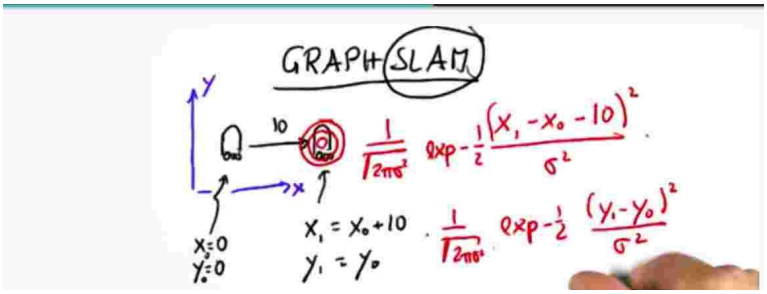
GRAPH SLAM

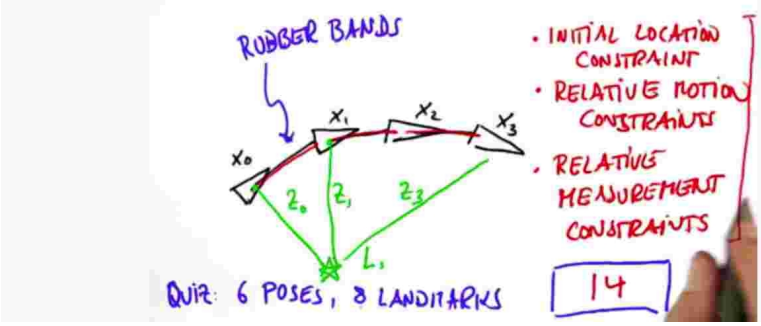
Source: Udacity AI course
Quick notes by P.Phairatt 7.12.16

Basic idea



Assume we have a robot moving and observing the landmarks with a Gaussian distribution. The product of these Gaussian distributions is presented as a constraint in which we want to maximise the likelihood of X1 given X0 (for a motion).

Graph Slam is the approach to define the probability using the sequence of constraints. Following examples demonstrate the method of Graph Slam to update the constraints of each position and landmark in the environment.

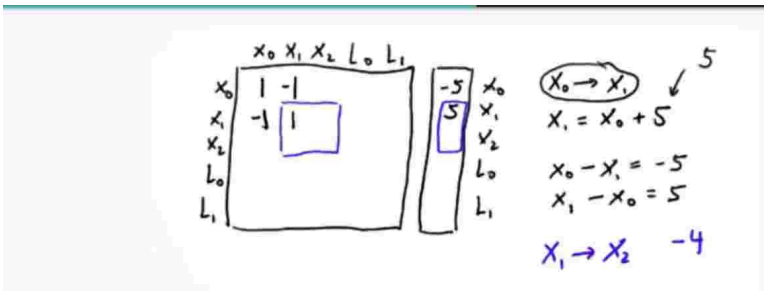


Let say, there are 5 motions from X0 and 8 measurements (to L1)

5 motions (result in 6 poses), and 8 landmark measurements = 14 constraints

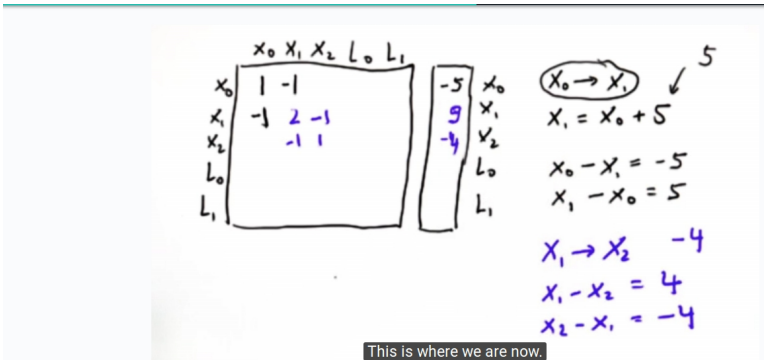
Information Matrix (Omega) and Vector (Si):

Updating position constraints



from X0 → X1

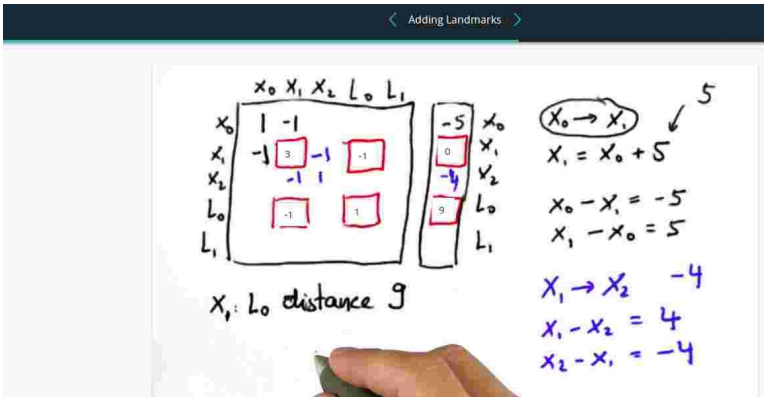
given x1 = x0 +5
x0-x1 = -5
-x0+x1 = 5
therefore
1 -1 -5
-1 1 5



from X1 → X2
given x2 = x1-4
x1 -x2 = 4
-x1+x2 = -4
therefore
1 -1 4
-1 1 -4

update	X0	X1	X2	Si
X0	1	-1		-5
X1	-1	1+1	-1	5+4
X2	-1	1		-4

Update landmark constraints

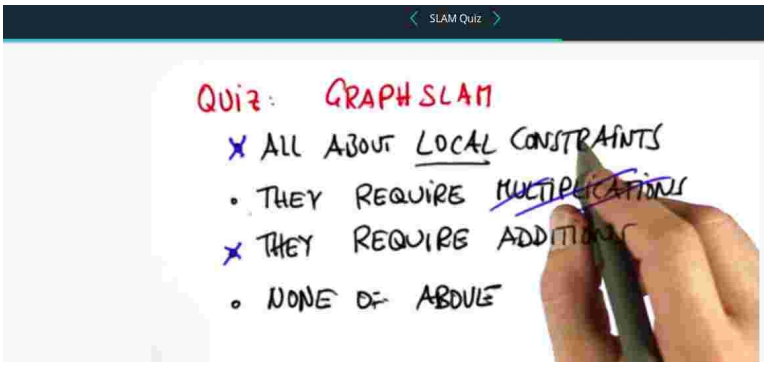


between X1 and L0
given landmark L0 = x1 + 9
x1 – Lo = -9
-x1 + Lo = 9
therefore,
1 -1 -9
-1 1 9
update

	X0	X1	X2	L0	Si
X0	1	-1			-5
X1	-1	2+1	-1	-1	9-9
X2		-1	1		-4
L0		-1		1	9

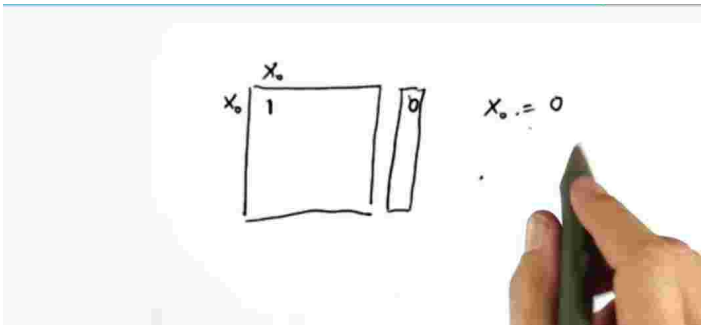
This information matrix and vector will later be called as Omega and Si.

Graph Slam manipulation



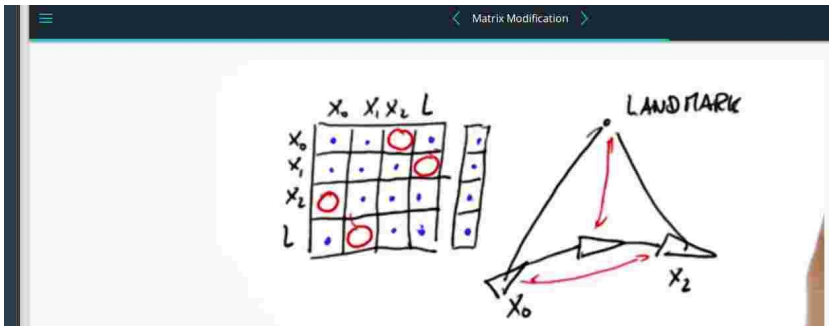
Graph Slam manipulation is just a direct local constraint between motions/landmarks and additions

Initial constraint update



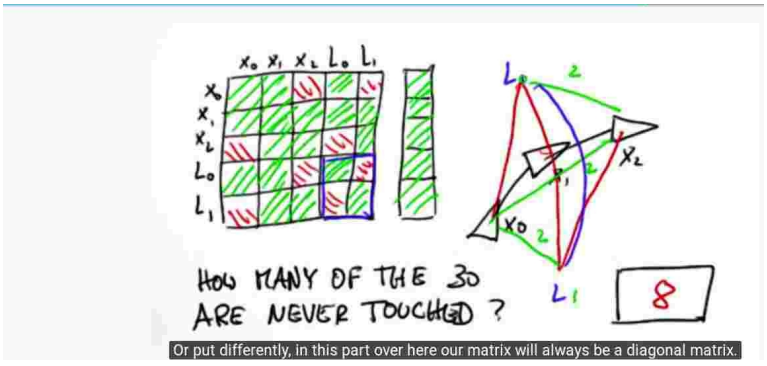
Initial constraint at X0, given initial X0 = 0, the information matrix and vector will be initialised with 1 and 0.

No constraint relationship



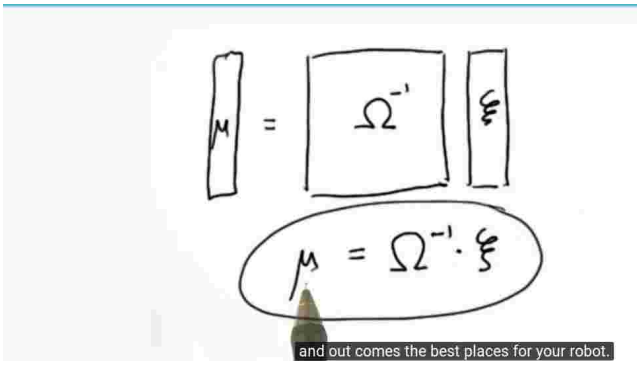
Red lines show the no direct constraint among the positions and the landmark in Graph Slam that result in no update in the matrix and vector (0).

No constraint relationship: more example

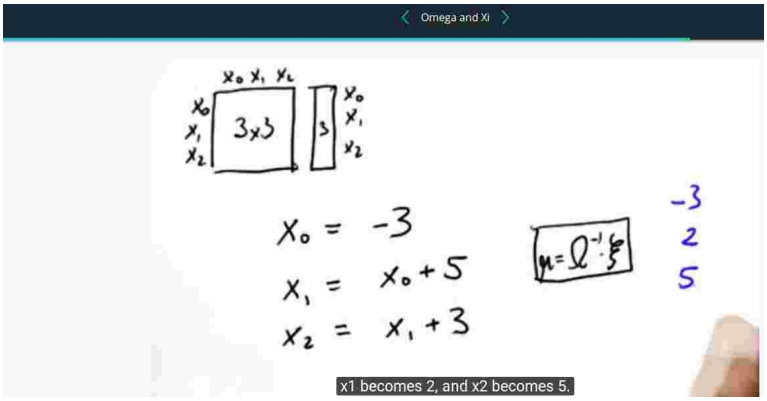


red/black lines = constrained lines
green/blue lines = no constrained lines
1 pairs = 2 constraints missing
There always no constraints between landmarks (e.g. L0 ↔ L1). It is therefore always a diagonal matrix.

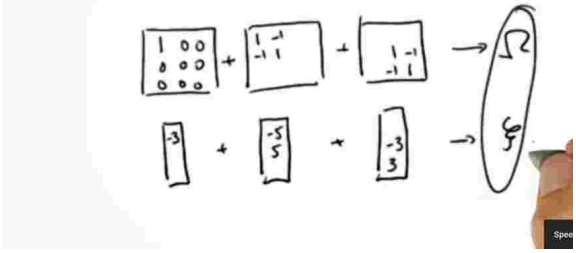
Graph Slam Formula



Information matrix (Omega), Information vector (Si), State (u - position and landmark positions)



Given initial position and constraints. The equation above should give the state u = [-3, 2, 5]

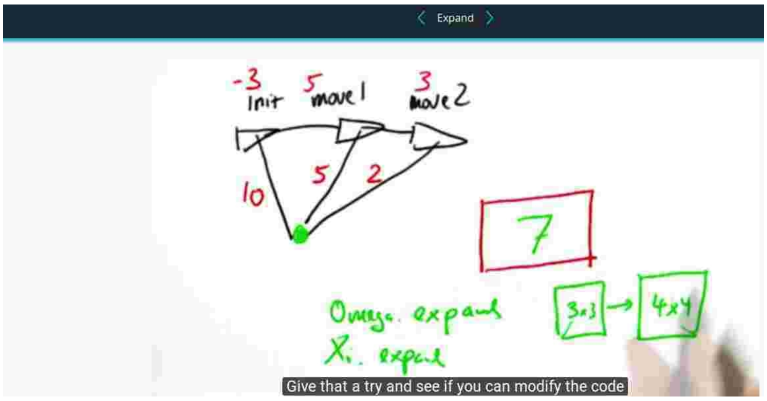


Update motions accordingly,

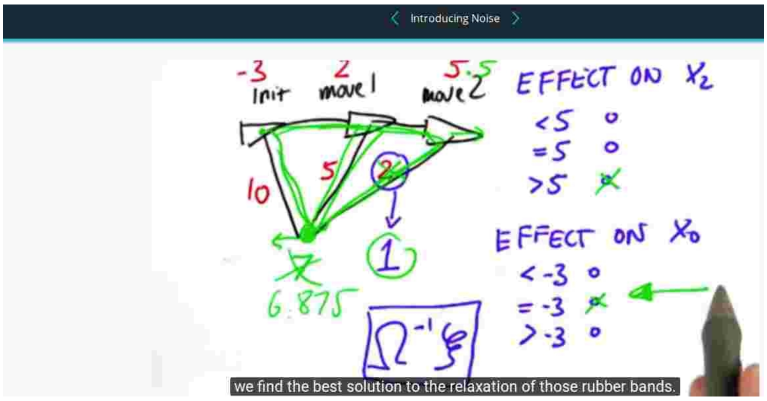
u = Omega_Inverse * Si = [-3, 2, 5]

Example with 1 landmark constraint and no noise

It is clear that the Lo = 7 (with no measurement noise).
X1 = X0 + 5, X2 = X1 + 3
L0 = X0+10, X1+5, X2+2 where X0 = -3, X1 = 2, X2 = 5



Example with noise

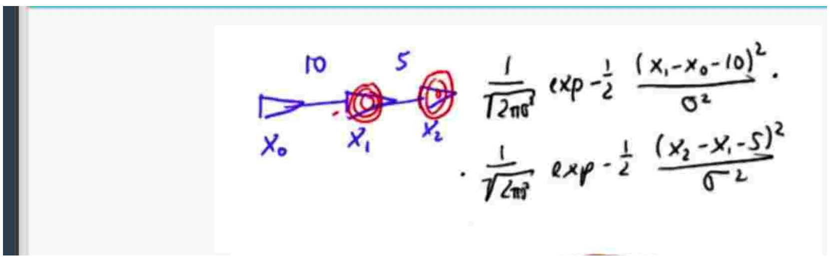


Let add noise to the last measurement constrained to X2! Say the measurement is now 1 rather than 2 (noise free case).
What will happen to L0, X0 and X1 (comparing with noise free case)?

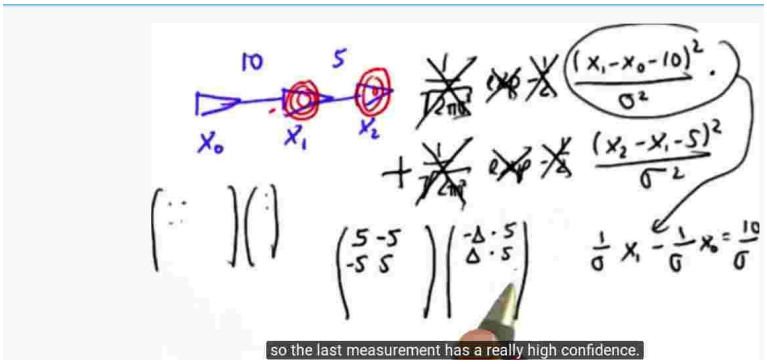
Imagine the rubber band between X2 ↔ L0, it has been pull together tighter by the measurement as the result
- the 'distance' between X2 and L0 will be reduced (pull closer) by **'increasing X2 and reducing L0'**
  (L0 – X2 = 2 noise free, L0 – X2 = 1 with noise)
- X1 has been pull up as well to maximise the likelihood
- X0 is fixed and never been affected

```
 5.0                                   7.0
 |.................................................|          noise free
X2                                     L0
      5.5                   6.88
       |..........................|                 with noise confidence scaling= 1
      X2                   L0
          5.7        6.82
          |...............|                         with noise confidence scaling= 5 (more confident in the measurement, see next)
      X2                   L0
```

Confident Measurement

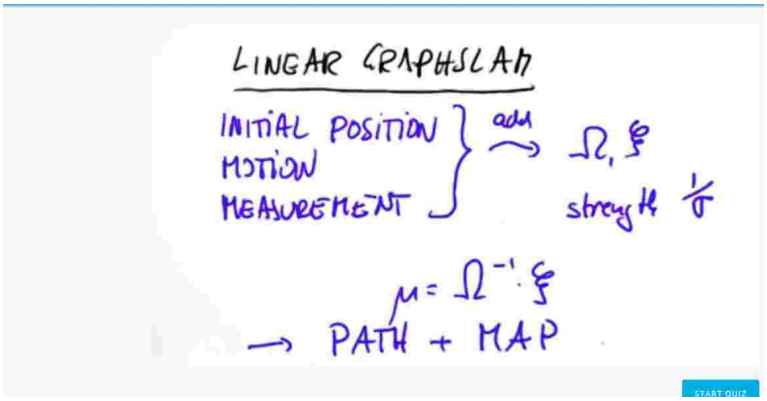


Total probability of the system is the product of these two constraints that we want to maximise.



The maximisation trick is the sum of these term scaling with the confidence level as depicted in the picture above in one motion step.
In the Graph Slam, it is simply multiples with the scaling number. Considering only one term (between X0 and X1), the scaling factor is proportional to the variance (low variance, high confidence → larger scaling factor)

Put it all together



Given initial position, then motion, then landmark update for each navigation with confident strength,
so we can build the estimate path the robot takes.

Note on the Graph Slam approach
Method 1- Separate X,Y vector (Si), sharing the same matrix (Omega)
Method 2- Combine X,Y state
full omega matrix dim = 2*(N motion + m landmarks) → motion include initial, 2x for X, Y Coordinate

Separate- matrix N+M

P0 P1 P2 L0 L1
P0
P1
P2 OMEGA
L0
L1

X0
X1
X2 SI_X
Lx0
Lx1

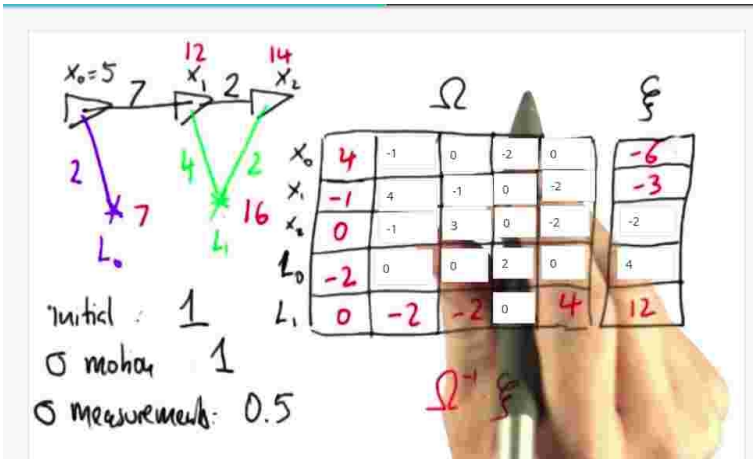
Y0
Y1
Y2 SI_Y
Ly0
Ly1

Combine- matrix 2*(N+M)

	X0	Y0	X1	Y1	X2	Y2	Lx0	Ly0	Lx1	Ly1
X0										
Y0										
X1										
Y1										
X2										
Y2						OMEGA				
Lx0										
Ly0										
Lx1										
Ly1										

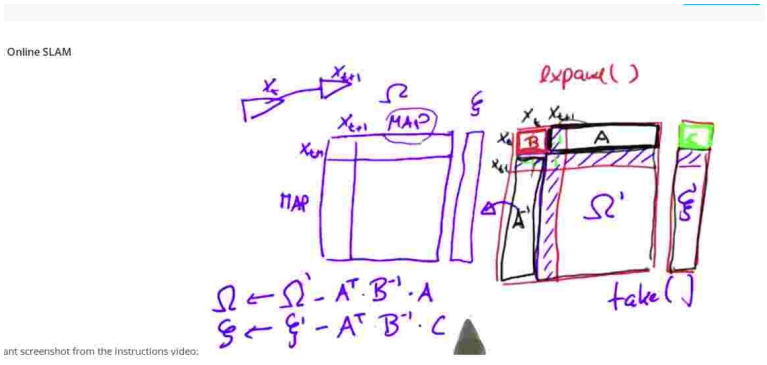
	X0	Y0	X1	Y1	X2	Y2	Lx0	Ly0	Lx1	Ly1
X0										
Y0										
X1										
Y1										
X2										
Y2						SI				
Lx0										
Ly0										
Lx1										
Ly1										

QUIZ Filling full Graph Slam matrix



# Motion constraints
X1 = X0 + 7
X2 = X1 + 2
# Landmark constraints
L0 = X0 + 2
L1 = X1 + 4
L1 = X2 + 2

Online GraphSLAM- to make it scalable



ant screenshot from the instructions video:

Scaling to the recent position by only keeping track of the recent robot position after the motion and measurement update. Add the motion/measurement update as the full slam, then summarise to the recent state by the above formula.