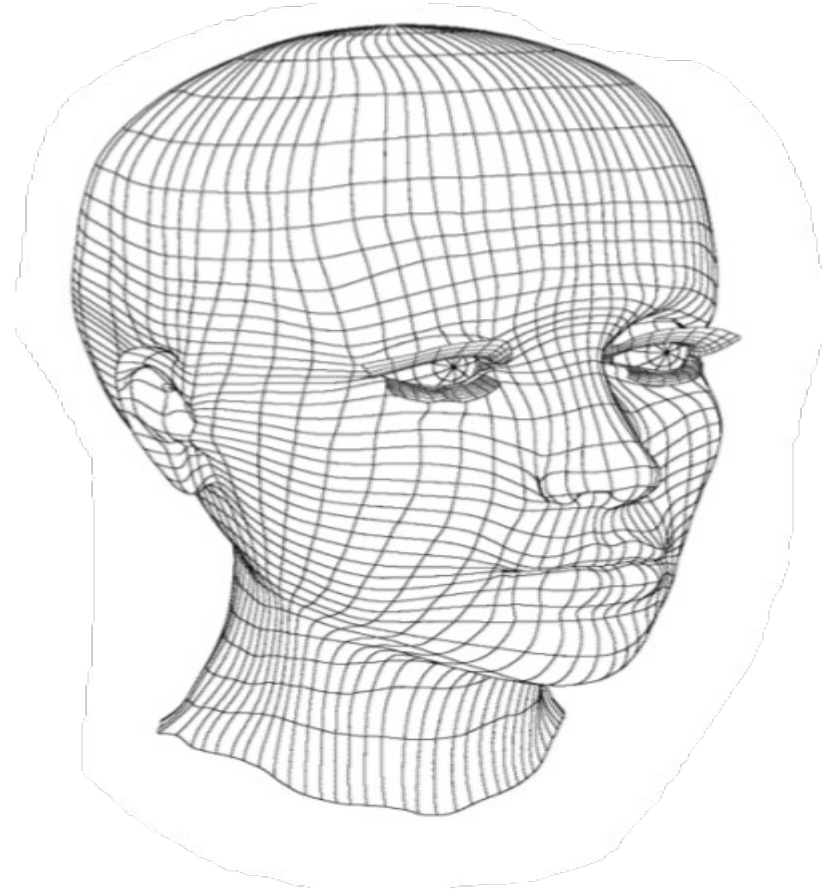
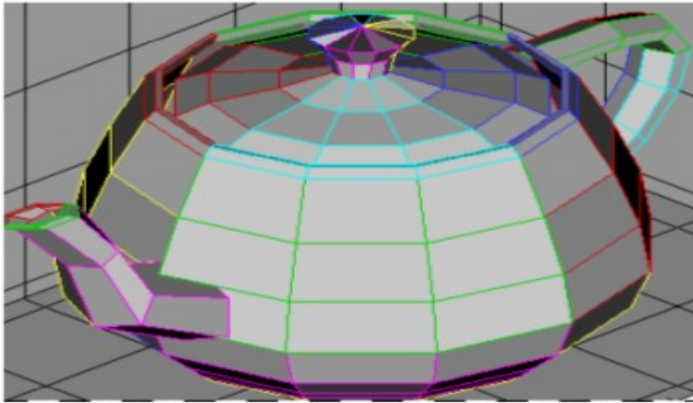


Modelagem Geométrica

- Representação da realidade
 - Geometria
 - Fotometria
- Como?
 - Interativo
 - Algorítmico
 - Scanning

Modelagem Geométrica

- Interativa

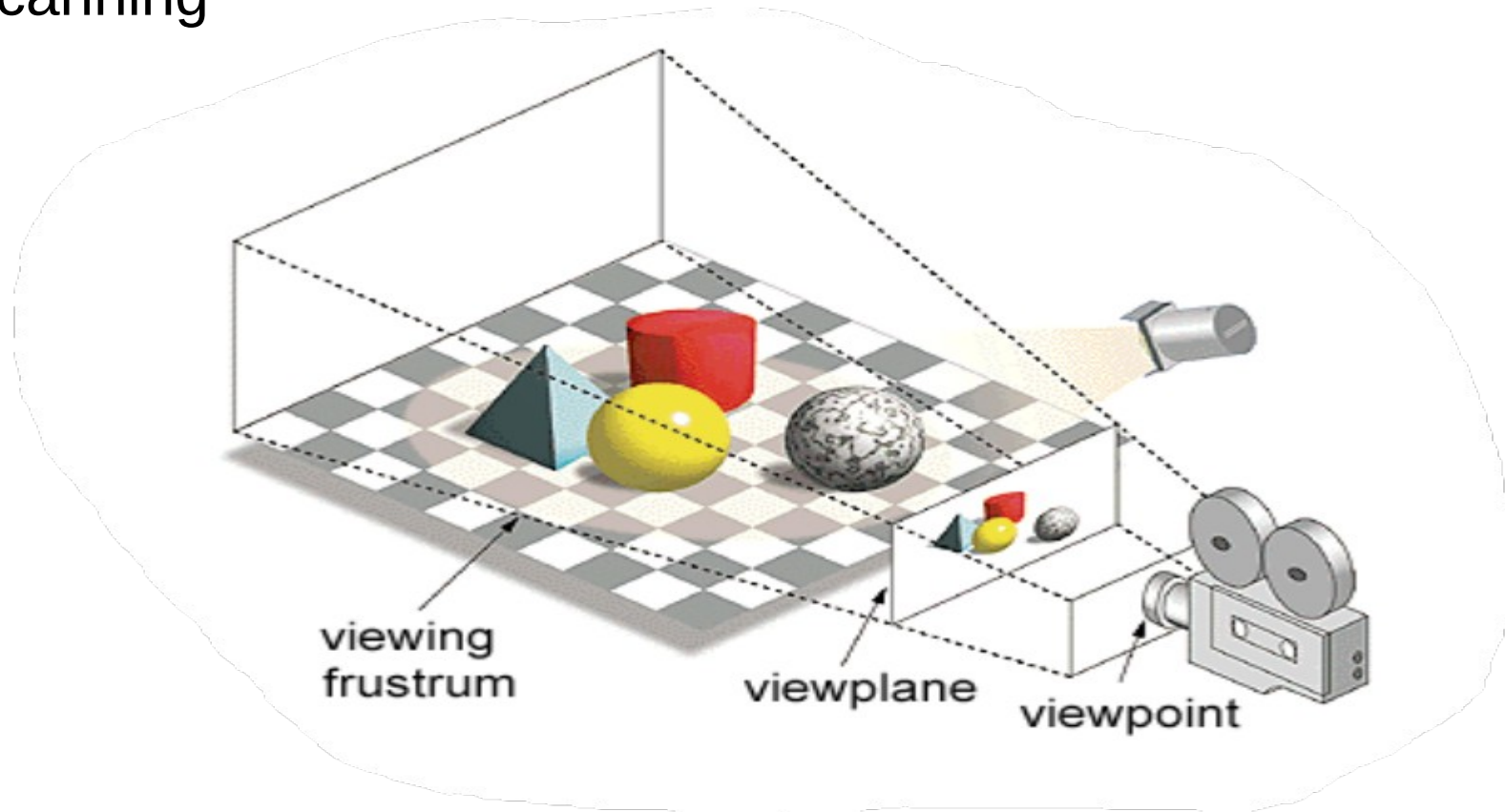


Modelagem Geométrica

- 3DS MAX
 - <http://www.autodesk.com/products/3ds-max/overview>
- SketchUp
 - <http://www.sketchup.com/>
- Blender
 - <https://www.blender.org/>
- POV-Ray
 - <http://www.povray.org/>

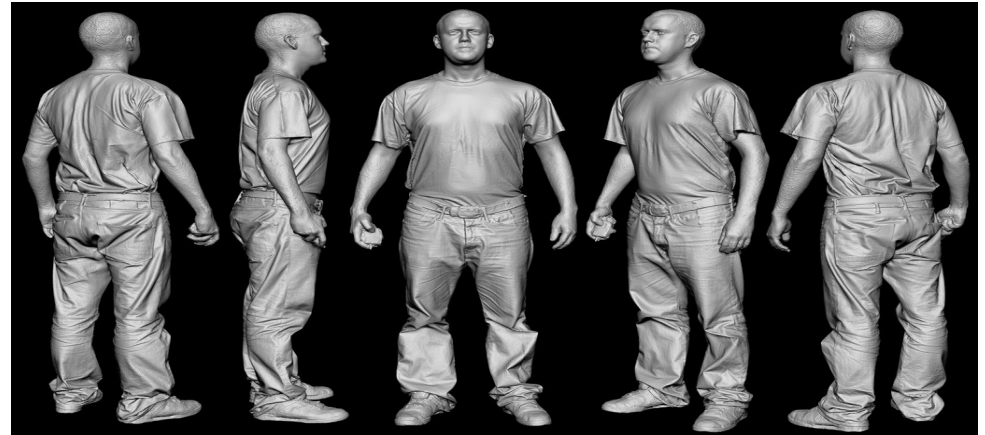
Modelagem Geométrica

- Scanning



Modelagem Geométrica

- Scanning



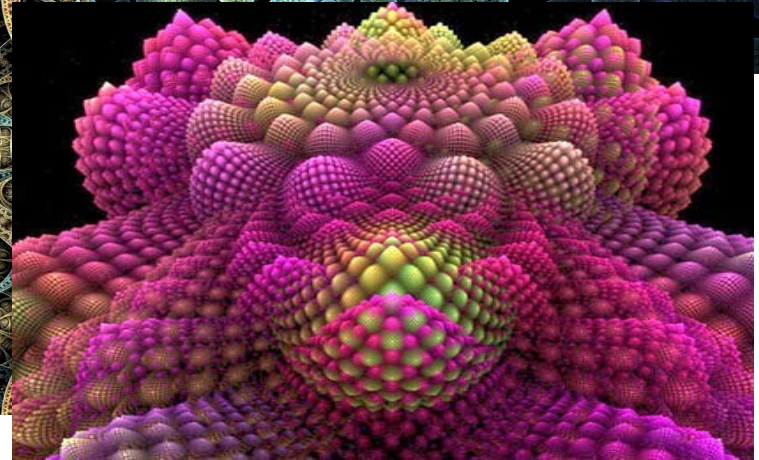
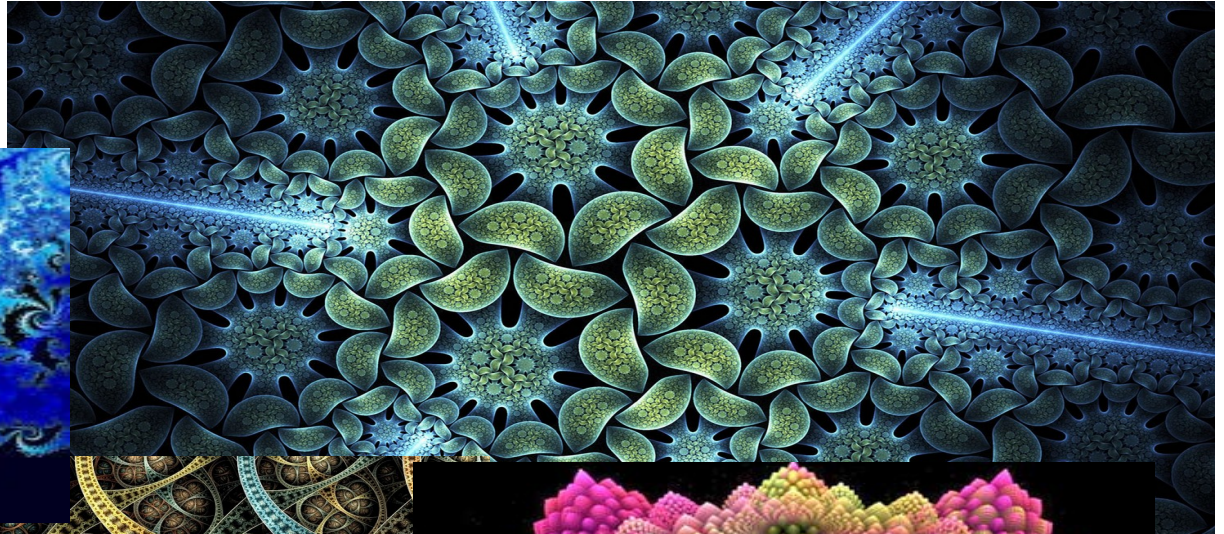
Modelagem Geométrica

- 480 câmeras
- 2 bilhões de polígonos
- 7000 imagens coloridas
- 32 Gigabytes
- 30 noites de escaneamento
- 22 pessoas envolvidas



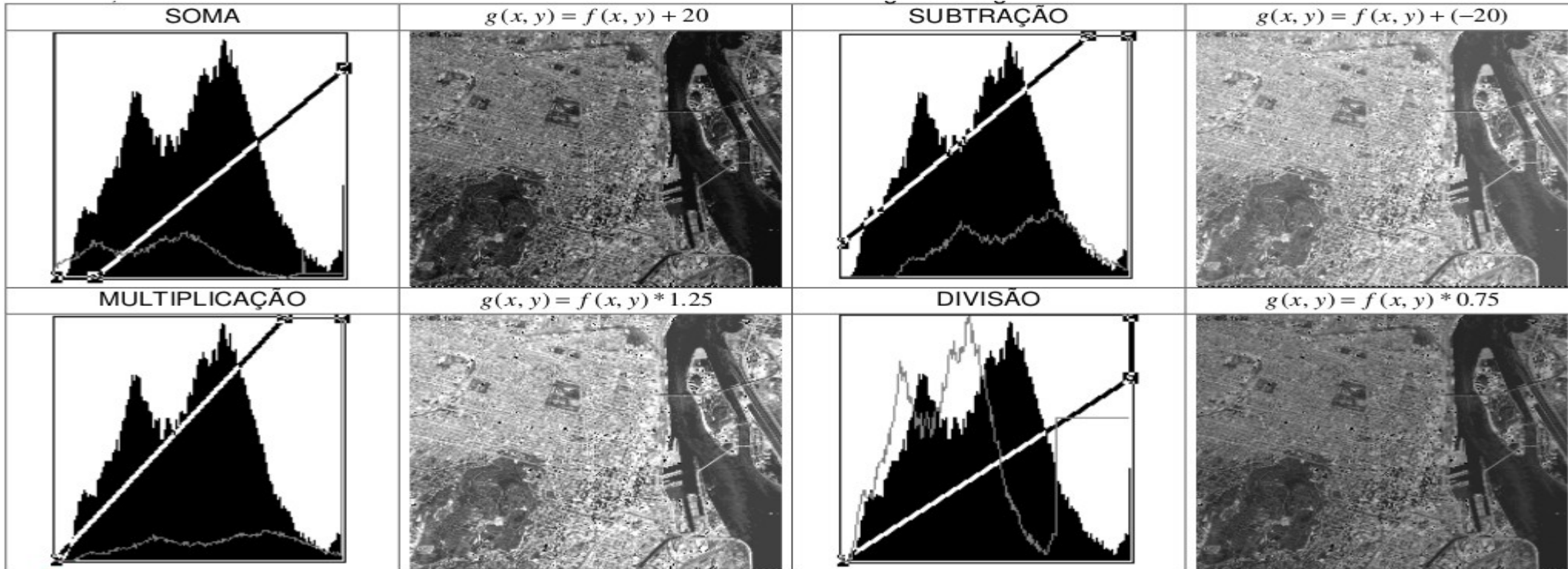
Modelagem Geométrica

- Algorítmico



Processamento de Imagens

- Ponto a ponto



Processamento de Imagens

- Filtros

Mediana: filtro útil para remoção de ruído sem alterar muito a definição das bordas



Variância: operador focal útil para caracterizar de maneira simples a textura.



Moda: filtro útil para remover pixels isolados numa classificação.



Realce local: filtro de realce local da textura; esse filtro reduz as frequências baixas.



Passa banda: filtro que deixa passar apenas algumas frequências; de 3 a 5 pixels neste caso.



Máximo: operador focal usado em conjunção com o mínimo para certos métodos de extração de informação.



Análise de Imagens

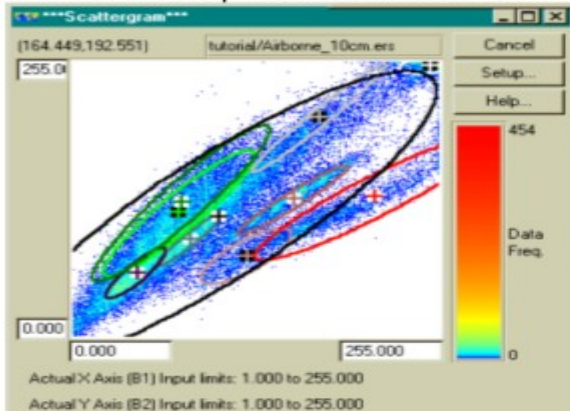
1. Adquirir uma intimidade com a imagem



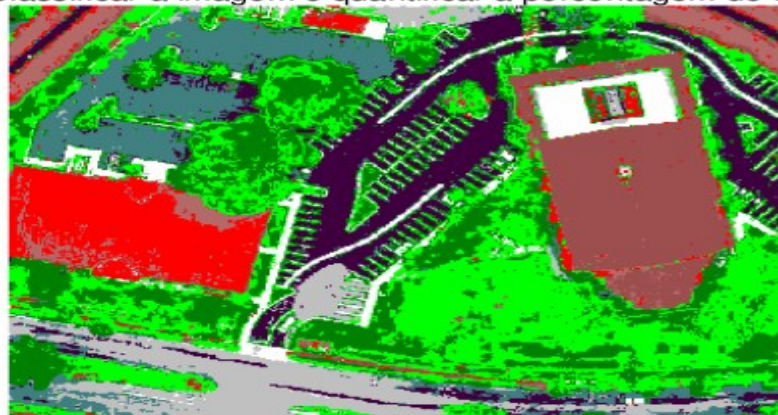
2. Criar as áreas de treinamento



3. Conferir a separabilidade das classes

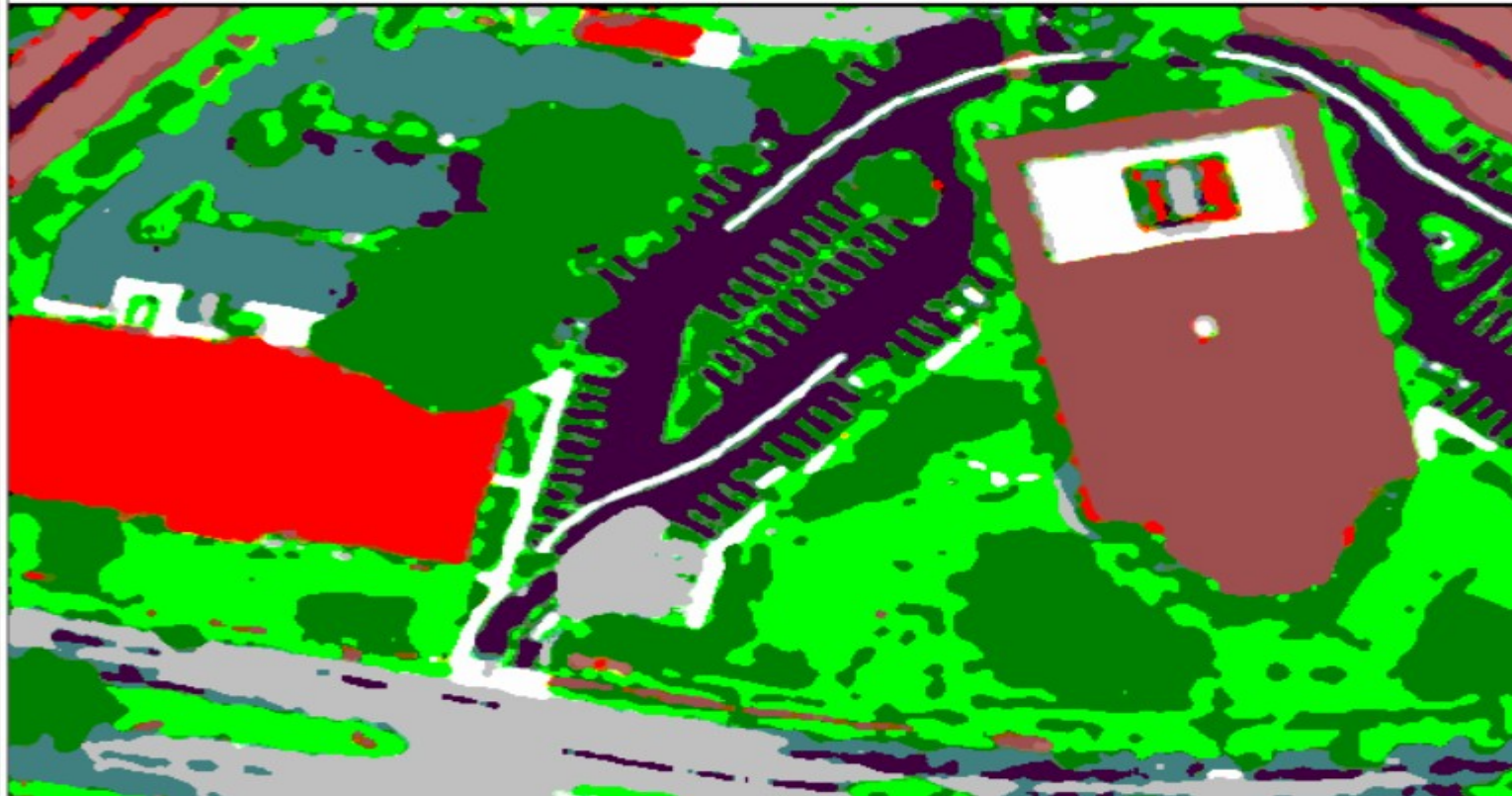


4. Classificar a imagem e quantificar a porcentagem de erros



Análise de Imagens

MAPA TEMÁTICA DA IMAGEM AÉREA DA ZONA RESIDENCIAL



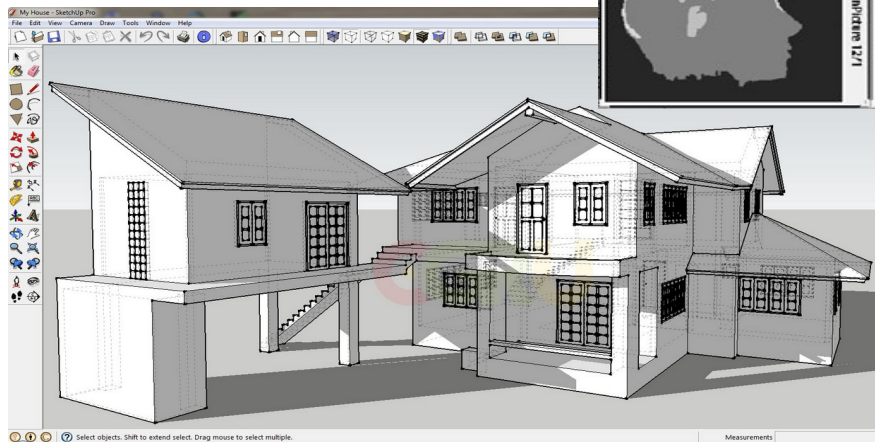
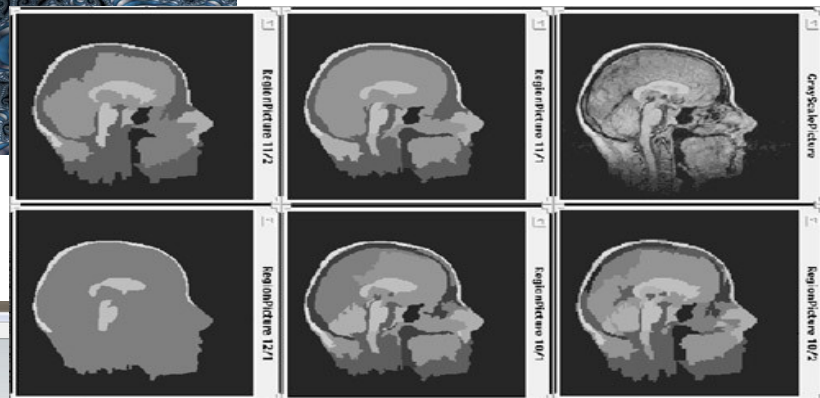
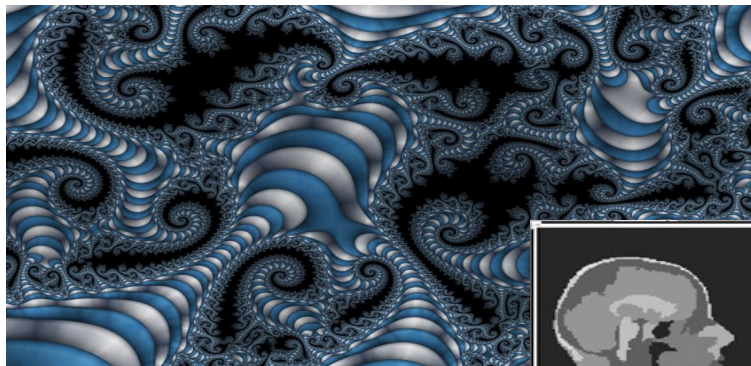
LEGENDA

-  **Telhado vermelho**
-  **Telhado marrom**
-  **Concreto**
-  **Asfalto novo**
-  **Asfalto semi-novo**
-  **Asfalto velho**
-  **Árvore**
-  **Grama**
-  **Terra**

5 metros

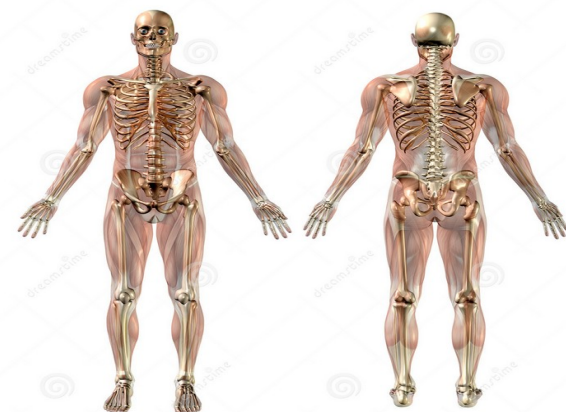
Aplicações

- Arte
- Medicina
- Arquitetura



Aplicações

- Astronomia
- Marketing
- Moda
- Lazer
- Psicologia
- Educação



Aplicações

Augmented Reality
Charizard



Aplicações

```
amaury@ipdm: ~/ipdm/gopro/edge-tracking  amaury@ipdm: ~/ipdm/gopro/opencv/samples/python2  amaury@ipdm: ~/ipdm/gopro/edge-tracking
```

```
amaury@ipdm:~/ipdm/gopro/edge-tracking$ python edge_tracking_partition.py ./GOPR0050.MP4
```

```
kernel = np.ones((1,1,np.uint8))
edges_ver = cv2.dilate(edges, kernel, flags=cv2.CV_4S)
kernel = np.ones((1,1,np.uint8))
edges_ver = cv2.erode(edges_ver, kernel, flags=cv2.CV_4S)

flow_hor = np.where(edges_hor == 0, flow, 0).astype(int)
flow_ver = np.where(edges_ver == 0, flow, 0).astype(int)

if idx < window:
    edgehist_hor[1][1:idx] = flow_hor
    edgehist_ver[1][1:idx] = flow_ver
else:
    edgehist_hor[1] = np.roll(edgehist_hor[1], -1, axis=0)
    edgehist_ver[1] = np.roll(edgehist_ver[1], -1, axis=0)
    edgehist_hor[1][1] = flow_hor.copy()
    edgehist_ver[1][1] = flow_ver.copy()

edgesf_hor = butter_bandpass_filter(edgehist_hor[1].copy(), freqlof, freqsup, 0, 1, real)
edgesf_ver = butter_bandpass_filter(edgehist_ver[1].copy(), freqlof, freqsup, 0, 1, real)

return edgesf_hor, edgesf_ver

I

return gray, grayant
except KeyboardInterrupt:
    print "Ctrl-C"
    return gray, []

threadn = cv2.getNumCores()/CPUF1
pool = ThreadPool(processes = threadn)
pending = deque()

# M1 - 41,2 Hz
# L1 - 55,0 Hz
# R1 - 73,42 Hz
# S1 - 98,0 Hz

first=True
idx=0
start = time.time()
while True:
    r = True
    for t in pending:
        r = r and t.ready()
    if len(pending) > 0 and r:
```

Aplicações

```
amaury@ipdm: ~/IPL2  amaury@ipdm: ~/ipdm/gopro/edge-tracking
```

```
amaury@ipdm:~/ipdm/gopro/edge-tracking$ python edge_tracking_partition.py GOPR0045.MP4
```

```
hsv = np.zeros((h, w, 3), np.uint8)
hsv[0:255,0:255,0] = ang * 180 / np.pi * 255
hsv[0:255,0:255,1] = 255
hsv[0:255,0:255,2] = np.minimum(255 - ang, 255)
bgr = cv2.cvtColor(hsv, cv2.COLOR_HSV2BGR)
return bgr

image = cv2.imread('')
imgout = img

img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

img = cv2.cvtColor(img, cv2.COLOR_GRAY2BGR)

I

img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

cap = cv2.VideoCapture(0)

cv2.namedWindow('edge')
cv2.createTrackbar('low', 'edge', 0, 255, nothing)
cv2.createTrackbar('high', 'edge', 255, 255, nothing)

window = cv2.WindowPropertyGetter('edge')
application = cv2.ApplicationPropertyGetter('edge')
record = cv2.WindowPropertyGetter('edge')
lo = cv2.WindowPropertyGetter('edge')
hi = cv2.WindowPropertyGetter('edge')

def main():
    frame = cap.read()
    (h,w) = frame.shape

    edges = cv2.Canny(gray,lo,hi)

    flow = cv2.calcOpticalFlowFarneback(grayant, gray, None, 0.5, 15, 3, 5, 1.2, 0)

    kernel = np.ones((1,1),np.uint8)
    edges_hor = cv2.dilate(edges,kernel,iterations=2)
    kernel = np.ones((1,1),np.uint8)
    edges_ver = cv2.erode(edges_hor,kernel,iterations=2)

122 kernel = np.ones((3,1),np.uint8)
123 edges_ver = cv2.dilate(edges,kernel,iterations=2)
124 kernel = np.ones((1,3),np.uint8)
125 edges_ver = cv2.erode(edges_ver,kernel,iterations=2)
126
127 flow_hor = np.where(edges_hor>200,flow[...],0)
128 flow_ver = np.where(edges_ver>200,flow[...],0)
129
130 if idx>window:
131     edgehist_hor[i][idx]=flow_hor
132     edgehist_ver[i][idx]=flow_ver
133 else:
134     #edgehist_hor[i][0:-2] = edgehist_hor[i][1:-1].copy()
135     #edgehist_hor[i][-1] = flow_hor.copy()
136     #edgehist_ver[i][0:-2] = edgehist_ver[i][1:-1].copy()
137     #edgehist_ver[i][-1] = flow_ver.copy()
```

Aplicações

- Segurança
 - Detecção de movimento



Aplicações

