DRAFT: Computer Vision for the Uninitiated

Sudeep Sarkar

February 4, 2020

Chapter 1

Segmentation using Mean Shift

- Section 5.3.2 of text book and paper by Peter Meer
- Core algorithmic task: Efficiently find the peaks of a distribution given a set of samples.
- Let $\mathbf{x}_i$ represent the vector representing a pixel in terms of color or other attributes.
- Kernel distribution is an estimate of a continuous probability density function from samples. It is achieved by replacing a continuous function, which we will call the kernel function $G(\mathbf{x})$ at sample and adding them up. It effectively gives us a continuous interpolation of the data between the samples. The most used kernel function is a Gaussian distribution.

  $$f(\mathbf{x}) = c\sum_{i=1}^{N} G(\mathbf{x}-\mathbf{x}_i) = c\sum_{i=1}^{N}\exp\left(-\frac{||\mathbf{x}-\mathbf{x}_i||^2}{2h^2}\right) \tag{1.1}$$

  The constant c is a normalizing constant to make the function integrate out to one. For a vector valued sample, this is basically a sum of Gaussian balls centered on the samples.
- **Multiple restart gradient ascent:** Our approach to finding the peaks in this continuous function would be to start from a random point and then to move it iteratively towards a peak. Ideally, we would like to start from many different points so that we are able to that converge around all the available peaks. How many samples

should we start from? That is an issue that is needs to be looked at during implementation time. This is a kin to hill climbing but from many different points.

Another way to visualize this process is to imagine throwing a random number of particles on this continuous function and then moving these particles up the hill wherever they are placed. The idea is that overtime the particles will accumulate on the hilltops nearest to them. Let us first see how we can quantify the step we need to take at each iteration from each particle or starting point.

- For each particle (or estimate of a possible peak, $\mathbf{y}_k$, we modify it by an uphill step, which is estimated by the gradient of the function at $\mathbf{y}_k$.
- So, first we need to an expression of the gradient of the continuous function $f(\mathbf{x})$. Note that the function is a scalar value define over a vector field.
- Worksheet problem: Draw the Gaussian function. Draw the first derivative of a Gaussian. Express the gradient of a one dimensional Gaussian kernel function and plot the function. Let

$$f(x) = c\sum_{i=1}^{N} \exp\left(-\frac{(x-x_i)^2}{2h^2}\right) \tag{1.2}$$

$$f'(x) = c\frac{df(x)}{dx} = \sum_{i=1}^{N} -\frac{x-x_i}{h^2}\exp\left(\frac{(x-x_i)^2}{2h^2}\right) \tag{1.3}$$

Note that this is equivalent to placing a first derivative of a Gaussian function at each sample point.

-

$$\begin{aligned} \nabla f(\mathbf{x}) &= c\sum_{i=1}^{N} -\frac{\mathbf{x}-\mathbf{x}_i}{h^2}\exp\left(\frac{||\mathbf{x}-\mathbf{x}_i||^2}{2h^2}\right) && (1.4)\\ &= c\sum_{i=1}^{N} -\frac{\mathbf{x}-\mathbf{x}_i}{h^2}G(\mathbf{x}-\mathbf{x}_i) && (1.5)\end{aligned}$$

•

$$\begin{aligned}
\nabla f(\mathbf{x}) &= \frac{c}{h^2}\sum_{i=1}^{N} -(\mathbf{x}-\mathbf{x}_i)G(\mathbf{x}-\mathbf{x}_i) && (1.6)\\
&= \frac{c}{h^2}\sum_{i=1}^{N} -\mathbf{x}G(\mathbf{x}-\mathbf{x}_i)+\mathbf{x}_iG(\mathbf{x}-\mathbf{x}_i) && (1.7)\\
&= \frac{c}{h^2}\left(-\mathbf{x}\sum_{i=1}^{N} G(\mathbf{x}-\mathbf{x}_i)+\sum_{i=1}^{N}\mathbf{x}_iG(\mathbf{x}-\mathbf{x}_i)\right) && (1.8)\\
&= \frac{c}{h^2}\left(\sum_{i=1}^{N} G(\mathbf{x}-\mathbf{x}_i)\right)\left(-\mathbf{x}+\frac{\sum_{i=1}^{N}\mathbf{x}_iG(\mathbf{x}-\mathbf{x}_i)}{\sum_{i=1}^{N}G(\mathbf{x}-\mathbf{x}_i)}\right) && (1.9)\\
&= \left[\frac{c}{h^2}\sum_{i=1}^{N} G(\mathbf{x}-\mathbf{x}_i)\right]\left(\frac{\sum_{i=1}^{N}\mathbf{x}_iG(\mathbf{x}-\mathbf{x}_i)}{\sum_{i=1}^{N}G(\mathbf{x}-\mathbf{x}_i)}-\mathbf{x}\right) && (1.10)
\end{aligned}$$

- Note that in the last expression above, the term in the square bracket is a scalar quantity, while the term in parenthesis is a vector. The direction of the vector is the direction of the gradient. We will denote this vector by $\mathbf{m}(\mathbf{x})$.

$$\mathbf{m}(\mathbf{x}) = \left(\frac{\sum_{i=1}^{N}\mathbf{x}_iG(\mathbf{x}-\mathbf{x}_i)}{\sum_{i=1}^{N}G(\mathbf{x}-\mathbf{x}_i)}-\mathbf{x}\right) \qquad (1.11)$$

- Now that we have an expression for the gradient vector, we can use it to modify the current estimate of the peak denoted by $\mathbf{y}_k$.

$$\mathbf{y}_{k+1} = \mathbf{y}_k + \mathbf{m}(\mathbf{y}_k) = \frac{\sum_{i=1}^{N}\mathbf{x}_iG(\mathbf{y}_k-\mathbf{x}_i)}{\sum_{i=1}^{N}G(\mathbf{y}_k-\mathbf{x}_i)} \qquad (1.12)$$

- The above question is basically saying that the new estimate of the peak location is the weighted mean of the sample, $\mathbf{x}_i$. The weights are local because the Gaussian function is locally weighted or has high values only with a small region around the center.

- A possible in class assignment would be to draw the Gaussian functions over each sample and see how the above expression is really a weighted mean of the local values.

- Intuitively, the above method works because by definition you have more samples around the (histogram) peaks and a random starting point will be pulled towards denser set of points.

- Although the above is a derivation for the Gaussian kernel, Comaniciu and Meer (2002) has shown that this also works for other kernel functions as long as they are monotonically decreasing with distance from the center. Another kernel functions studied by them was the Epamechnikov kernel that is defined over the unit ball given by:

$$E(\mathbf{x}-\mathbf{x}_i) = \max\left(0, 1 - ||\mathbf{x}-\mathbf{x}_i||^2\right) \tag{1.13}$$

- Possible worksheet: What is the $\mathbf{m}(\mathbf{x})$, the direction of the gradient at any point, for this kernel? What is the update equation?

- There are many methods that have been proposed in the literature to make this the update process more efficient and fast. The speed depends on the number of pixels, which can be very large for big images. Complexity is $\mathcal{O}(N^2)$, where N is the number of pixels.

- To prevent clustering of accidentally similar colored pixels from distant parts of the image, in practice, one can expand the pixel value vector to include the location (x, y). We have to use different values for the radius h (kernel width) for the different dimensions as they represent different things. We use h_s for the spatial dimension and h_r for the color dimension.

$$\begin{aligned} G(\mathbf{x}-\mathbf{x}_i) &= \exp\left(-\frac{||\mathbf{x}-\mathbf{x}_i||^2}{2h^2}\right) && (1.14)\\ &= \exp\left(\frac{(x-x_i)^2+(y-y_i)^2}{2h_s^2}\right) && (1.15)\\ &\quad +\exp\left(\frac{(r-r_i)^2+(g-g_i)^2++(b-b_i)^2}{2h_r^2}\right) && (1.16)\end{aligned}$$

- The choice of the parameters h_s and h_r is by trial and error. Literature survey will reveal several strategies.

Chapter 2

Linear Filtering and Edges

- Section 3.2, correlation, convolution, Gaussian smoothing, 2D filtering using 1D filters.
- Linear filtering is a basic operation done on the image for a variety of purposes, to reduce noise, sharpen features, emphasize certain kinds of features such as edges, corners, etc. See Fig 3.11 for examples.
- Most common version is the weighted sum of neighboring pixels. This is called a **correlation** given by

$$\begin{aligned} g(i,j) &= h \otimes f && (2.1)\\ &= \sum_{k=1}^{N}\sum_{l=1}^{N} h(k,l) f(i+k, j+l) && (2.2)\\ &= f \otimes h && (2.3)\\ &= \sum_{k,l} f(k,l) h(i+k, j+l) && (2.4) \end{aligned}$$

  The function $h(i,j)$ is called a filter or mask. Place the filter at a location, multiple, and then add to get the value. Note that the operation is commutative. So, equivalently shift the image, keeping the filter fixed, and then multiply and add.

- The **convolution** flips the filter about the origin and then correlates with the image function.

$$f \star h = \sum_{k,l} f(k,l)h(i-k, j-l) \tag{2.5}$$

$$h \star f = \sum_{k,l} h(k,l)f(i-k, j-l) \tag{2.6}$$

For circularly symmetric filters, convolution is the same as correlation.

- **Worksheet:** Let $h(i,j)$ be a zero valued mask with only one non-zero entry at the center equal to 1. What is the convolution of any image with this mask?

- Convolution is a linear operation: convolution of the sum of two images is the sum of the convolution of each of them. Convolution of a scaled version of a image is the scaled version of the convolution of the image.

$$h \star (af_1 + f_2) = a(h \star f_1) + (h \star f_2) \tag{2.7}$$

- Convolution is a shift invariant operation since the filter is the same for each shift.

- So, convolution is a linear, shift-invariant (LSI) operation.

- Continuous version

$$f \star h = \int_u \int_v f(u,v)h(x-u, y-v)dudv \tag{2.8}$$

$$= \int_u \int_v h(u,v)f(x-u, y-v)dudv \tag{2.9}$$

- Derivative of a convolution is the convolution.

$$\frac{\partial}{\partial x} f \star h = \int_u \int_v f(u,v) \left(\frac{\partial}{\partial x} h(x-u, y-v) \right) dudv \tag{2.10}$$

$$= \int_u \int_v f(u,v)h_x(x-u, y-v)dudv \tag{2.11}$$

- Filtering with Gaussian shaped filters. Consider just one dimensional filtering for now. Let the filter be

$$h(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp -\frac{x^2}{2\sigma^2} \tag{2.12}$$

And let the "image" be a step function $u(x)$.

$$s(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{2.13}$$

- $g(x) = s(x) \star h(x)$ will smooth the sharp edge.

$$\begin{aligned} s(x) \star h(x) &= \int_u s(u)h(x-u)du & (2.14) \\ &= \int_{u=0}^{\infty} h(x-u)du & (2.15) \\ &= \int_{v=-\infty}^{x} h(v)dv & (2.16) \end{aligned}$$

- The derivative of the output is a Gaussian function. We can use this operation, i.e. convolution with a Gaussian followed by a derivative for "edge" detection. The point of maximum marks the edge. This is equivalent to convolution with a derivative of a Gaussian function.

$$h'(x) = -\frac{x}{\sqrt{2\pi}\sigma^3} \exp -\frac{x^2}{2\sigma^2} \tag{2.17}$$

- Taken another derivative produces a zero at the edge location. This is equivalent to convolution with a second derivative of a Gaussian.

$$\begin{aligned} h''(x) &= -\frac{1}{\sqrt{2\pi}\sigma^3} \exp -\frac{x^2}{2\sigma^2} + \frac{x^2}{\sqrt{2\pi}\sigma^5} \exp -\frac{x^2}{2\sigma^2} & (2.18) \\ &= -\frac{1}{\sqrt{2\pi}\sigma^3} \left(1 - \frac{x^2}{\sigma^2}\right) \exp -\frac{x^2}{2\sigma^2} & (2.19) \end{aligned}$$

- **Worksheet:** Plot the above function.
- The 2D equivalent is the Laplacian of a Gaussian

$$\nabla h(x,y) = \frac{\partial^2}{\partial x^2} h(x,y) + \frac{\partial^2}{\partial y^2} h(x,y) \tag{2.20}$$

- 2D convolution by Gaussian is separable.

$$\begin{aligned} h(x,y) &= \frac{1}{2\pi\sigma^2}\exp -\frac{x^2+y^2}{2\sigma^2} && (2.21) \\ &= h(x)h(y) && (2.22) \end{aligned}$$

$$f(x,y) \star h(x,y) = (f(x,y) \star h(x)) \star h(y) \qquad (2.23)$$

Chapter 3

Point Features

- Figure 4.2 and 4.3
- Point features are stable with respect to viewpoint and lighting.
- To track points across video frames possible matching criterion is weighted image difference.

$$E(\mathbf{u}) = \sum_i w(\mathbf{x}_i)\left(I_1(\mathbf{x}_i + \mathbf{u}) - I_0(\mathbf{x}_i)\right)^2 \tag{3.1}$$

- The value of this would be low when regions match, but could be low for other reason too, such as dark patches.
- How can one select points to track? Find points in image such as the error criterion is stable with respect to small variations, on the same image. (see Fig 4.5)

$$E_{AC}(\triangle\mathbf{u}) = \sum_i w(\mathbf{x}_i)\left(I_0(\mathbf{x}_i + \triangle\mathbf{u}) - I_0(\mathbf{x}_i)\right)^2 \tag{3.2}$$

- Using Taylor series expansion

$$\begin{aligned} I_0(\mathbf{x}_i + \triangle\mathbf{u}) &\approx I_0(\mathbf{x}_i) + \triangledown I_0(\mathbf{x}_i)^T \Delta\mathbf{u} && (3.3)\\ &= I_0(\mathbf{x}_i) + \begin{bmatrix} I_x(\mathbf{x}_i) & I_y(\mathbf{x}_i) \end{bmatrix} \begin{bmatrix} \triangle x \\ \triangle y \end{bmatrix} && (3.4) \end{aligned}$$

- The error criterion on the same image can be approximated using the Taylor expansion above.

$$\begin{aligned}
E_{AC}(\triangle\mathbf{u}) &= \sum_i w(\mathbf{x}_i)\left(I_0(\mathbf{x}_i+\triangle\mathbf{u})-I_0(\mathbf{x}_i)\right)^2 && (3.5)\\
&\approx \sum_i w(\mathbf{x}_i)\left(\bigtriangledown I_0(\mathbf{x}_i)^T\Delta\mathbf{u}\right)^2 && (3.6)\\
&= \sum_i w(\mathbf{x}_i)\left(\bigtriangledown I_0(\mathbf{x}_i)^T\Delta\mathbf{u}\right)^T\left(\bigtriangledown I_0(\mathbf{x}_i)^T\Delta\mathbf{u}\right) && (3.7)\\
&= \sum_i w(\mathbf{x}_i)\begin{bmatrix}\triangle x & \triangle y\end{bmatrix}\begin{bmatrix}I_x(\mathbf{x}_i)\\ I_y(\mathbf{x}_i)\end{bmatrix}\begin{bmatrix}I_x(\mathbf{x}_i) & I_y(\mathbf{x}_i)\end{bmatrix}\begin{bmatrix}\triangle x\\ \triangle y\end{bmatrix} && (3.8)\\
&= \begin{bmatrix}\triangle x & \triangle y\end{bmatrix}\left(\sum_i w(\mathbf{x}_i)\begin{bmatrix}I_x(\mathbf{x}_i)\\ I_y(\mathbf{x}_i)\end{bmatrix}\begin{bmatrix}I_x(\mathbf{x}_i) & I_y(\mathbf{x}_i)\end{bmatrix}\right)\begin{bmatrix}\triangle x\\ \triangle y\end{bmatrix} && (3.9)\\
&= \begin{bmatrix}\triangle x & \triangle y\end{bmatrix}\mathbf{A}\begin{bmatrix}\triangle x\\ \triangle y\end{bmatrix} && (3.10)
\end{aligned}$$

$$\mathbf{A} = \begin{bmatrix}\sum_i w(\mathbf{x}_i)I_x^2(\mathbf{x}_i) & \sum_i w(\mathbf{x}_i)I_x(\mathbf{x}_i)I_y(\mathbf{x}_i)\\ \sum_i w(\mathbf{x}_i)I_y(\mathbf{x}_i)I_x(\mathbf{x}_i) & \sum_i w(\mathbf{x}_i)I_y^2(\mathbf{x}_i)\end{bmatrix} \tag{3.11}$$

$$\mathbf{A} = \begin{bmatrix}\sum_i w(\mathbf{x}_i)I_x^2 & \sum_i w(\mathbf{x}_i)I_xI_y\\ \sum_i w(\mathbf{x}_i)I_yI_x & \sum_i w(\mathbf{x}_i)I_y^2\end{bmatrix} \tag{3.12}$$

$$\mathbf{A} = \sum_i w(\mathbf{x}_i)\begin{bmatrix}I_x^2 & I_xI_y\\ I_yI_x & I_y^2\end{bmatrix} \tag{3.13}$$

- The matrix **A** is an estimate of the local image tensor.
- $\mathbf{A}^{-1}$ is a measure of the uncertainty of the location, i.e. how sharp the valley is?
- If λ_0 and λ_1 are eigenvalues of A, where $\lambda_0 < \lambda_1$, then λ_0^{-1} and λ_1^{-1} are the eigenvalues of $\mathbf{A}^{-1}$. The largest eigenvalue of $\mathbf{A}^{-1}$ should be small, i.e. λ_0^{-1} should be small, or in other words λ_0 should be high (Shi and Tomasi).
- $\lambda_0\lambda_1 = \det(\mathbf{A})$ and $\lambda_0+\lambda_1 = Trace(\mathbf{A})$

- For Harris points: $\lambda_0\lambda_1 - \alpha(\lambda_0+\lambda_1)^2$ should be high, typically $\alpha = 0.06$
- Others suggest the harmonic mean, $\frac{\lambda_0\lambda_1}{\lambda_0+\lambda_1}$
- Feature tracking - Section 4.1.4. We expect motion from frame to frame is small.
- Detect stable features in one frame.
- Correlate patches over small neighborhoods and pick point of smallest squared difference.
- If features are tracked over long sequences, appearance can change, so need to update template, but not too often.

Chapter 4

Kalman Tracking

- Consider the task of tracking moving stuff in videos taken from fixed cameras. This is the simplest imaging model we can consider for this problem. Whatever is developed here can be generalized and formulated for other complex cases such as moving cameras with moving objects. Of course, the equations in the world will be more complicated.

- We are going to differentiate between the measurement we make about the moving objects and the state of the moving objects. The states of the moving objects are not directly observed. We can make measurements about some of the components of the states of the moving objects.

- Let $\mathbf{s}_t$ denote the state of the moving objects at time t. Possible choices of the state vector are: just location, $\begin{bmatrix} x & y \end{bmatrix}^T$ or location and velocity $\begin{bmatrix} x & y & v_x & v_y \end{bmatrix}^T$ or location, velocity, and bounding box size $\begin{bmatrix} x & y & w & h & v_x & v_y \end{bmatrix}^T$ or location, velocity, acceleration, and bounding box size $\begin{bmatrix} x & y & w & h & v_x & v_y & a_x & a_y \end{bmatrix}^T$

- This state is really a random variable as we do not know the exact value. We assume it is a Gaussian random variable, with a mean value, $\hat{\mathbf{s}}_t$, and a covariance $\mathbf{P}_t$ capturing the uncertainty about the state.

- A measurement model relates the measurement with the underlying model states. In general the dimension of the measurement vector would be lower than the state vector. For instance, we can measure the location directly, but not the velocity or acceleration; so the measurement vector in that case will consist of just the location values.

Let the measurement vector be $\mathbf{m}_t$ and the relationship with the state vector $\mathbf{s}_t$ be related by

$$\mathbf{m}_t = \mathbf{H}\mathbf{s}_t + \mathbf{w} \tag{4.1}$$

where $\mathbf{w}$ is the measurement error, which again will be assume to zero mean with some covariance, $\mathbf{R}$, capturing the quality of the measuring modality. For instance, if you are measuring the location sum of squared differences between patches from earlier point features, you can use the inverse of the local image tensor (Eq. 3.13) as a measure of uncertainty.

One example of a measurement model is given by.

$$\begin{bmatrix} r \\ c \end{bmatrix}_{t+1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix}_t + \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} \tag{4.2}$$

- We assume a linear model of the state change.

$$\mathbf{s}_{t+1} = \mathbf{A}\mathbf{s}_t + \mathbf{n} \tag{4.3}$$

where $\mathbf{n}$ is zero mean noise with covariance $\mathbf{Q}$. Note that $\mathbf{s}_{t+1}$ is linear combination of two Gaussian random variable so it is also a Gaussian random variable. One example of a constant velocity model is given by

$$\begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix}_{t+1} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix}_t + \begin{bmatrix} n_0 \\ n_1 \\ n_2 \\ n_3 \end{bmatrix} \tag{4.4}$$

- At each time instant, t, the following variables are updated: the mean value of the state $\hat{\mathbf{s}}_t$ and its covariance $\mathbf{P}$. These quantities are effected by the state evolution function and also indirectly by the measurement. So, the update is in two steps.

- **Forward Update:** Update based on the state equation. Note the noise is zero measure and remember the addition of Gaussian random variables is also a Gaussian. Let $\mathbf{Y} = \mathbf{X}_1 + \mathbf{X}_2$, where each of them is Gaussian random variable. Then $\widehat{\mathbf{Y}} = \widehat{\mathbf{X}_1} + \widehat{\mathbf{X}_2}$ and $\mathbf{\Sigma}_Y = \mathbf{\Sigma}_{X_1} + \mathbf{\Sigma}_{X_2}$.

  The intermediate estimate of the new state and its covariance, just based on the forward model is given by

$$\begin{aligned} \hat{\mathbf{s}}^-_{t+1} &= \mathbf{A}\hat{\mathbf{s}}_t && (4.5) \\ \mathbf{P}^-_{t+1} &= \mathbf{A}\mathbf{P}_t\mathbf{A}^T + \mathbf{Q} && (4.6) \end{aligned}$$

- **Kalman Gain:** blends the model covariance and measurement error covariance

$$\mathbf{K} = \mathbf{P}^-_{t+1}\mathbf{H}^T\left(\mathbf{H}\mathbf{P}^-_{t+1}\mathbf{H}^T + \mathbf{R}\right)^{-1} \tag{4.7}$$

- **Measurement Update:**

$$\begin{aligned} \hat{\mathbf{s}}_{t+1} &= \hat{\mathbf{s}}^-_{t+1} + \mathbf{K}\left(\mathbf{m}_{t+1} - \mathbf{H}\hat{\mathbf{s}}^-_{t+1}\right) && (4.8) \\ \mathbf{P}_{t+1} &= \left(\mathbf{I} - \mathbf{K}\mathbf{H}\right)\mathbf{P}^-_{t+1} && (4.9) \end{aligned}$$

  Note how the measurement error, i.e. the terms in parenthesis, is added to the state to update it.

- Work through the following example with constant velocity model example shown earlier, with the following values of the other variables you would need.

  We assume half a pixel model error noise, which implies that we have faith in the model.

$$\mathbf{Q} = \begin{bmatrix} \frac{1}{4} & & 0 & \\ & \frac{1}{4} & & \\ & & \frac{1}{4} & \\ & 0 & & \frac{1}{4} \end{bmatrix} \tag{4.10}$$

  Initial estimate of the covariance model assumes that it is diagonal, uncorrelated dimensions, with 3 pixel standard deviation in location and 5 pixel standard deviation in velocity.

$$\mathbf{P}_0 = \begin{bmatrix} 9 & & 0 & \\ & 9 & & \\ & & 25 & \\ & 0 & & 25 \end{bmatrix} \tag{4.11}$$

The measurement error is diagonal too with a standard deviation of 1 pixel.

$$\mathbf{R} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.12}$$

Let $\hat{\mathbf{s}}_0 = \begin{bmatrix} 100 \\ 170 \\ 0 \\ 0 \end{bmatrix}$ and measurement be $\mathbf{m}_1 = \begin{bmatrix} 103 \\ 163 \end{bmatrix}$, what is $\mathbf{s}_1$?

$$\hat{\mathbf{s}}_1^- = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 100 \\ 170 \\ 0 \\ 0 \end{bmatrix} \tag{4.13}$$

$$= \begin{bmatrix} 100 \\ 170 \\ 0 \\ 0 \end{bmatrix} \tag{4.14}$$

$$\mathbf{P}_1^- = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 9 & & \mathbf{0} & \\ & 9 & & \\ & & 25 & \\ & \mathbf{0} & & 25 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} + \begin{bmatrix} \frac{1}{4} & & \mathbf{0} & \\ & \frac{1}{4} & & \\ & & \frac{1}{4} & \\ & \mathbf{0} & & \frac{1}{4} \end{bmatrix} \tag{4.15}$$

$$= \begin{bmatrix} 34.25 & 0 & 25 & 0 \\ 0 & 34.25 & 0 & 25 \\ 25 & 0 & 25.25 & 0 \\ 0 & 2 & 0 & 25.25 \end{bmatrix} \tag{4.16}$$

This state prediction gives us a location to relocate the new location and the covariance estimate gives us an area to consider. In our case this is a circle centered at $(100, 170)$ and with radius $\sqrt{34.25} = 5.85$ pixels. We can run the SSD detector and make a new measurement. $\mathbf{m}_1 = \begin{bmatrix} 103 \\ 163 \end{bmatrix}$