CMSC426: Project 3 Rotobrush: Hands-On

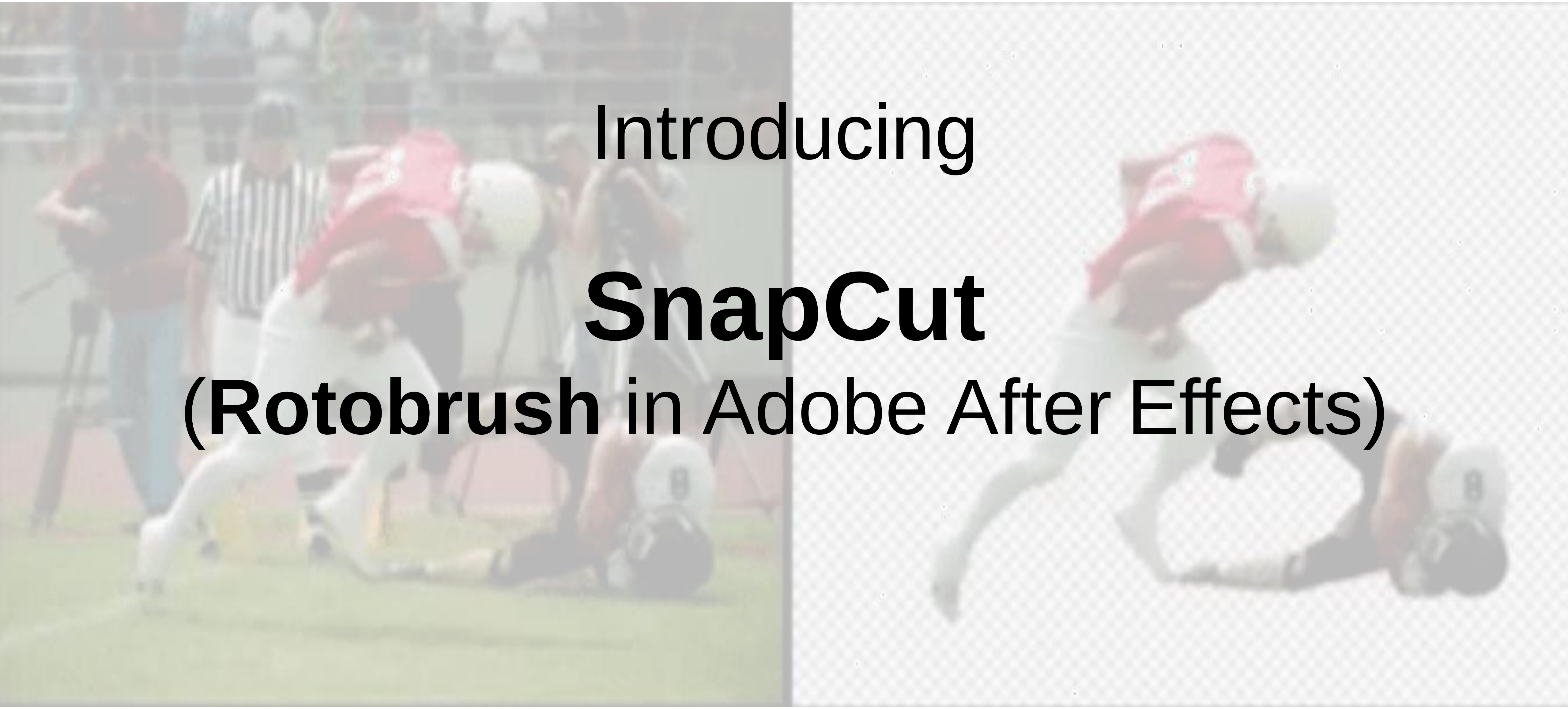
Chahat Deep Singh and Chethan M Parameshwara

Perception and Robotics Group, University of Maryland, College Park
prg.cs.umd.edu

PRG





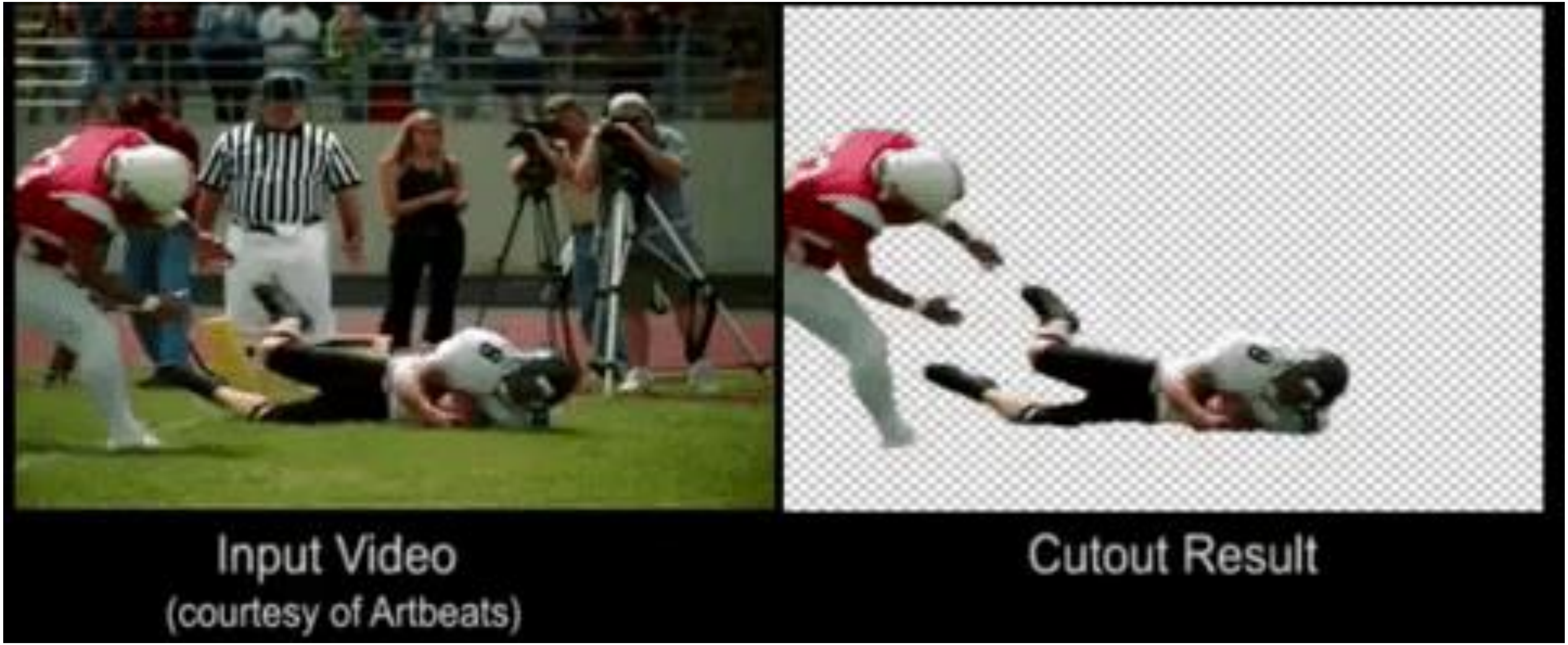



Introducing

SnapCut

(**Rotobrush** in Adobe After Effects)

Rotobrush Output

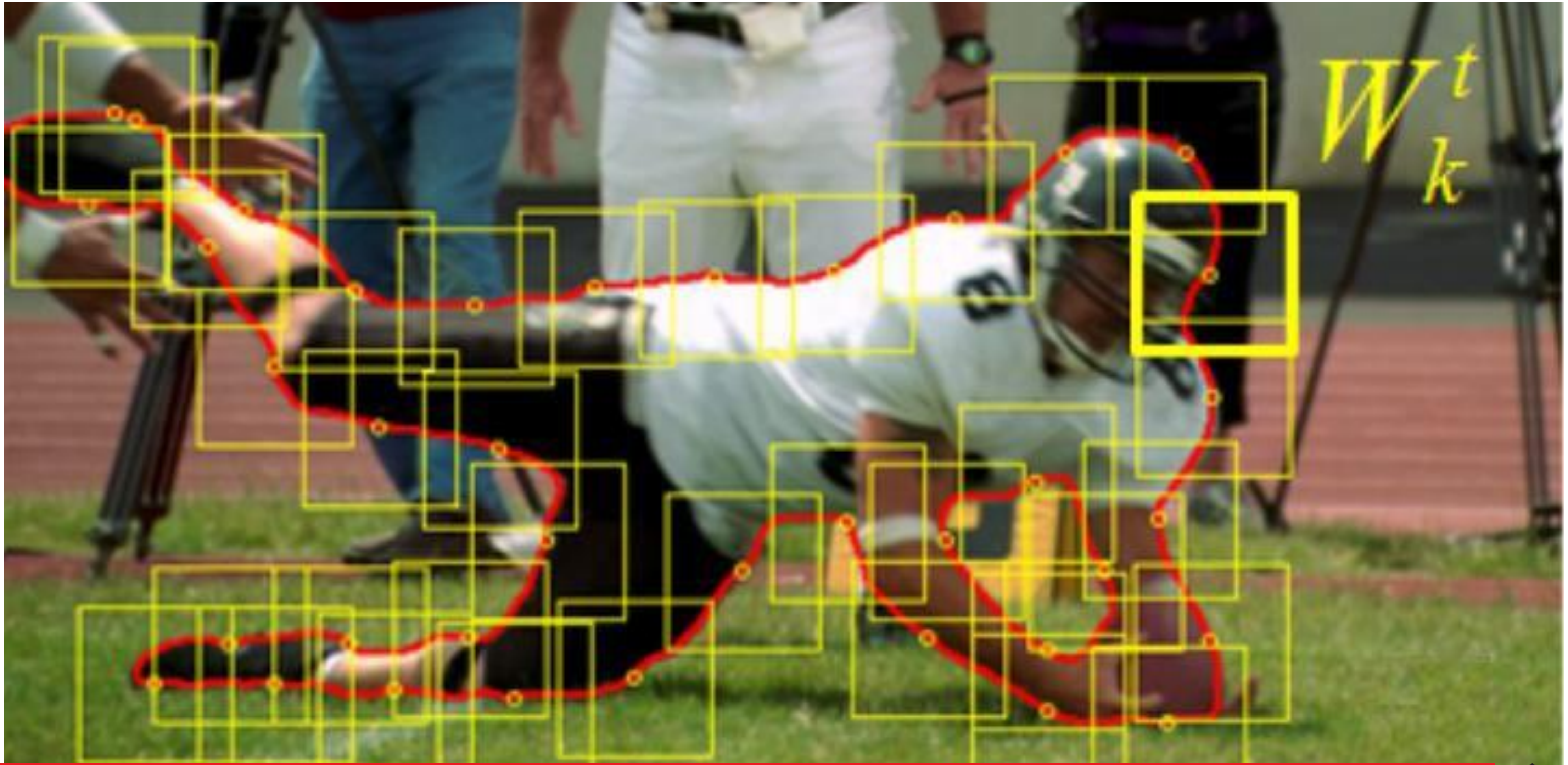


Segmenting the object of interest in the first frame

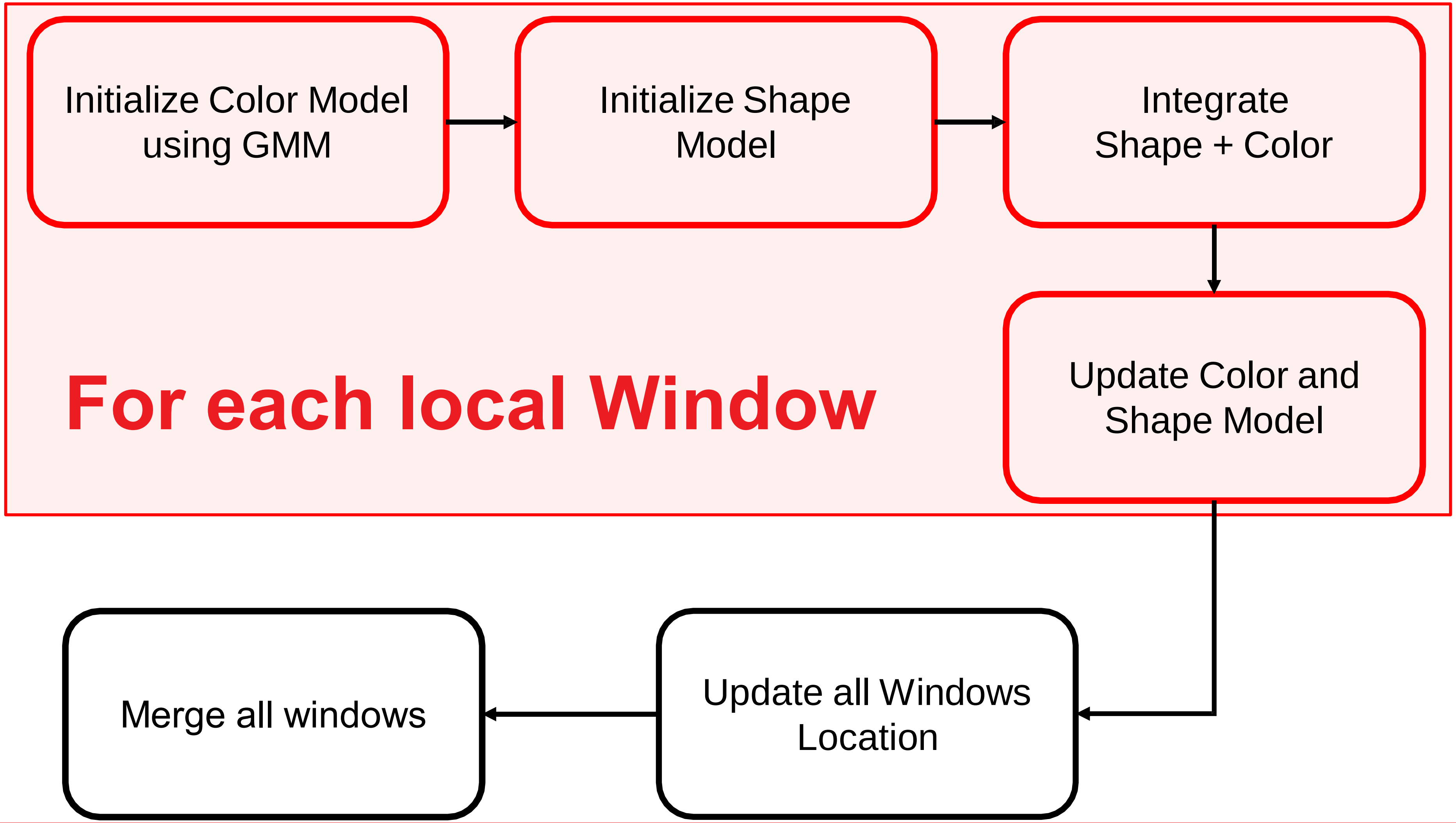
- Let's call it foreground or $\mathcal{F}$ for ease.
- Rest everything is background or $\mathcal{B}$
- Use `roipoly` in MATLAB.

Create Local Classifiers

Create Local Windows



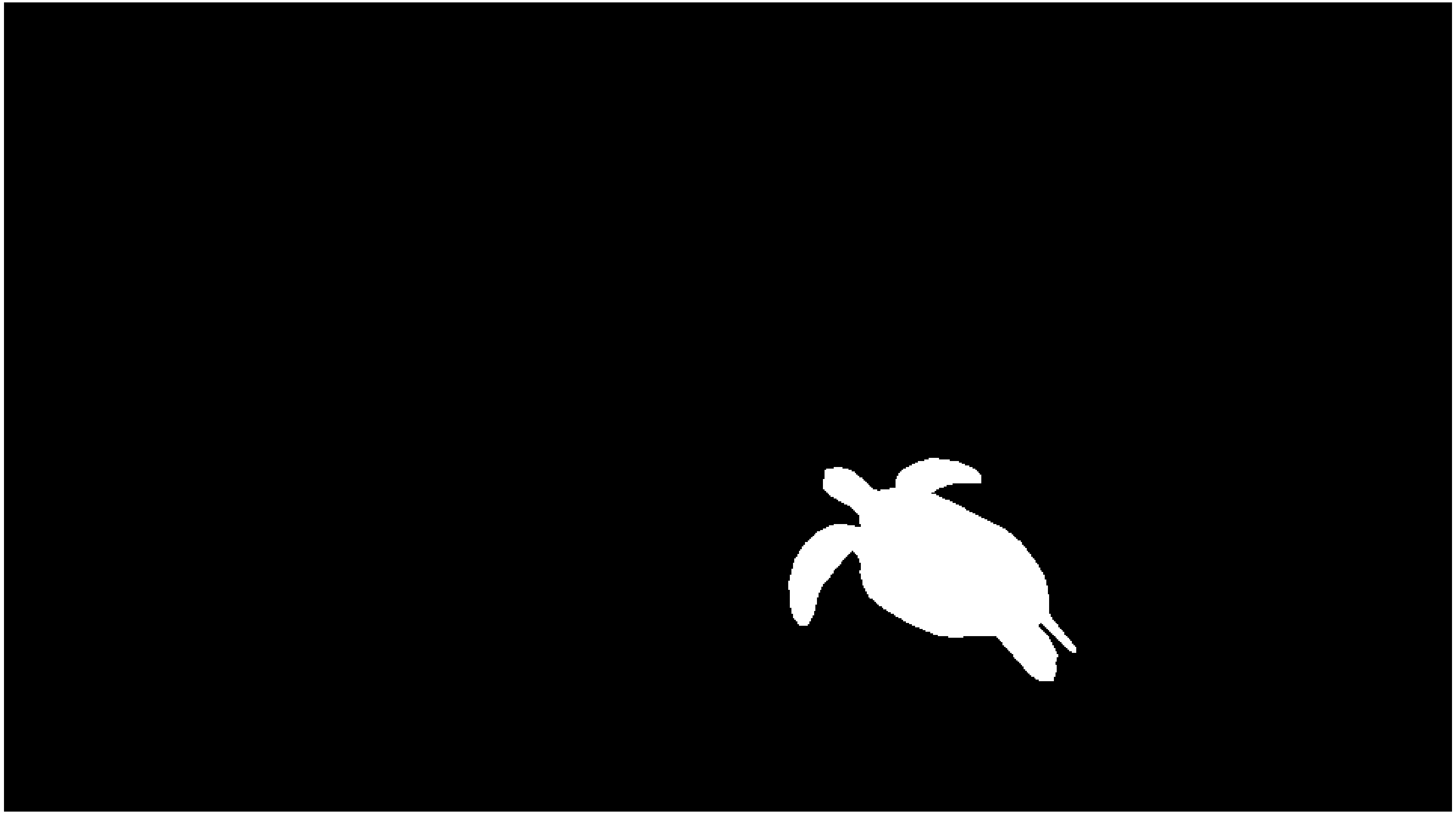
An Overview


Initialize Color Model using GMM
Initialize Shape Model
Integrate Shape + Color
Update Color and Shape Model
For each local Window
Update all Windows Location
Merge all windows

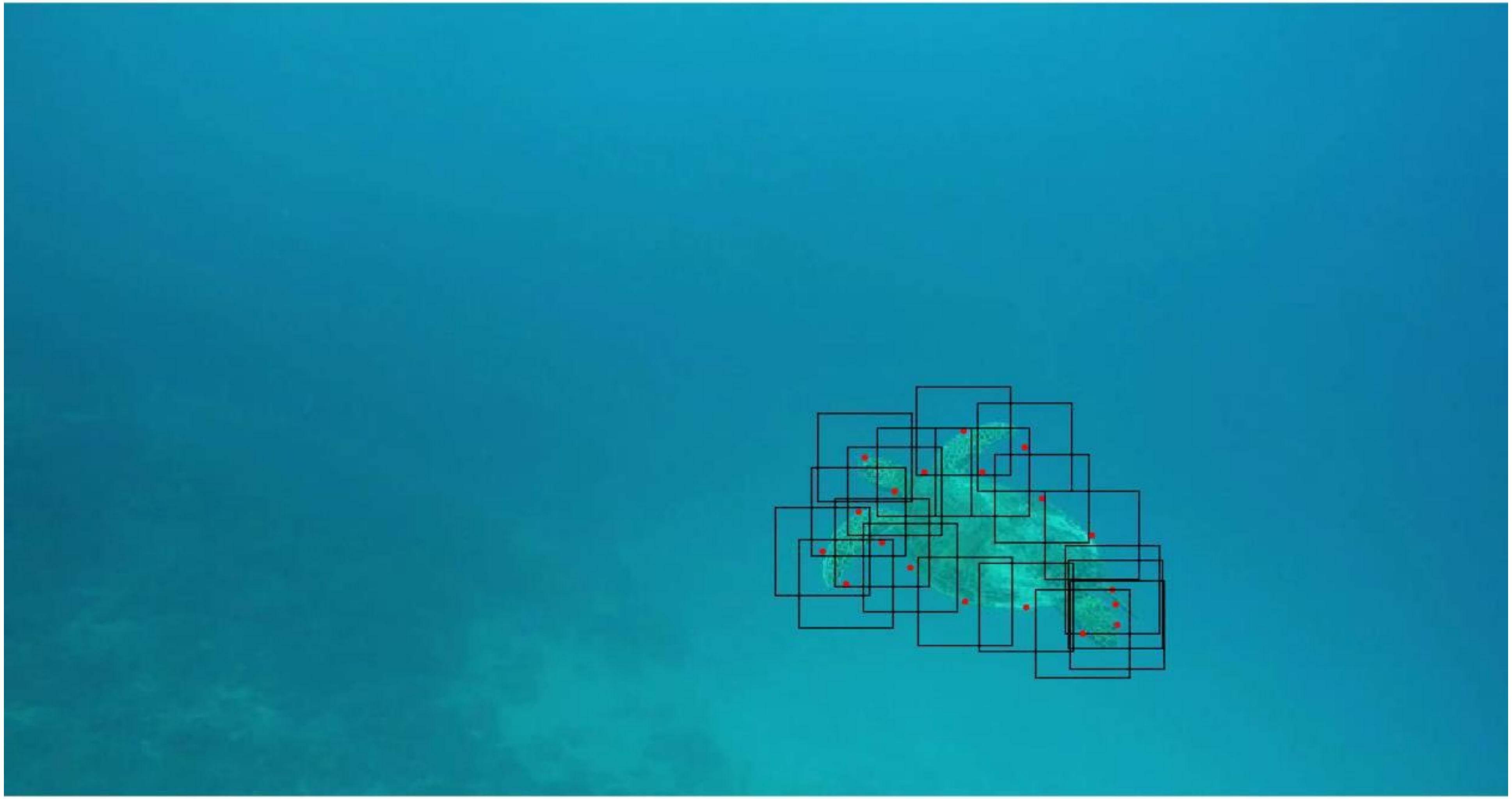
Let’s dig deep

Input

Create Mask



Local Window (`initLocalWindows.m`)



Local Window (initLocalWindows.m)

```
[mask_outline, LocalWindows] = initLocalWindows(images{i}, mask,…
                                        NumWindows, WindowWidth, flag)
```

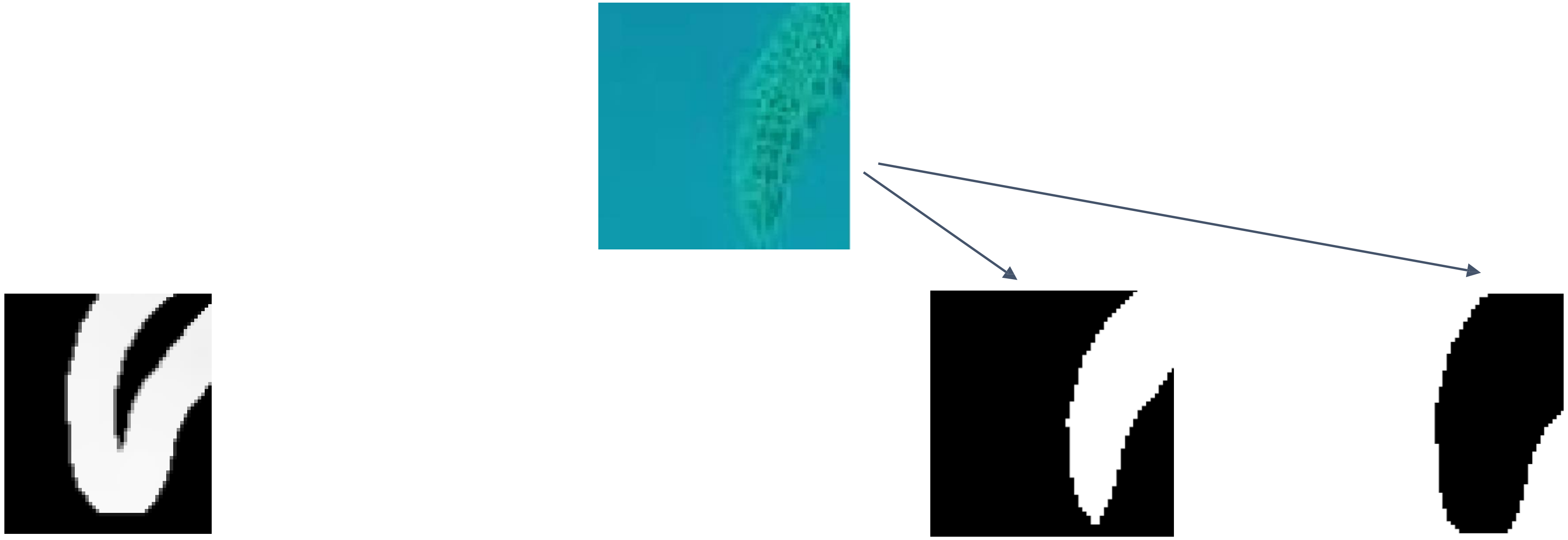
For each boundary point:

```
[WindowCentersX, WindowCentersY] = ...
             equidistantPointsOnPerimeter(Boundary(:,2), Boundary(:,1),
PointsForBoundary(i));

  AllWindowCentersX =[AllWindowCentersX; WindowCentersX];
  AllWindowCentersY =[AllWindowCentersY; WindowCentersY];
```

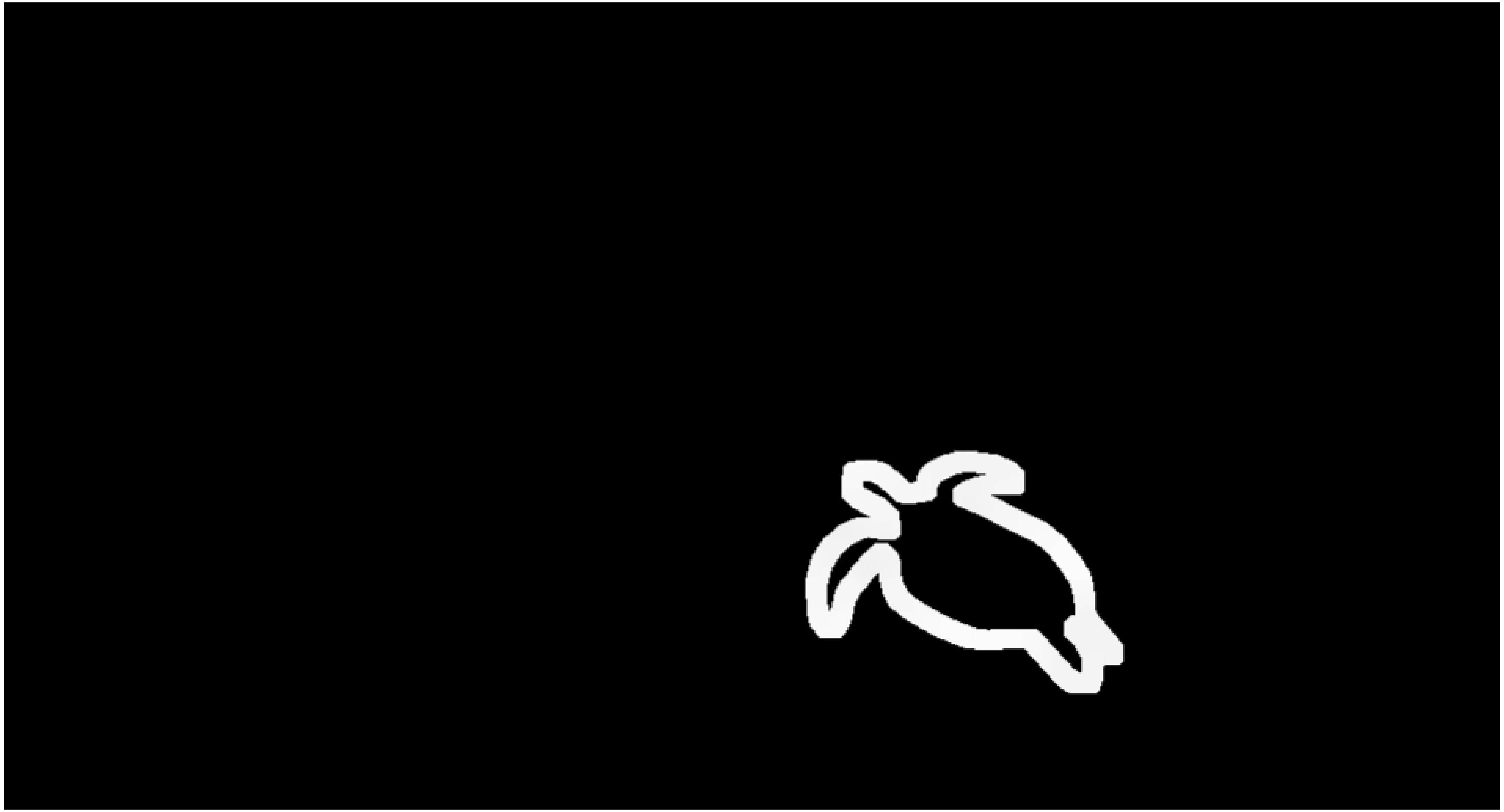
Where
`EquidistantPointsOnPerimeter` finds the equally spaced points along the perimeter of a polygon. Check implementation from MATLAB blog.
Source: https://blogs.mathworks.com/steve/2012/07/06/walking-along-a-path/#5d684ab6-0edb-4c52-8df2-4bce4a79ee80

Color model (`initColorModels.m`)



$$p_c(x) = p_c(x|\mathcal{F}) \;/\; \left(p_c(x|\boxed{\mathcal{F}}) + p_c(x|\boxed{\mathcal{B}})\right)$$

Color model (`initColorModels.m`)



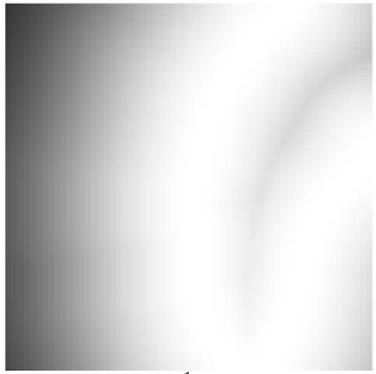
Color model (`initColorModels.m`)

L_t is the known segmentation label

p_c is foreground probability

$d(x)$ is the spatial distance between x and the foreground boundary

σ_c is fixed as half of the window size



$$\omega_c(x) = exp(-\boxed{d^2(x)} \,/\, \sigma_c^2)$$

$$f_c = 1 - \frac{\int_{W_k} \boxed{|L^t(x) - p_c(x)|} \cdot \boxed{\omega_c(x)}\, dx}{\int_{W_k} \omega_c(x)\, dx}$$

Color model (initColorModels.m)

```
ColorModels = initColorModels(images{1}, mask, mask_outline,…,
        LocalWindows, BoundaryWidth, WindowWidth, DistTransMaskOutline);
```

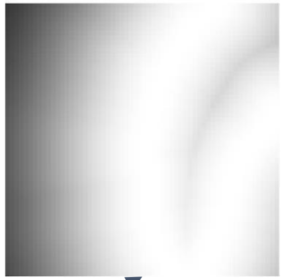
where `DistTransMaskOutline = bwdist(mask_outline); %Find global distance transform` → d

```
F_GMM_Model = fitgmdist(ForegroundPixelsRGB, k, 'RegularizationValue',
0.001);
B_GMM_Model = fitgmdist(BackgroundPixelsRGB, k, 'RegularizationValue',
0.001);
```

`For all x and y:`

- Compute `pdf` of both the models above: `Pf` and `Pb`
- Compute `fc`

Shape model (`initShapeConfidences.m`)

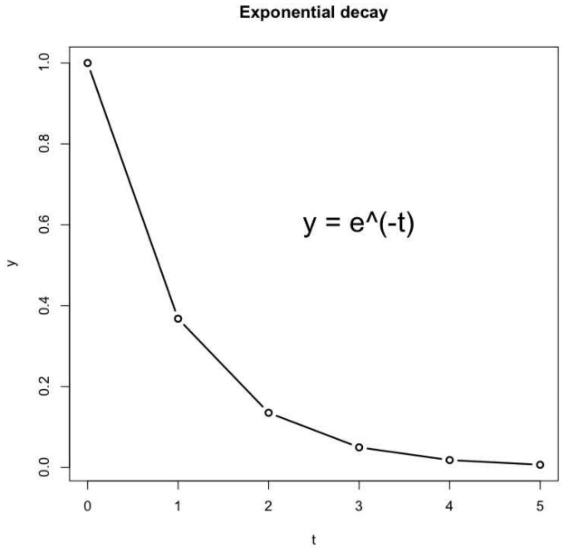


$$f_s(x) = 1 - exp\left(\boxed{-d^2(x)}\ /\ \boxed{\sigma_s^2}\right)$$

$\boldsymbol{f_s}$ is shape confidence mask

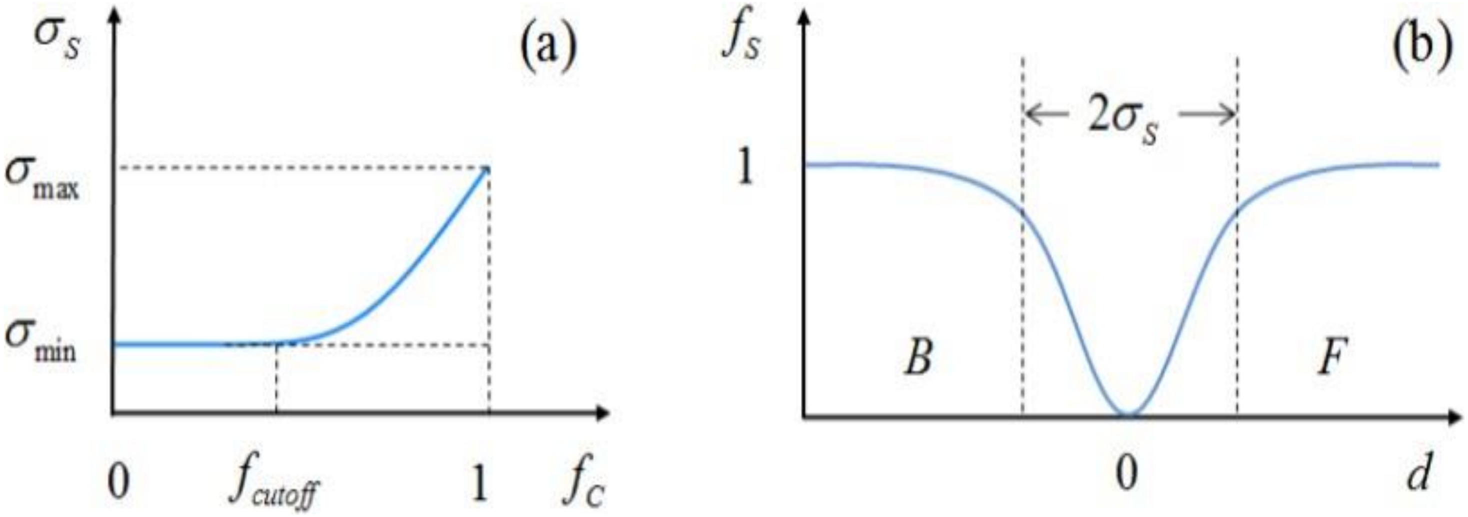
$\boldsymbol{d(x)}$ is the spatial distance between x and the foreground boundary

$\boldsymbol{\sigma_s}$ is a parameter which depends on $\boldsymbol{f_c}$

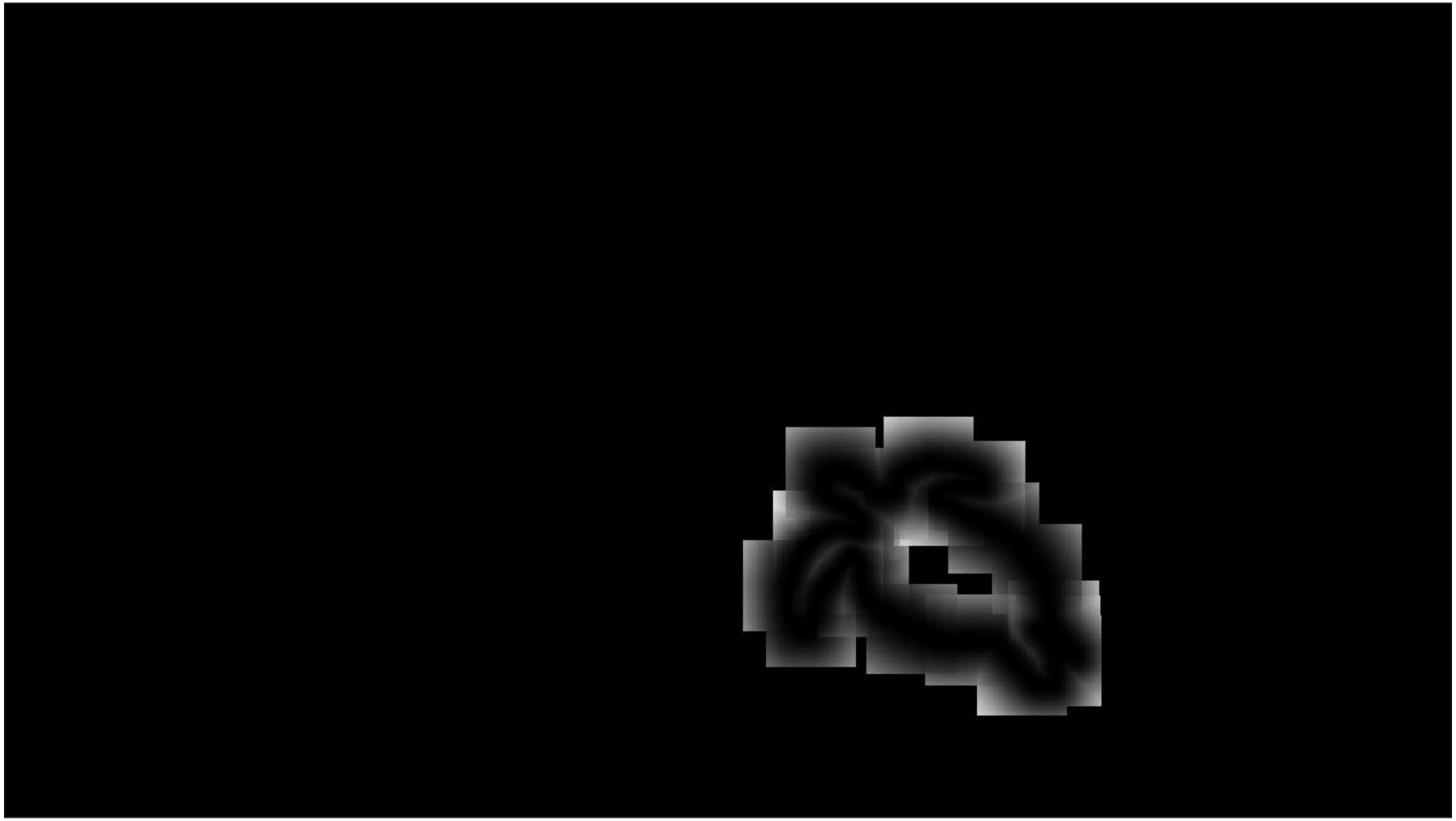


Shape model (`initShapeConfidences.m`)

$$\sigma_s = \begin{cases} \sigma_{min} + a(f_c - f_{cutoff})^r & f_{cutoff} < f_c \le 1, \\ \sigma_{min} & 0 \le f_c \le f_{cutoff}, \end{cases}$$



Shape model (`initShapeConfidences.m`)



Shape model (`initShapeConfidences.m`)

```
ShapeModels = initShapeConfidences(LocalWindows, ColorModels,...
                WindowWidth, SigmaMin, a, fcutoff, R,DistTransMaskOutline);
```

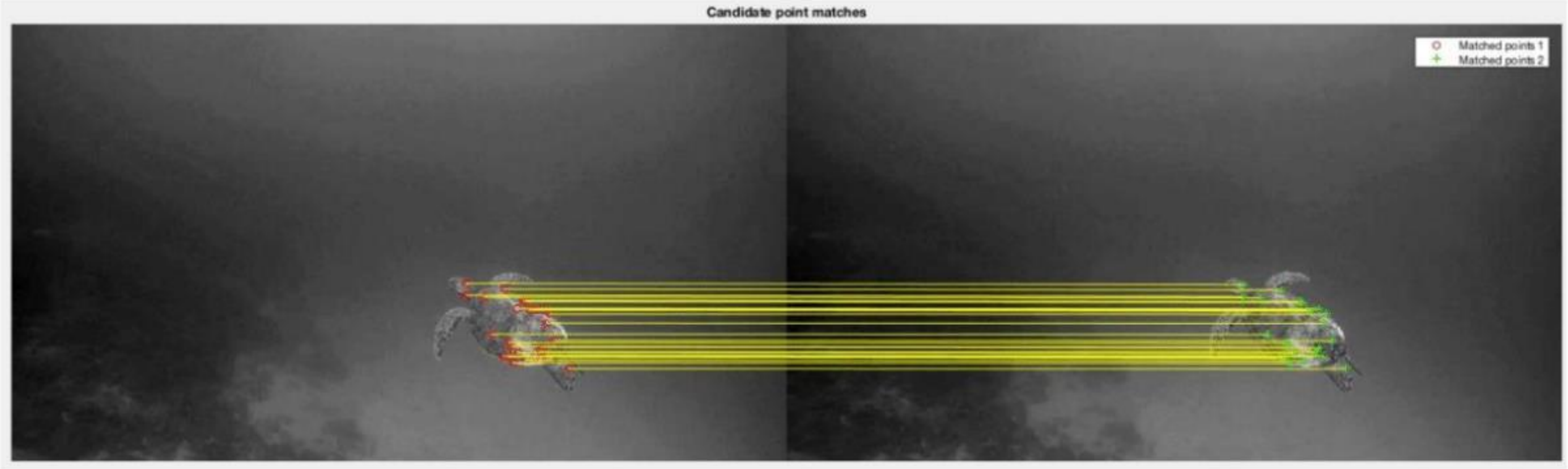
```
a = (sigmaMax - sigmaMin)/((1-fcutoff)^R);
```

Define `sigmaS` for two different `fc` cases

For all x and y:

```
        f_s = 1 - exp(-(d^2)/(sigmaS^2));
```

Local Window Propagation (`calculateGlobalAffine.m`)



Local Window Propagation (`calculateGlobalAffine.m`)

```
[warpedFrame, warpedMask, warpedMaskOutline, warpedLocalWindows] =
                        calculateGlobalAffine(images{prev}, images{curr}, …,
                                                    mask, LocalWindows);
```

- Detect and match features between the two frames
- Estimate geometric transformation from matching point pairs. Use `estimateGeometricTransform` and get the `tform`
- Warp the image/Mask using `imwarp`
- Get `WarpedMaskOutline` using `bwperim`
- Now find the `WarpedLocalWindows` using the `invert` of the `tform`

Optical Flow Wrapping (`localFlowWrap.m`)

Optical Flow Wrapping (`localFlowWrap.m`)

```
[NewLocalWindows] = localFlowWarp(WarpedPrevFrame, CurrentFrame, … ,
                                          LocalWindows, Mask, WindowWidth)
```

- Compute Optical Flow: use `opticalFlowFarneback` and `estimateFlow`
- For all `localWindows`
  - Find the new Local Window location by something like:
    `old_window_location + average_flow` for that windows

Updating the Shape and Color Models (`updateModels.m`)

$$p_{\mathcal{F}}^k(x) = f_s(x)L^{t+1}(x) + (1 - \boxed{f_s(x)})\, p_c(x)$$

$$p_{\mathcal{F}}(x) = \frac{\sum_k \boxed{p_{\mathcal{F}}^k(x)}\boxed{(|x - c_k|}+\epsilon)^{-1}}{\sum_k (|x - c_k| + \epsilon)^{-1}}$$

L^{t+1} is the warped segmentation label from previous frame

f_s is the shape confidence mask

p_c is the foreground probability

c_k is the center of window

e is a small constant

Updating the Shape and Color Models (`updateModels.m`)

```
ShapeModels = updateShapeConfidences(LocalWindows, ColorModels,...,
                    WindowWidth, SigmaMin, a, fcutoff, R, DistTransMaskOutline)
```

where `DistTransMaskOutline = Bwdist(warpedMaskOutline)`

```
a = (sigmaMax - sigmaMin)/((1-fcutoff)^R);
```

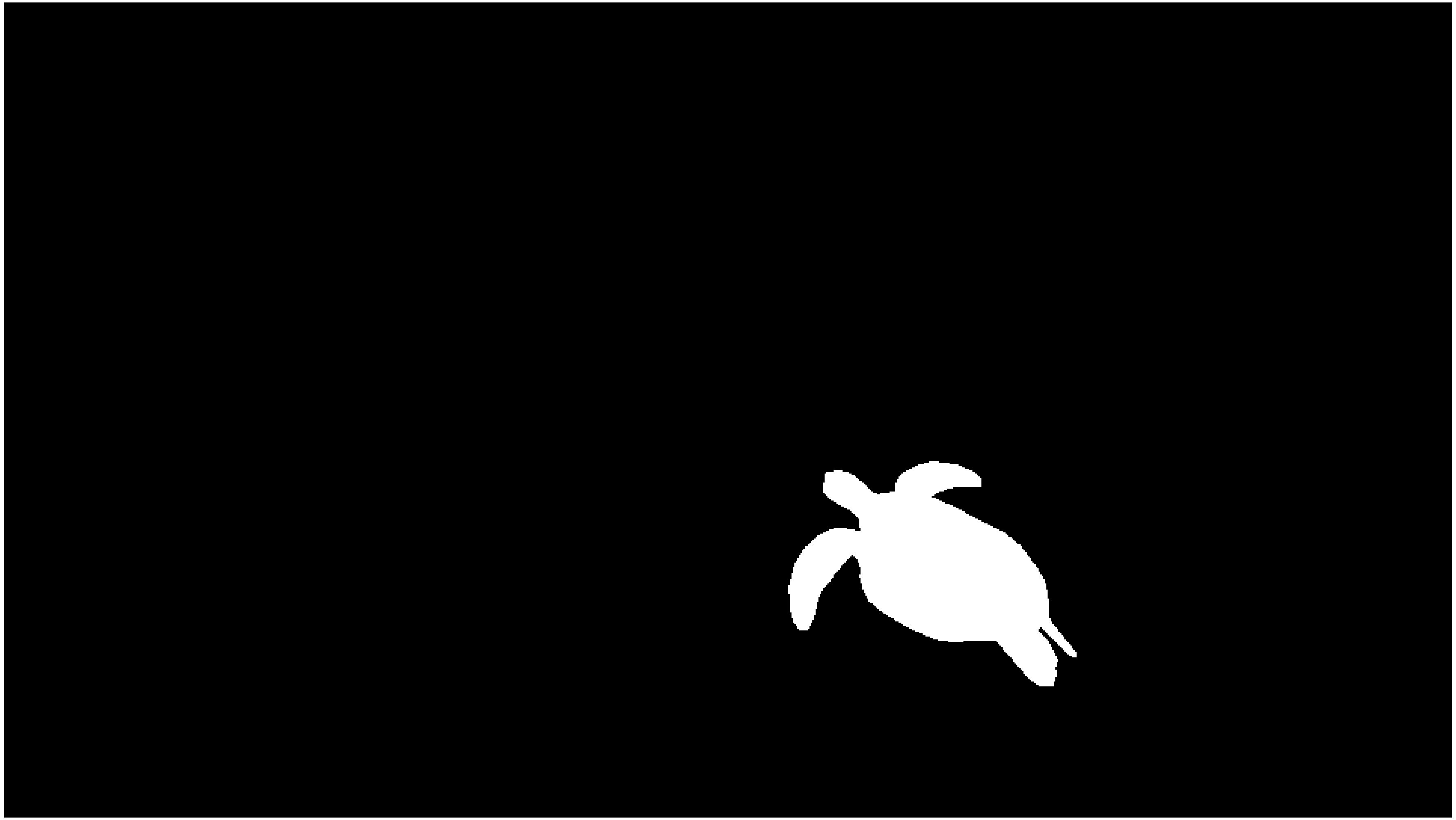
- Define `sigmaS` for two different `fc` cases
- For all x and y:
  ```
  f_s = 1 - exp(-(d^2)/(sigmaS^2));
  ```

Then Combine Shape and Color models

- For each window:
  - `pf = fs.*L + (oneArr-fs).*pc;`

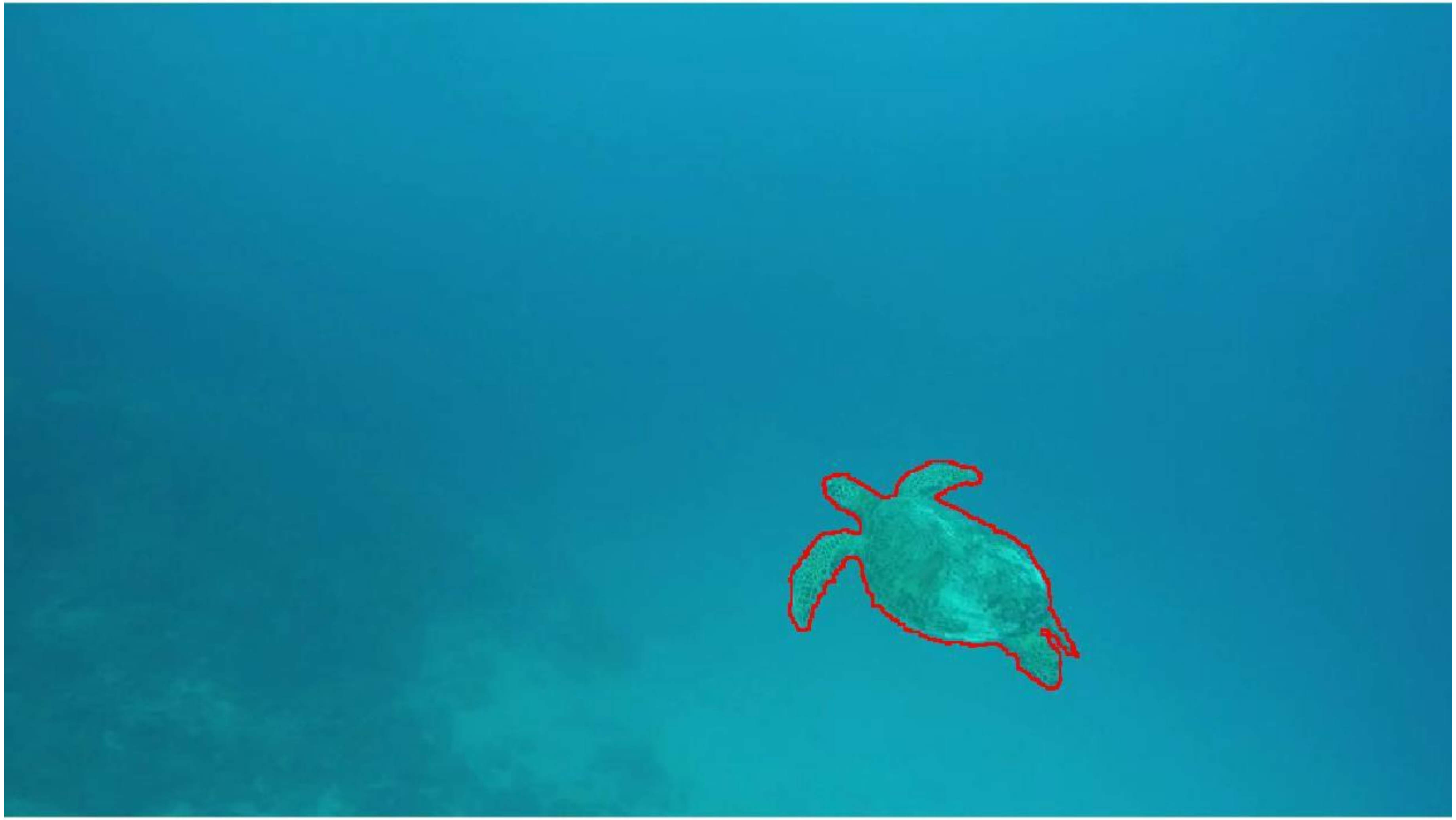
$$p_{\mathcal{F}}^{k}(x) = f_s(x)L^{t+1}(x) + (1 - f_s(x))\, p_c(x)$$

Updating the Shape and Color Models (`updateModels.m`)



Updating the Shape and Color Models (`updateModels.m`)

Hint: Update the color model based on the area of the new foreground probability mask



Pseudo-code (`myRotobrush.m`)

Algorithm 1 Rotobrush

```
procedure MYROTOBRUSH
    set parameters
    load images
    create mask
    initLocalWindows()                  ▷ initialize local window
    initColorModels()                   ▷ initialize Color model
    initShapeConfidences()              ▷ initialize Shape model
    for every image do
        calculateGlobalAffine()         ▷ transform between previous and current frames
        localFlowWarp()                 ▷ local warping based on optical flow
        updateModels()                  ▷ update color and shape model
    end for
end procedure
```

CMSC426: Project 3
Deadline: Nov 14 2018
(Midnight)

Chahat Deep Singh and Chethan M Parameshwara

Perception and Robotics Group,
University of Maryland, College Park
[prg.cs.umd.edu](prg.cs.umd.edu)

PRG

COMPUTER SCIENCE
UNIVERSITY OF MARYLAND

Thank You!

PRG

prg.cs.umd.edu