Quick Shift (QS) summary
1. Applications

• QS: Mode seeking method, used for clustering (on any data), faster than Medoid Shift and Mean Shift (Mean Shift was spearheaded by a famous Rutgers professor, Prof. Peter Meer).

• On images: region segmentation, creating superpixels (blobs). Most likely we want to separate the regions corresponding to different objects, e.g. picking out a red ball from a bunch of blue balls on a green field. These superpixels can be further used in tracking, object localization and recognition, etc.

• Nowadays this type of passive segmentation is superseded by better machine learned models with far better performance.
2. Method

• QS comprises of 2 main, separate steps
  1. Calculating pixel density, for every pixel, using a sliding Parzen window. The window size is controlled by $\sigma$.
  2. Growing a tree upon a seeding pixel. Given a pixel and its density, find among its neighborhood a pixel whose density is higher, why color their distance is within the $\tau$ limit.

• In the paper, Brian Fulkerson and Stefano Soatto, “Really quick shift: Image segmentation on a GPU”, ECCVW’10, there is another hyperparameter $\lambda$, to scale (R,G,B) pixels down to balance their importance w.r.t. their spatial positions (x,y). In the code, $\lambda$ is fixed as 32/255 in the `image_to_matlab()` function.
2.1. Pixel density

• Density function: Given $N$ data points $x_1, ..., x_N$, compute a Parzen density estimate around each point using a Gaussian kernel

$$P(x) = \frac{1}{2\pi\sigma^2N} \sum_{i=1}^{N} e^{-\frac{\|x-x_i\|^2}{2\sigma^2}}$$

• For image, $x = (r,g,b,x,y)$ - color values + spatial position.
• Pixel inside neighborhood of similar pixels has high density.
• $\sigma$ is the Gaussian deviation ($\sigma^2$ is variance).
• Actual implementation:
  
  ```python
  def computeDensity():
      for x in all pixels:
          P[x] = 0
      for n in all pixels less than 3*sigma away:
          P[x] += exp(-(f[x]-f[n])**2 / (2*sigma*sigma))
  ```

• Pixel inside neighborhood of similar pixels has high density.
2.2. Growing neighborhood trees

- Given a pixel $x$, find within a window of size $2\tau$ centered at $x$ a pixel $y$ st:
  - $P(y) > P(x)$ (go from the edge to the center of the object/segment)
  - $D(x,y) = |I(x) - I(y)| < \tau$
  - $y = \text{argmin}(D(x,y))$

- The function:
  - The links are stored in an adjacency matrix and will be sorted to output image segments.

```python
function linkNeighbors()
    for x in all pixels
        for n in all pixels less than tau away
            if $P[n] > P[x]$ and distance($x,n$) is smallest among all $n$
                $d[x] = \text{distance}(x,n)$
                $\text{parent}[x] = n$
```
2.3. Values of $\sigma$ and $\tau$

- Increasing $\sigma$ smoothes the underlying estimate of the density, providing fewer modes, i.e. fewer & bigger segments.
- Increasing $\tau$ increases the average size of a region as well as the error in the distance estimate, i.e. fewer & bigger segments.
- Increasing either of them requires more (quadratic) computation.
2.3. Values of $\sigma$ and $\tau$

- An example

![Original](image1.png)

$\sigma = 2$

$\tau = 10$

$\sigma = 10$

$\tau = 20$
2.4. Discussion

• You want to improve your chances to recognize objects in an image. If you know something about the image properties (domain expert knowledge), would you choose the parameter settings differently, thereby producing a segmentation that is better for the given recognition task?

Examples:
- image of a few objects on a shelf
- image of a busy intersection
- image of a single car/person on a neutral background
- image of a building
Shelf

- Small objects: $\sigma = 6$, $\tau = 10$; can make out blocks of different drinks
Intersection

- Localize cars and buses: $\sigma = 10$, $\tau = 20$; small objects (moderate $\sigma$ so that we don’t smooth out the contour) but we want bigger blobs (higher $\tau$)
Single car on a dark, complex background

- $\sigma = 10$, $\tau = 20$; moderate $\sigma$ to not blur out the edge, higher $\tau$ to have bigger blobs.
Building

- $\sigma = 10, \tau = 120$: moderate $\sigma$, bigger value will blend the building with the tree. Very high $\tau$ because the building occupies large part of the image.