

## SLAM

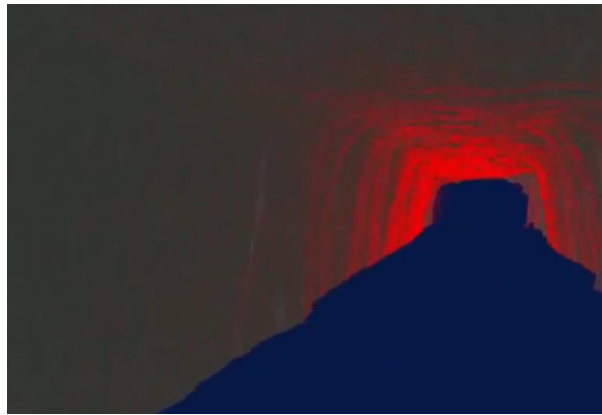
SLAM is Method for mapping.

It`s short for simultaneous localization and mapping.

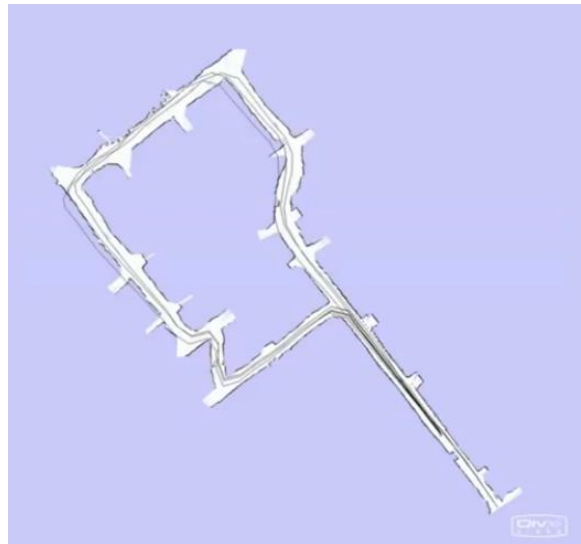
Mapping is all about building maps of the environment.

You might remember in the localization classes; we assume the map of the given.

One of my big passions in my life has been to understand how to make a robot make these maps like this map over here, which is a 3D map of an abandoned underground coal mine in Pennsylvania near Carnegie Mellon University.



Different method for building maps that are quite sophisticated Like this particular pillar method over here that you can see.



All these methods have in common that we build a model of the environment while also addressing the fact that the robot itself accrues uncertainty valid moves.

When in this example here, **the loop is being closed**.

You can see how our mapping technology is able to accommodate this and find a consistent map despite the fact that the robot drifted along the way.

So the key insights in building maps is that the robot itself might lose track of where it is by virtue of its **motion uncertainty**.

You **accommodated the in localizations** by using existing map but now we don't have an existing map.

We're building the map.

That's why SLAM comes to play SLAM doesn't stand for slamming a robot.

What it really means is **simultaneous localizations and mapping**.

**Graph SLAM:** which also is by far the easiest method to understand we will reduce the mapping problem to a couple of very **intuitive additions** into a big matrix and a vector and that's it.

## Simultaneous Localization and Mapping

*In the previous lessons, you learned all about localization methods that aim to locate a robot or car in an environment given sensor readings and motion data, and we would start out knowing close to nothing about the surrounding environment.*

***In practice, in addition to localization, we also want to build up a model of the robot's environment so that we have an idea of objects, and landmarks that surround it and so that we can use this map data to ensure that we are on the right path as the robot moves through the world!***

*In this lesson, you'll learn how to **implement SLAM (Simultaneous Localization and Mapping) for a 2-dimensional world!** You'll combine what you know about*

- 1- robot sensor measurements and movement to create locate a robot and 2- create a map of an environment from only sensor and motion data gathered by a robot, over time. SLAM gives you a way to track the location of a robot in the world in real-time and identify the locations of landmarks such as
  - a. 1-buildings
  - b. Trees
  - c. rocks,
  - d. and other world features.

### Graph SLAM

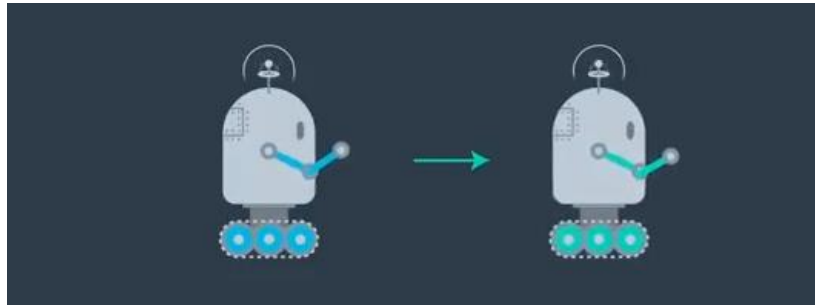
*This is one method for slam and the abundance by far the easiest to explain.*

*Let`s assume we have a robot and it`s called arbitrarily location, for this example, we just assume the board has a perfect compass, and we don`t care about heading direction just to keep things simple.*



## SIMULTANEOUS LOCALIZATION AND MAPPING

Let's assume the robot moves to the right in x direction by 10.

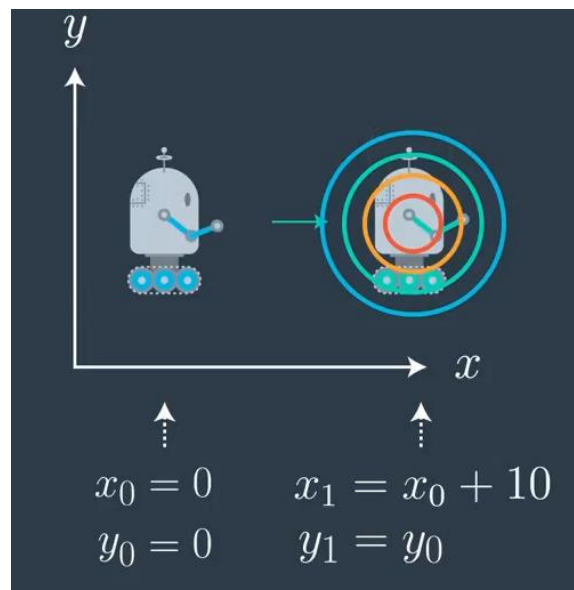


In a perfect world, you would know that  $x_1$ , the location after motion  $x_1 = x_0 + 10$

And  $y_1$  is the same as  $y_0$ .

But we learn the location is actually uncertain.

So, rather than assuming in our X-Y coordinate system the robot moved to the right by 10 exactly we know that the actual location is a Gaussian centered around 10, 0 but it's possible that over to somewhere else.



Remember, we worked out the math for this Gaussian.

Here's how it looks just for the x variable. Rather than setting  $x_1$  to  $x_0 + 10$  we tried to express the Gaussian that peaks when these two things are the same.

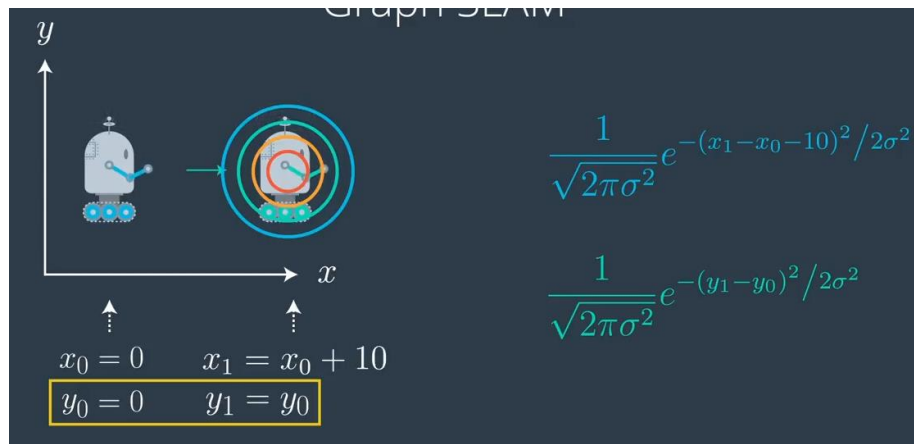
So, if you subtract from  $x_1, x_0$  and 10, put the into a square format, and turn this into gaussian.

We get a probability distribution that relates  $x_1$  and  $x_0$ .

$$\frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x_1 - x_0 - 10)^2 / 2\sigma^2}$$

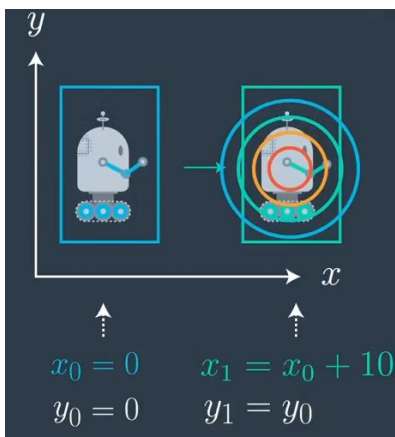
When can do the same for y.

Since there is no change in y according to our motion, all the assert that  $y_1$  and  $y_0$  are as close together as possible.



The product of these two Gaussian is now our constraint

We wish to maximize the likelihood of the position  $x_1$  given that the position  $x_0$  is zero



So the Graph slam does is defining our probabilities using a sequence of such constraints.

Say we have a robot that moves in some spaces, and each location is now characterized by a vector  $x_0$  and a vector  $x_1, x_2, x_3$  and  $x_4$ .

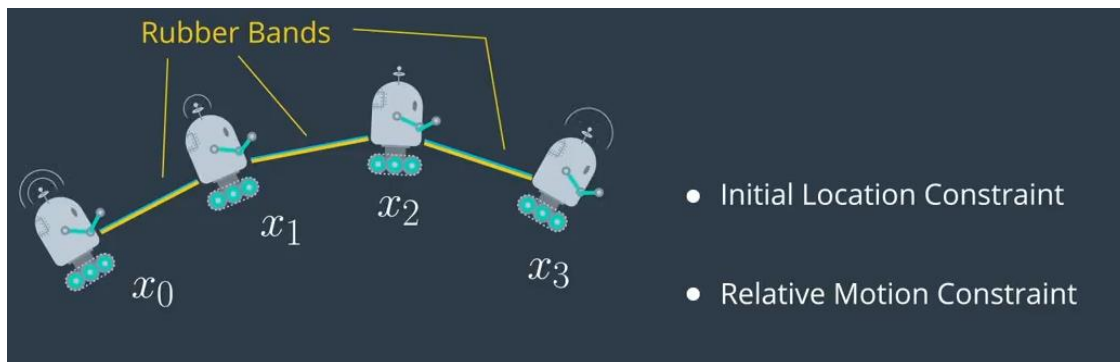
Often, they are three-dimension vectors, then we will graph some collects is initial location, which is a zero, zero, zero, usually.

Over is a here it is a different.

Then really importantly, the lots if relative constraints that relate each robot pose to the previous world pose. We call them relative motion constraints.

You can think of those as rubber bands.

In expectation, this rubber band will be exactly the motion the robot sensed or commanded.



But in reality, it might have to bend a little bit to make the map more consistent.

Speaking about maps, there's huge landmarks as an example.

So, Suppose there's a landmark out here, and the landmark is being seen from the robot with some relative measurement  $z_0, z_1, z_3$ .

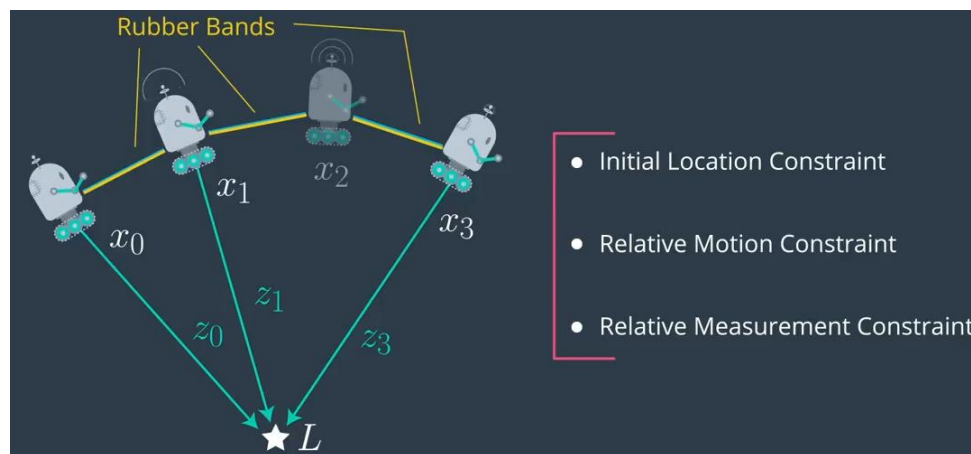
All these are also relative constraints, very much like the one before.

Again, the capture by Gaussians, and we get relative measurement constraints.

**One such constraint every time the robot sees a landmark.**

So, Graph Slam collects those constraints.

As well see, they are insanely easy to collect, and it just relax this the set of rubber bands to find the most likely configuration of robot path along with the location of landmarks, and that is the mapping process.



Suppose we have six robot poses that is one initial, and five motions and we have eight measurement of landmarks that we've seen these might be multiple landmarks and some doesn't move it so more than one.

How many total constraints do we have if we count each of these constraints for exactly one constraint?

For a series of robot motions and landmark locations, how many constraints will there be for 6 total poses and 8 landmarks?

- ☐ 12
- ☒ 14
- ☐ 32
- ☐ 48

## Constraints

For our 6 poses, we have:

- 1 initial location constraint
- 5 additional, relative motion constraints, and finally,
- 8 relative measurement constraints for the 8 landmark locations

Adding all of these up gives us a total of **14** constraints.

Now, consider the image above, with 4 poses (including the initial location  $x_0$ ) and one landmark. We can use this same math and say that there are 5 total constraints for the given image, but in practice there are usually many more measurements and motions!

### Implementing Constraints

You also may have noticed that not all of these constraints will provide us with meaningful information, such as in our example: we do not have a measurement between the pose  $x_3$  and the landmark location. Next, let's see how we can represent these constraints in a **matrix** and their relationships with values in that matrix and a constraint **vector**.

### Implementing Constraints Solution V1

We label matrix, which is quadric with all the poses and all the landmarks.

Here we assume the landmarks are distinguishable and every time we make an observation, say between **two poses**, they become little addition locally in the four elements in the matrix defined over those poses.

	$x_0$	$x_1$	$x_2$	$L_0$	$L_1$		
$x_0$							$x_0$
$x_1$							$x_1$
$x_2$							$x_2$
$L_0$							$L_0$
$L_1$							$L_1$



For example, if the robot moves from  $x_0$  to  $x_1$ .

We therefore believe  $x_1$  should be the same as  $x_0$  say + 5.

The way we enter this into the matrix is in two ways.

1 =>  $x_0$

-1 =>  $x_1$

Add together should be -5

	$x_0$	$x_1$	$x_2$	$L_0$	$L_1$		
$x_0$	1	-1				-5	$x_0$
$x_1$							$x_1$
$x_2$							$x_2$
$L_0$							$L_0$
$L_1$							$L_1$

$x_0 \rightarrow x_1$   
 $x_1 = x_0 + 5$

We look at the equation here  $x_0 - x_1$  equals -5

These are added into the matrix that starts with 0 everywhere and it's a constraint that relates  $x_0$ ,  $x_1$  by -5 , it's that simple.

Secondly, we do the same with  $x_1$  as positive so we add one over here and for that,  $x_1 - x_0 = 5$  , you put 5 over here and -1 over here.

	$x_0$	$x_1$	$x_2$	$L_0$	$L_1$		
$x_0$	1	-1				-5	$x_0$
$x_1$	-1	1				5	$x_1$
$x_2$							$x_2$
$L_0$							$L_0$
$L_1$							$L_1$

$x_0 \rightarrow x_1$   
 $x_1 = x_0 + 5$   
 $x_0 - x_1 = -5$   
 $x_1 - x_0 = 5$

Put differently the motion constraint that relates  $x_0$  to  $x_1$  by the motion 5 has modified incrementally by adding values, the matrix for all elements that fall between  $x_0$  and  $x_1$ .

We basically wrote that constraints twice in both cases we make sure that the diagonal element was positive and the viewer the correspondents off-diagonal element as a negative value.

We added the corresponding value on the right side.

	$x_0$	$x_1$	$x_2$	$L_0$	$L_1$			
$x_0$	1	-1				-5	$x_0$	$x_0 \rightsquigarrow x_1$ $x_1 = x_0 + 5$
$x_1$	-1	1				5	$x_1$	$x_0 - x_1 = -5$ $x_1 - x_0 = 5$
$x_2$							$x_2$	
$L_0$							$L_0$	
$L_1$							$L_1$	

Quiz:

Suppose we know we go from  $x_1$  to  $x_2$  and whereas the motion over here was +5.

Not it's -4 so we moving back in the opposite direction

What will be the new value as for the matrix over here?

Note : They only affects values that occur in the region between  $x_1, x_2$  and over here and remember these are additive.

	$x_0$	$x_1$	$x_2$	$L_0$	$L_1$			
$x_0$	1	-1				-5	$x_0$	$x_0 \rightsquigarrow x_1$ $x_1 = x_0 + 5$
$x_1$	-1	1				5	$x_1$	$x_0 - x_1 = -5$
$x_2$							$x_2$	$x_1 - x_0 = 5$
$L_0$							$L_0$	$x_1 \rightarrow x_2 - 4$
$L_1$							$L_1$	

### Constraint Matrices

Next, you'll see how to implement relationships between robot poses and landmark locations. **These matrices should look familiar from the section of linear algebra**, but I also find it helpful to think of the values in these matrices as **weights** kind of like you've seen in **convolutional kernels**, only these weights imply how much a pose or landmark should be weighted in a set of equations.

For the highlighted values in the square above, what will the new weights relating  $x_1$  and  $x_2$  be after this motion update? They start as:

$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

Which can also be read as:  $\begin{bmatrix} 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \end{bmatrix}$

☐

$$\begin{bmatrix} 0 & 1 \end{bmatrix}, \begin{bmatrix} -1 & 1 \end{bmatrix}$$

☐

$$\begin{bmatrix} 1 & -1 \end{bmatrix}, \begin{bmatrix} 0 & 0 \end{bmatrix}$$

☐

$$\begin{bmatrix} 2 & -1 \end{bmatrix}, \begin{bmatrix} -1 & 1 \end{bmatrix}$$

☐

$$\begin{bmatrix} 2 & 1 \end{bmatrix}, \begin{bmatrix} -1 & 1 \end{bmatrix}$$

☐

$$\begin{bmatrix} 2 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 0 \end{bmatrix}$$

☐

$$\begin{bmatrix} 4 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 2 & -1 \end{bmatrix}, \begin{bmatrix} -1 & 1 \end{bmatrix}$$

For the highlighted values in the small rectangle above, what will the new values relating  $x_1$  and  $x_2$  be after this motion update? They start as:

$\begin{bmatrix} 5 \\ 0 \end{bmatrix}$

Which can also be read as:  $\begin{bmatrix} 5 \end{bmatrix}, \begin{bmatrix} 0 \end{bmatrix}$

☐

$\begin{bmatrix} 0 \end{bmatrix}, \begin{bmatrix} 5 \end{bmatrix}$

☐

$\begin{bmatrix} 1 \end{bmatrix}, \begin{bmatrix} 4 \end{bmatrix}$

☐

$\begin{bmatrix} 1 \end{bmatrix}, \begin{bmatrix} -4 \end{bmatrix}$

☐

$\begin{bmatrix} 9 \end{bmatrix}, \begin{bmatrix} 5 \end{bmatrix}$

☐

$\begin{bmatrix} 9 \end{bmatrix}, \begin{bmatrix} -4 \end{bmatrix}$

$\begin{bmatrix} 9 \end{bmatrix}, \begin{bmatrix} -4 \end{bmatrix}$

**Solution**

$$x_1 - x_2 = 4$$

$$\text{and } x_2 - x_1 = -4$$

So I have add + 1 over here , -1 over here + 1 in this diagonal element over here -1  
let me just do this

$$\begin{aligned} x_1 - x_2 &= 4 \\ x_2 - x_1 &= -4 \end{aligned}$$

These number L , the end transformer the first number over here to 2 we get a -1 for the off-diagonal elements and 1 over here,

	$x_0$	$x_1$	$x_2$	$L_0$	$L_1$
$x_0$	1	-1			
$x_1$	-1	2	-1		
$x_2$		-1	1		
$L_0$					
$L_1$					

-5	$x_0$	$x_1 = x_0 + 5$
5	$x_1$	$x_0 - x_1 = -5$
	$x_2$	$x_1 - x_0 = 5$
	$L_0$	$x_1 \rightarrow x_2 - 4$
	$L_1$	$x_1 - x_2 = 4$ $x_2 - x_1 = -4$

Now we add 4 to 5 , which give us 9 and -1 to 4 to 0 gives -4

	$x_0$	$x_1$	$x_2$	$L_0$	$L_1$
$x_0$	1	-1			
$x_1$	-1	2	-1		
$x_2$		-1	1		
$L_0$					
$L_1$					

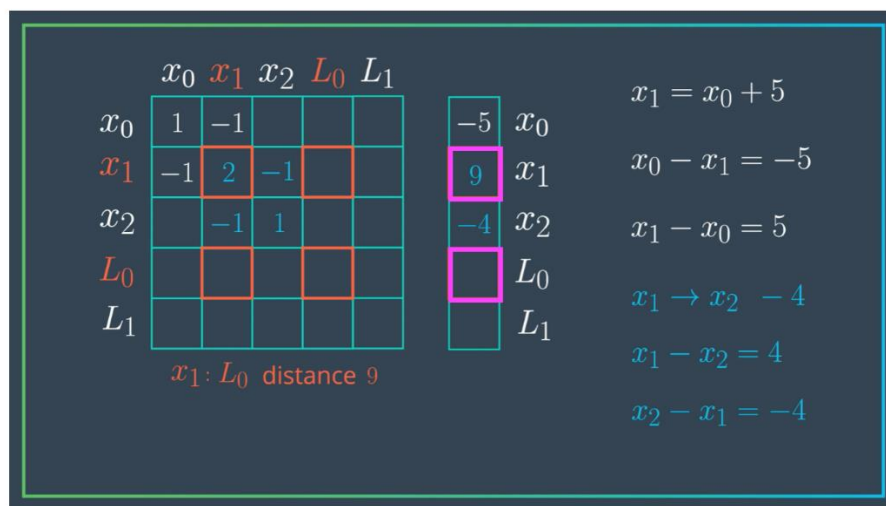
-5	$x_0$	$x_1 = x_0 + 5$
9	$x_1$	$x_0 - x_1 = -5$
-4	$x_2$	$x_1 - x_0 = 5$
	$L_0$	$x_1 \rightarrow x_2 - 4$
	$L_1$	$x_1 - x_2 = 4$ $x_2 - x_1 = -4$

Let me another quiz:

Supposing:  $x_1$  we see landmark  $L_0$  at the distance 9 this gives me a **relative constraint between**  $x_1$  over here and landmark zero, which is over here.

So just like before these link two things together relatively the  $x_1$  and the  $L_0$

Now this doesn't look like a sub matrix, but it's spread a little bit apart But I want you to modify those four values in the matrix and those two values in the vector to accommodate that we believe the location of  $L_0$  is 9 greater than the robot position  $x_1$



**For the two highlighted values in the column vector above, what will there new value be after this landmark relationship is added to these constraint matrices? Right now the values are 9, 0.**

☐

9, 0

☒

18, 9

☐

-5, 9

☐

0, 9



-5, 4

The answer:

$x_1 - L_0 = -9$  because  $L_0 - x_1 = 9$  is a measurement of 9

So let's add this in,

3 -1

-1 1

	$x_0$	$x_1$	$x_2$	$L_0$	$L_1$			
$x_0$	1	-1				-5	$x_0$	$x_1 = x_0 + 5$
$x_1$	-1	3	-1	-1		0	$x_1$	$x_0 - x_1 = -5$
$x_2$		-1	1			-4	$x_2$	$x_1 - x_0 = 5$
$L_0$		-1		1		9	$L_0$	$x_1 \rightarrow x_2 - 4$
$L_1$							$L_1$	$x_1 - x_2 = 4$

$x_1 : L_0$  distance 9  
 $x_1 - L_0 = -9$   
 $L_0 - x_1 = 9$

$x_2 - x_1 = -4$

Matrix modification

Initial robot location, If we define  $x_0$  to be zero, which is the origin of the map then what this mean is we add one over here and zero over here and the reason why is this constraint is that  $x_0$  zero.

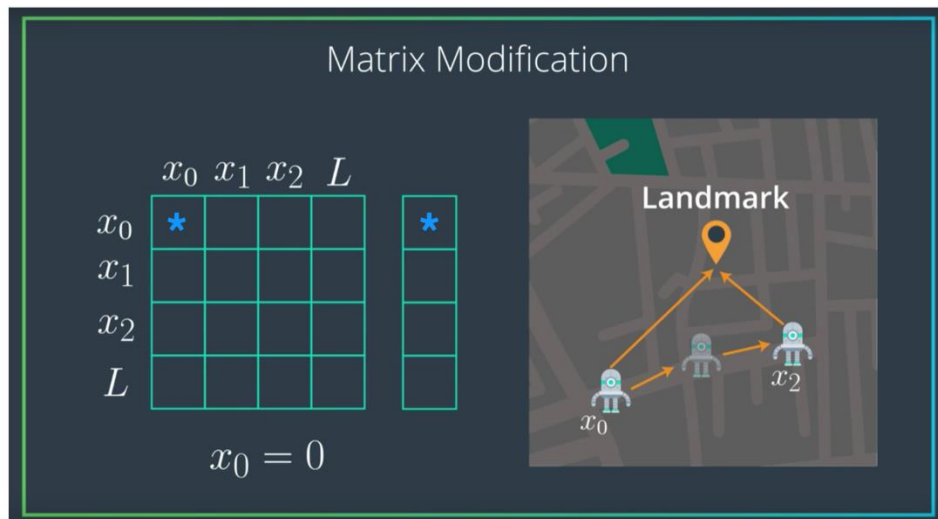
	$x_0$				
$x_0$	1				0

$x_0 = 0$

So let's take a robot moving around and let's see it sees a landmark, from the first pose  $x_0$  and from the third pose  $x_2$  but not from the second pose.

I want you in this matrix over here, checkmark all the fields that are being modified by graham's law, to binary check and the same for the vector without putting actual number in.

So go to the matrix and ask yourself, which field will be zero the ones that are untouched and which ones will be not zero the ones you will modified.



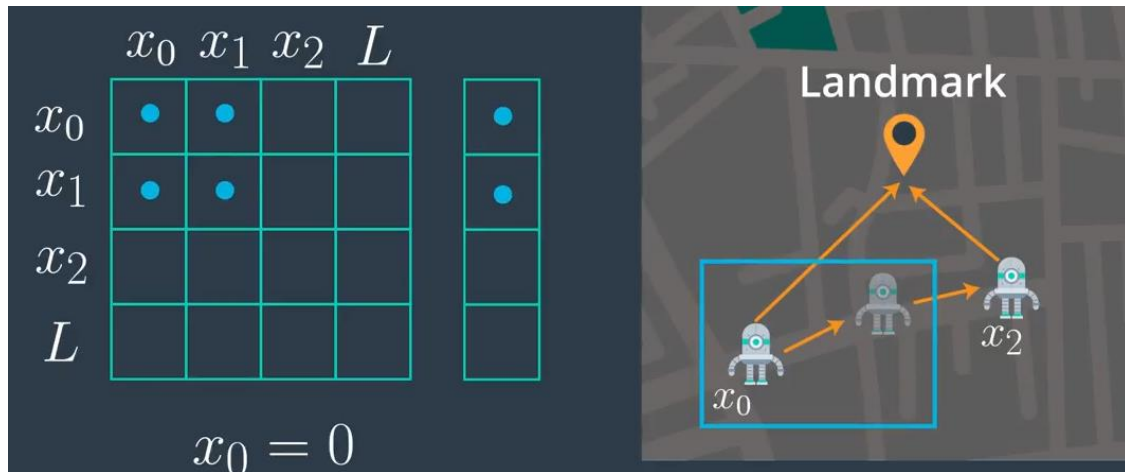
## Mark the Relationships

For this quiz, take out a piece of paper and draw these 4x4 and 4x1 grids. Draw a dot (or other mark) wherever there is a relationship between poses and/or landmarks. If nothing will change about a cell, it will remain 0 and you can leave it blank.

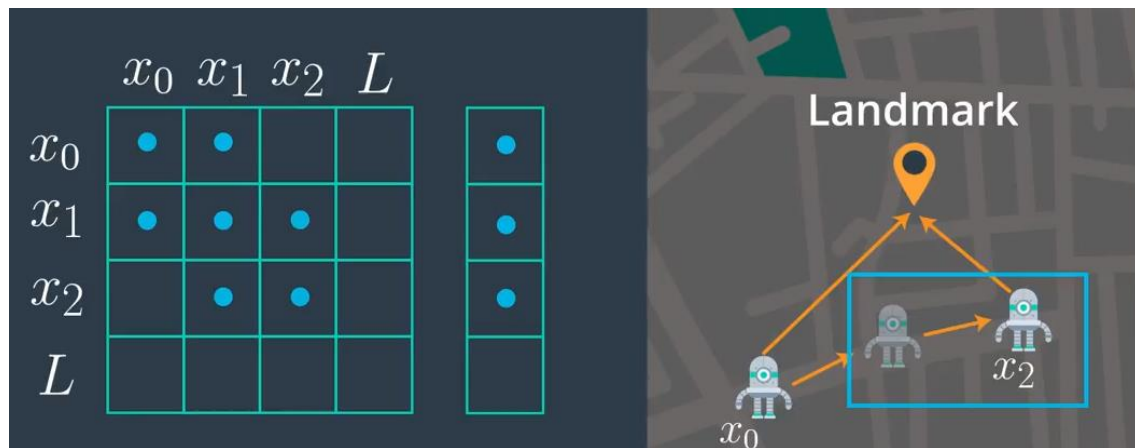
An example has been started in the image below, note that the initial constraint has been marked for you!

- 1- The initial is  $x_0$
- 2- The Van to the second motion touches these things over here.

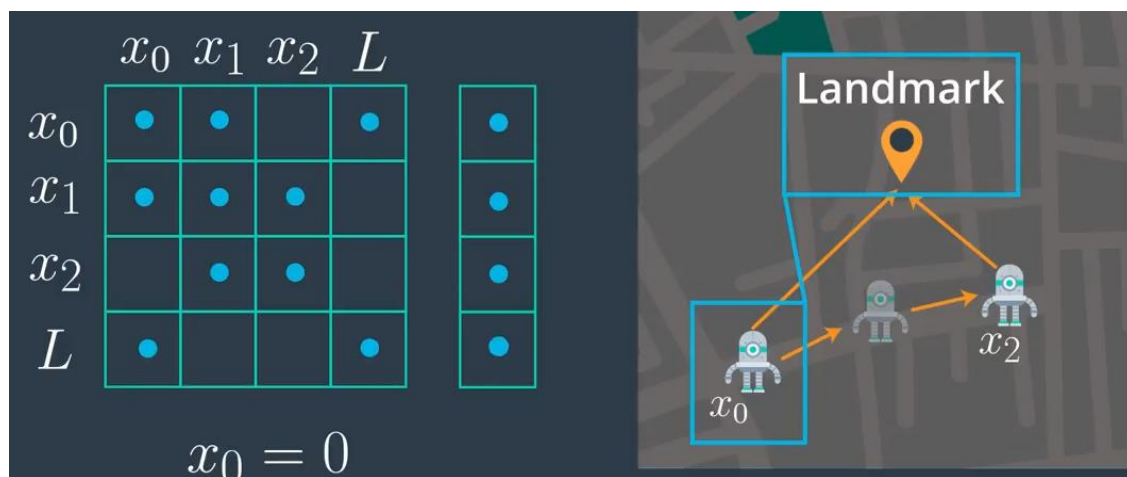




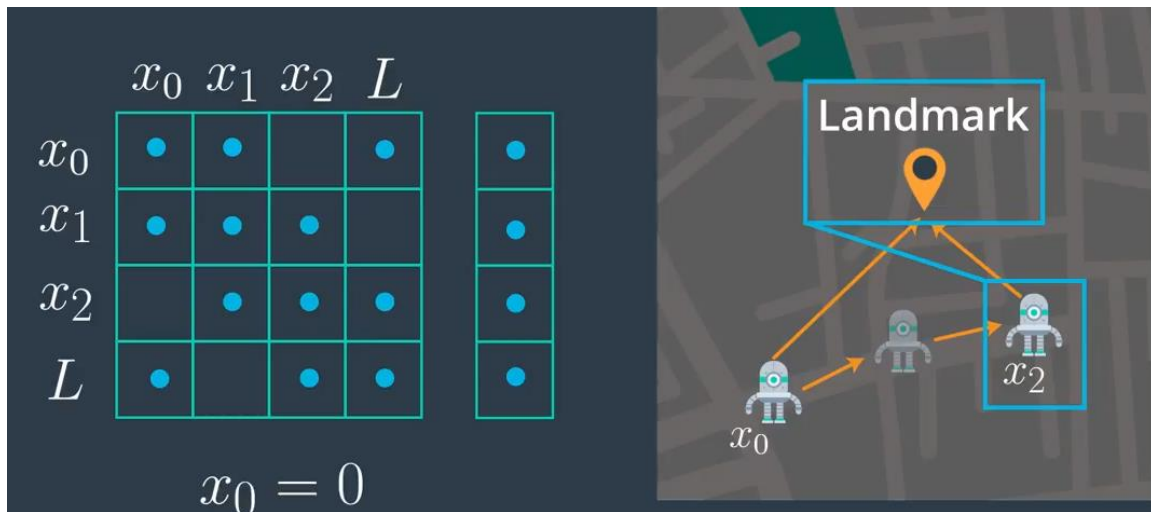
3- The second to third, these guys and then landmark observation over here.



4- Put something between  $x_0$  and the landmark,



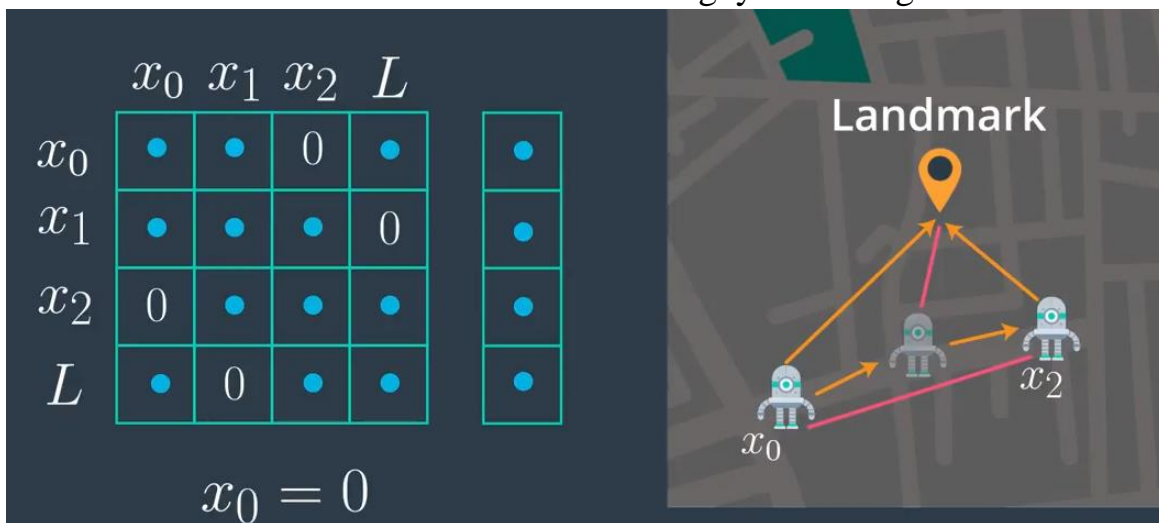
5- The observation over here put something between  $x_2$  in the landmark.



6- That means we have only the following places that are still zero.

These means there is no direct constraint between  $x_2$  and  $x_0$ .

There is this no direction motion information between these guys and there's no direct constraint  $x_1$  and  $L$  which is this guy is missing over here.



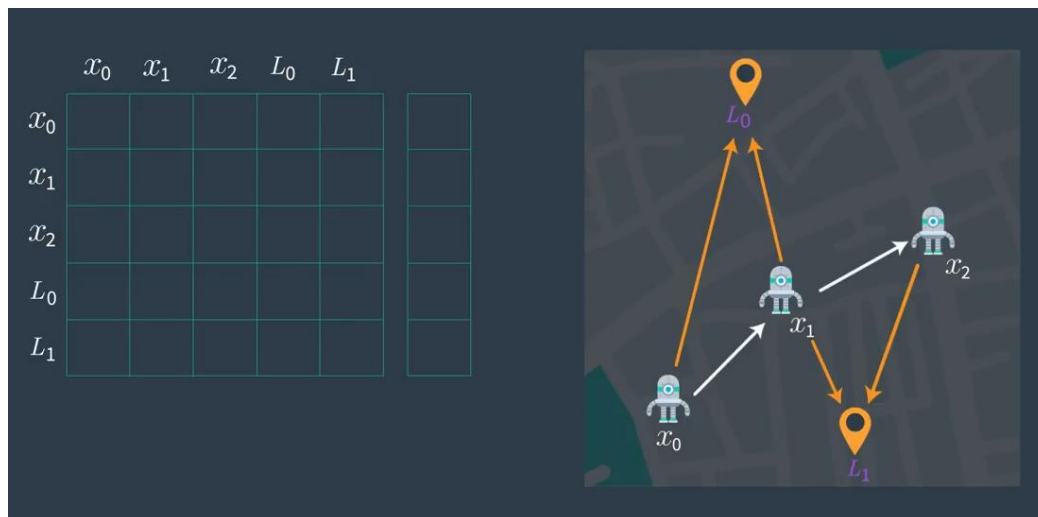
## SIMULTANEOUS LOCALIZATION AND MAPPING

Quiz:

Now we have two landmarks and the picture i'm giving you is robot with three total positions.

There's a landmark here, and a landmark here, and say it's being in these two positions in these landmark two poses, but landmark  $L_1$  is not seen in  $x_2$  and landmark  $L_2$  is not seen in  $x_0$  and of the 30 fields over here

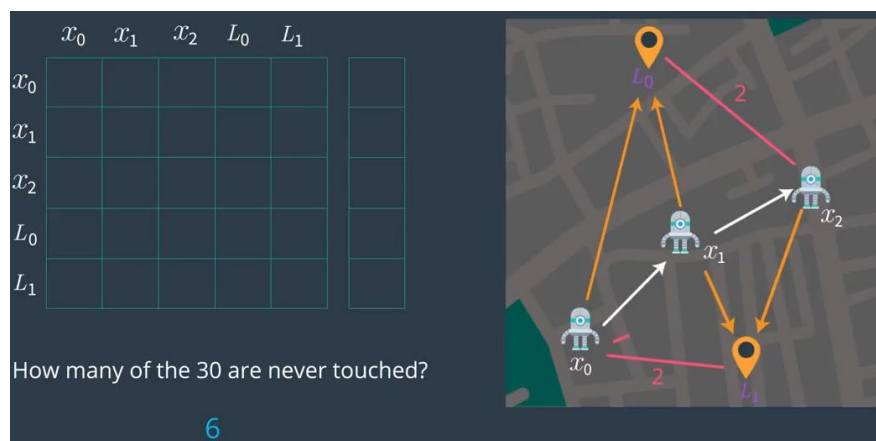
How many of the 30 fields are never touched ?



### QUESTION:

Given the above constraints, after taking all of these measurements into account, how many of the 30 grid cells in the constraint matrices are never changed (and remain 0)? The answer should be a single integer. = 6 this missing value

- 1- This is guy here is given me two values in the matrix , this guy here another two and the link here is also missing

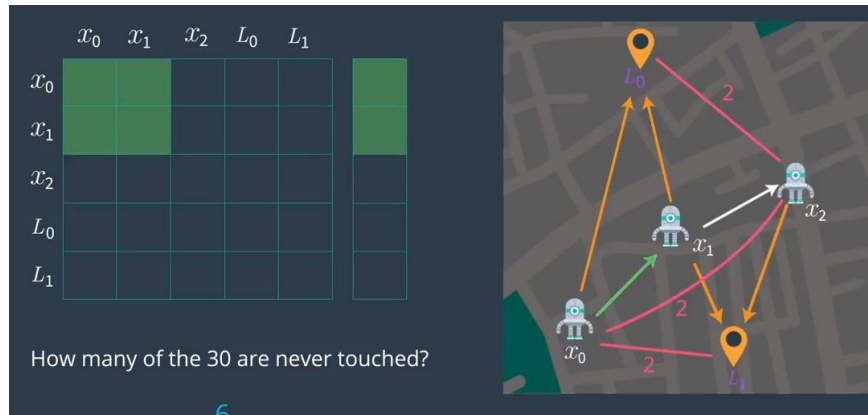


2- This guy here another two and this link here is also missing.

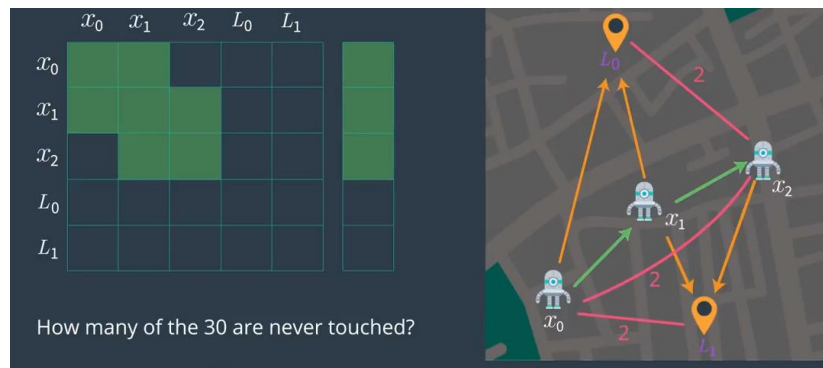
So it's six values missing.

Let's prove it ourselves.

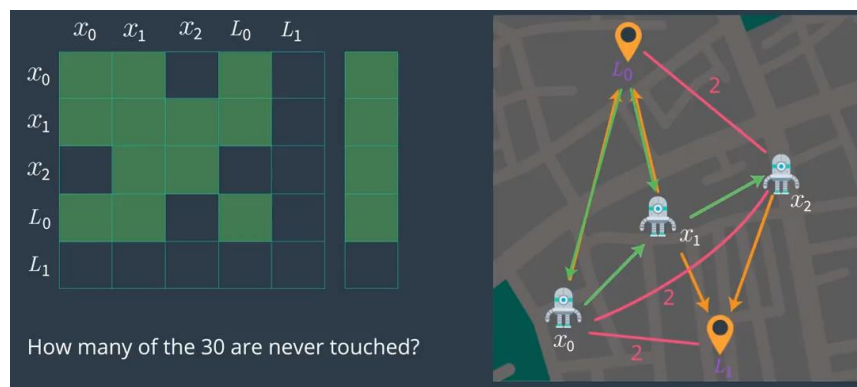
$x_0$  to  $x_1$  filter this area



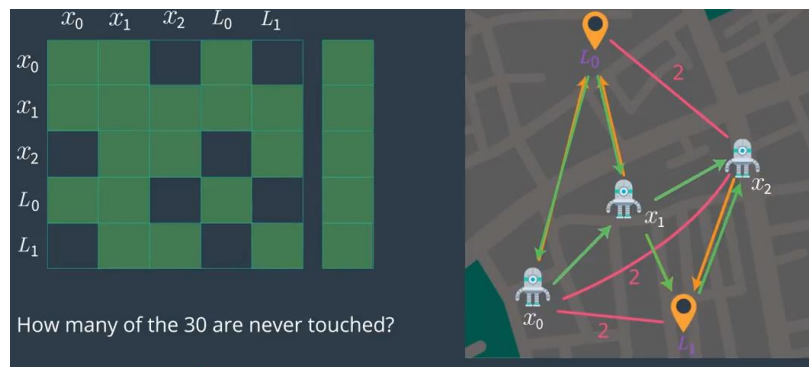
$x_1$  to  $x_2$  filter this area



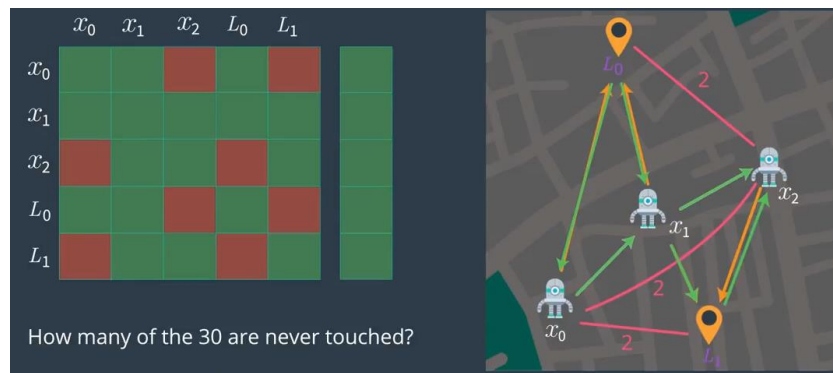
So seeing end mark  $L_0$  from  $x_0$  and  $x_1$  means you fill these guys over here. And the main the diagonal



There is the other mark from  $x_1$  to  $x_2$  means fill these guys over here

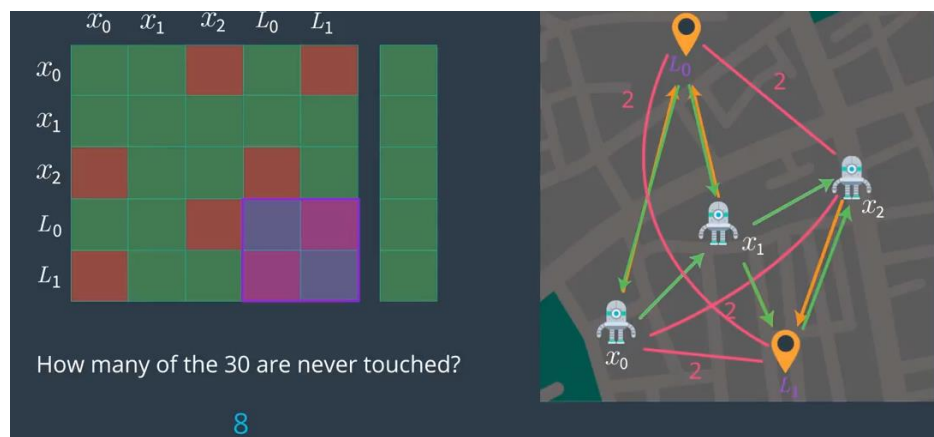


So let's count the ones that are still open and here're the other ones. And my answer was actually wrong it's eight I overlooked



there is no direct link from  $L_1$  to  $L_2$  either. So my apologies that I gave you the wrong number but it proves the utilizations not an easy question it's harder that I thought and the reason is there's also no direct link that constraints  $L_1$  and  $L_2$  directly.

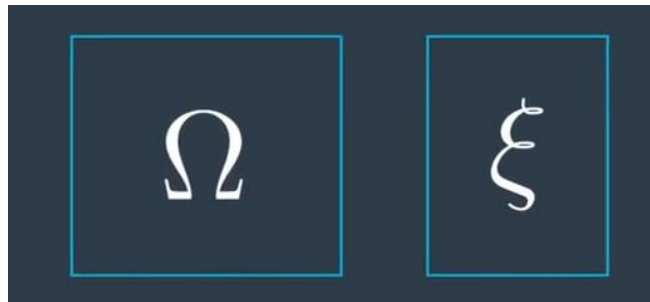
Landmark can't see So they can't put a direct link between any two landmarks or put differently in this part over here our matrix will always be a diagonal matrix



Last thing after programming why this make any sense.

Suppose you fill the matrix, which I call omega, and the vector which will give the Greek of Xi

Then I can find the best solution for all the landmark position or the work position, by very simple mathematical trick and that is completely counter intuitive.



I invert the omega, I right multiply with Xi, and out comes a vector of mu

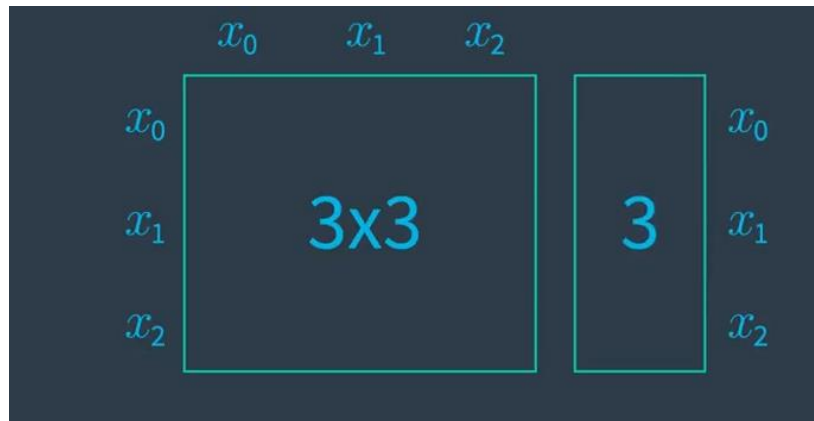
$$\mu = \Omega^{-1} \xi$$

Which gives me the best estimate for all the location and the landmark locations,

Now, that is quite amazing, because all it means in graph lemmas, that we keep adding numbers to these matrices every time sequence straight and when you are done with it, you run a very simple procedure, and out comes the best places for your robot.

Let`s program it

I`m giving you my matrix class, so I can ask to do what i`m asking you do is to build a 3 x 3 matrix, and of course a 3 x 1



By which you shall state that our initial location is  $-3$

$X_1$  exploitation is obtained by adding  $x_0 + 5$

And  $X_2 = 3 + x_1$

So you expectation what we should get out, we run  $\mu$  equals  $\omega - 1$  times  $\xi$  trick

$X_0 = -3$  ,  $X_1 = 2$  ,  $X_2 = 5$

I'm giving you a matrix class,

What I want you to write code that incrementally step-by-step constructs the  $\omega$  vector and the  $\xi$  function, and then returns to me those result over here these empty function in the code that accepts as parameter the initial

```
: import numpy as np

def mu_from_positions(initial_pos, move1, move2):

    ## TODO: construct constraint matrices
    ## and add each position/motion constraint to them

    # Your code here
    omega = np.array([[1,0,0],
                      [-1,1,0],
                      [0,-1,1]])
    xi = np.array([[initial_pos],[move1],[move2]])
    # display final omega and xi
    print('Omega: \n', omega)
    print('\n')
    print('xi: \n', xi)
    print('\n')
    mu = np.dot(np.linalg.inv(np.matrix(omega)),xi)
    ## TODO: calculate mu as the inverse of omega * xi
    ## recommended that you use: np.linalg.inv(np.matrix(omega)) to calculate the inverse

    return mu
```



```
: # call function and print out `mu`
mu = mu_from_positions(-3, 5, 3)
print(mu.shape)
print('Mu: \n', mu)
```

```
Omega:
[[ 1  0  0]
 [-1  1  0]
 [ 0 -1  1]]
```

```
xi:
[[-3]
 [ 5]
 [ 3]]
```

```
(3, 1)
Mu:
[[-3.]
 [ 2.]
 [ 5.]]
```

## Omega and Xi

To implement Graph SLAM, a matrix and a vector (omega and xi, respectively) are introduced. The matrix is square and labelled with all the robot poses (xi) and all the landmarks (Li). Every time you make an observation, for example, as you move between two poses by some distance  $\Delta x$  and can relate those two positions, you can represent this as a numerical relationship in these matrices.

Below you can see a matrix representation of omega and a vector representation of xi.



### Solving for x, L

To "solve" for all these poses and landmark positions, we can use linear algebra; all the positional values are in the vector  $\mu$  which can be calculated as a product of the inverse of omega times xi.

Quiz: Construct constraints for 3 motions and return  $\mu$

In the following example, you will complete the function call `mu_from_positions(-3, 5, 3)`, which takes in 3 robot poses/moves:

- initial pose: -3
- moves by 5



- moves by 3

In this function, you should construct the constraint matrices  $\omega$  and  $\mathbf{x}_i$  and calculate  $\mu$ . The final call should result in a  $\mu$  of:

```
[[ -3.0],
 [ 2.0],
 [ 5.0]]
```

## Constraint Updates

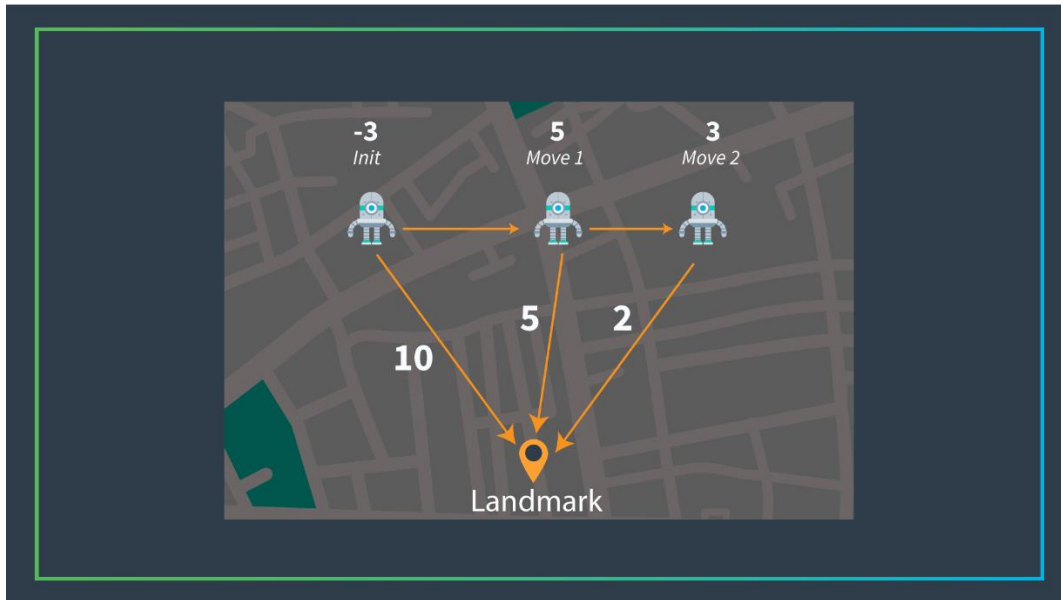
We will not consider landmark sensor measurements in this example, only robot poses.

### *Motion*

When your robot moves by some amount  $dx$  update the constraint matrices as follows:

- Add  $\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$  to  $\omega$  at the indices for the intersection of  $x_t$  and  $x_{t+1}$
- Add  $-dx$  and  $dx$  to  $\mathbf{x}_i$  at the rows for  $x_t$  and  $x_{t+1}$

## Quiz: Landmark position

**QUESTION:**

Given only the above constraints between poses ( $x_0$ ,  $x_1$ ,  $x_2$ ) and the landmark position. Where do you think the landmark is? (The answer should be a single integer value.)

## Landmark Position

The landmark,  $L$ , is at the value  $7$ .

Think about the relationship between  $x_0$  and  $L$ .  $x_0$  is at  $-3$  and sees  $L$  a distance of  $10$  away:  $L = -3 + 10$ .

And if we keep adding poses and sensor measurements this remains consistent. As  $x_0$  moves to  $x_1$ , we get  $L = -3 + 5 + 5$ .

And the final loop all the way to  $x_2$ :  $L = -3 + 5 + 3 + 2$ .

## 2D case

In these examples, we've been showing you change in only one dimension, the  $x$ -dimension. In the project, it will be up to you to represent  $x$  and  $y$  positional values in  $\omega$  and  $\xi$ . One solution could be to create an  $\omega$  and  $\xi$  that are  $2x$  larger than the number of robot poses (that will be generated over a series of time steps) and the number of landmarks, so that they can hold both  $x$  and  $y$  values for poses and landmark locations. I might suggest drawing out a rough solution to graph slam as you read the instructions in the next notebook; that always helps me organize my thoughts. Good luck!

## Landmark position

Let's add the landmark.

Let's say the landmark is being seen at all time steps

Let's say in the very first time the difference between position and landmark is 10 obviously this is a 1-dimensional example and not 2-dimensional as the picture suggests over here.

Then it's 5 and then 2

Now I want to extend your routine to accommodate the landmark. Specifically I want you to use function that I coded for you first that is very useful that is called `expand`.

You can run `omega.expand`

`xi.expand` to take of 3x3 matrix or vector and move it to a 4x4 vector that you actually need when you have to include landmark itself to now you have additional parameter input parameters of measurement 0,1 and 2

the function parameter :

1- Init

It is the motion commands

2- Move1

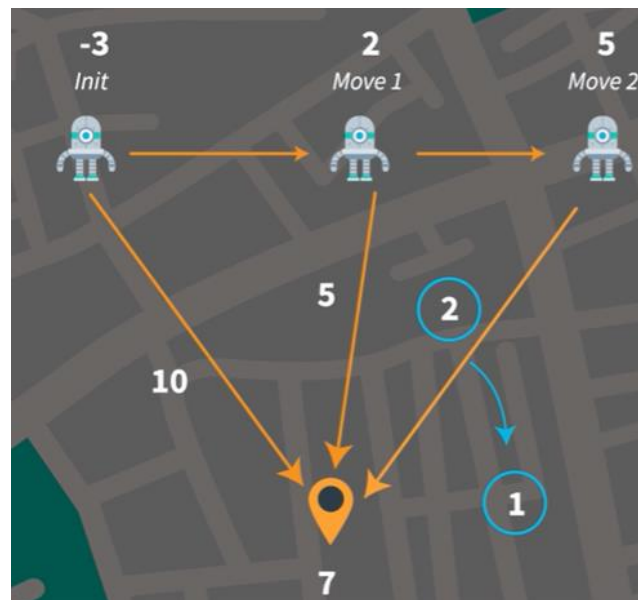
3- Move2

And the 3 measurement commands for the 3 different poses then using expansion command

## Introducing noise

Let's look at robot forward robot motion and say I changed the last measurement from two down to one.

You might remember the robot poses, -3, 2 and 5 before their modification, then my position was seven but if one doesn't really add up, a one suggest I might be at a different distance to the seven than over here that come from this side



Quiz:

First, I want you to know if I make this modification, What is the effect on  $x_2$  ?

Will estimate be smaller than five as we shrink the robot path a little bit ?

Will be exactly equals 5 Like before or will be larger than 5 ?

Or will be larger then 5

Also we know what's the effect on  $x_0$  will it be smaller than -3

Equals - 3

Solution:

When implement in code

- 1- The first position is completely unchanged, [-3.000]
- 2- The third one , went from five to 5.5
- 3- And the landmark went to down from 7 to 6.8

So graphically this guy becomes larger than 5 and this landmark even shift a liitle bit to the left mark these two things and make to closer together that they were before when they had a separation distance of two.

Now this picture doesn't really explain it well because it's a 2D but in 1D that's exactly what's happening

- 1- The initial position is unchanged.

So, these are the correct answer

Now I will be blown away if you guess them correctly.

The reason why the original position doesn't change is the only information we have about the absolute coordinate location is the very first initial position anchor that we said it has to be -3

None of the relative rubber bands change the fact that we need this guy to be -3

A relative strength changed in these two things over here means the rubber band is different , but it's relative thing

- This is the only absolute constraint we put in
- -3
- Clearly, the absolute location of the first position doesn't change

The reason why it becomes larger than 5 iis well think about rubber bands

Our landmark is at around 7 , we belief to be a position 5 in the noise free case, we just a tight rubber band between them

It's not two anymore m it's now one

**That mean we are inclined to move the landmark and this position closer function. And that's exactly what happens.**

If you go find position become 5.5 and the landmark become 6.872

Now, this is case where the rubber bands don't add up

This is one of places where graphs limit is just magical

Before everything add up in this structures and these cycles might not add up because we have noise and motionless measurements .

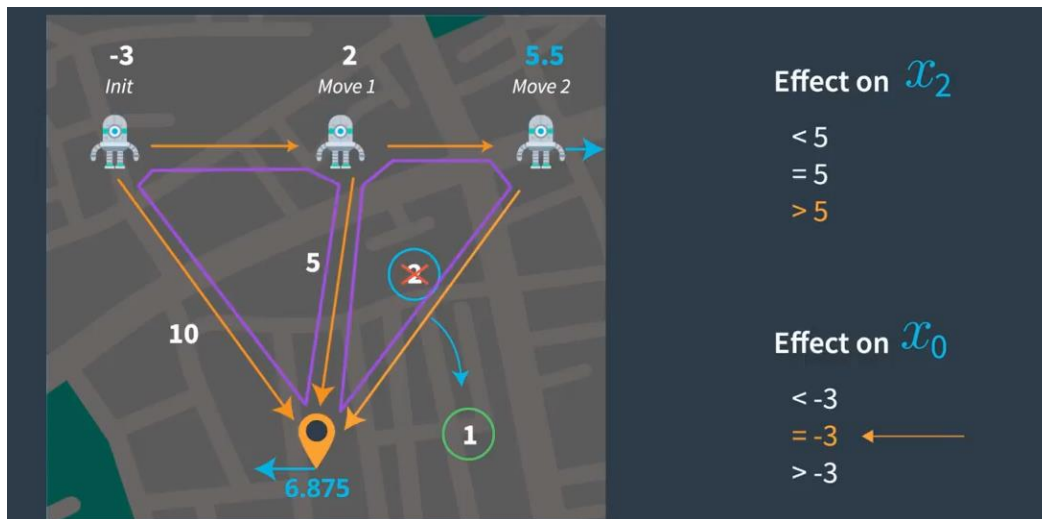
الخلاصة ان عند اخر motion

بتحاول مش يكون فى اى نويز بيبقى noise free

بتعمل اية بتخلى ال idstance 1 not 2

علشان تبقى قريبة من landmark

What our method does by computing we find the best solution to the relaxation of those rubber bands

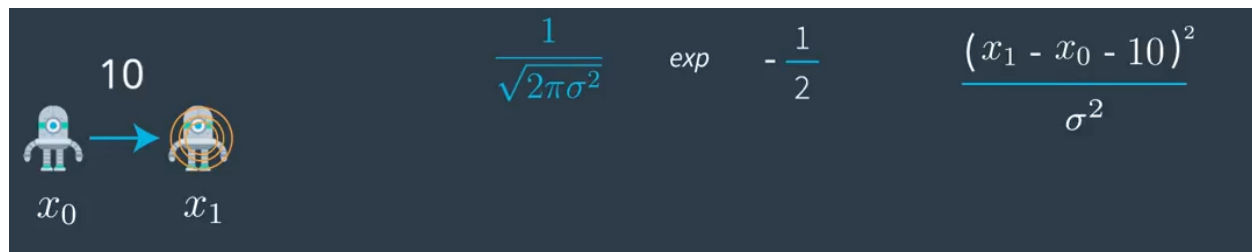


## Confident Measurement

Suppose we have two roller position  $X_0$  and  $x_1$  and we know that 10-apart with some Gaussian noise, we know that the Gaussian noise and exploitation moves the rightward position 10-off the leftward position, but there is some uncertainty.

When about talked about common filters, we talked about Gaussians and this uncertainty might look as follows.

There is a constant , exponential in the expression that  $x_1 - x_0$  should relax to 10, but deviate from it.



This Gaussian constraint over here characterized a constraint  $X_1, x_0$  and wishes them to be exactly 10-apart.

The Gaussian is maximum with this equation is fulfilled but if the residual is not equals zero

مش بیسای صفر Reminder طول ما

يبقى كدة انت موصلت للماكسيمام لجاوسين

There `s still a probability associated with it.

Let's now model a second motion. Say  $x_2$  is 5-apart.

We know get an even bigger Gaussian relative to the very first one, but the local constraint over here reads just like the constraint over here.




$$\frac{1}{\sqrt{2\pi\sigma^2}} \exp -\frac{1}{2} \frac{(x_1 - x_0 - 10)^2}{\sigma^2}$$

$$\frac{1}{\sqrt{2\pi\sigma^2}} \exp -\frac{1}{2} \frac{(x_2 - x_1 - 5)^2}{\sigma^2}$$

Now the total probability of this entire over here is the product of these two things.

If we want to maximize the product, we can play a number of interesting tricks

- 1- First, the constant has no bearing on the maximum just on the absolute value So, if we want to find the best values for  $x_0$  and  $x_1$  and so on, we can drop the constant.



$$\frac{1}{\sqrt{2\pi\sigma^2}} \exp -\frac{1}{2} \frac{(x_1 - x_0 - 10)^2}{\sigma^2} \cdot \frac{1}{\sqrt{2\pi\sigma^2}} \exp -\frac{1}{2} \frac{(x_2 - x_1 - 5)^2}{\sigma^2}$$

- 2- Secondly, we can drop the exponential if we are willing to turn the product into an addition, and remember, we edit things in Omega and in Sigma, that's Why.



$$\frac{1}{\sqrt{2\pi\sigma^2}} \exp -\frac{1}{2} \frac{(x_1 - x_0 - 10)^2}{\sigma^2} + \frac{1}{\sqrt{2\pi\sigma^2}} \exp -\frac{1}{2} \frac{(x_2 - x_1 - 5)^2}{\sigma^2}$$

## SIMULTANEOUS LOCALIZATION AND MAPPING

- 3- And Finally, we can actually drop the minus a half Turn out it also plays no role in the maximization of this expression.

$$\begin{aligned}
 & \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\frac{(x_1 - x_0 - 10)^2}{\sigma^2}\right) \\
 + & \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\frac{(x_2 - x_1 - 5)^2}{\sigma^2}\right)
 \end{aligned}$$

So it turn out what you added, where constraints just like these and you even added them as a certain strength of one over sigma square.

In particular if you really believe that a constraints is true, you should add a larger value in this matrix over here and on the right side you should multiply the right constraint with an even longer value.

For definitely, talking the expression like this and multiplying the sigma square, you get something of this nature over here where one over sigma regulates how confident you are for a small sigma, so five is much larger than one

That means you have much more confidence.

$$\begin{pmatrix} \vdots & \vdots \end{pmatrix} \begin{pmatrix} \vdots \end{pmatrix} \begin{pmatrix} 5 & -5 \\ -5 & 5 \end{pmatrix} \begin{pmatrix} -\Delta & 5 \\ \Delta & 5 \end{pmatrix} \quad \frac{1}{\sigma} x_1 - \frac{1}{\sigma} x_0 = \frac{10}{\sigma}$$

So, the last measurement has a really high confidence So I wanted to multiply the last measurement between X2 and our landmark with the factor of 5

$$x_2 * L \Rightarrow 5$$



## Code

```
import numpy as np

def mu_from_positions(initial_pos, move1, move2, Z0, Z1, Z2):

    ## TODO: construct constraint matrices
    ## and add each position/motion constraint to them

    # initialize constraint matrices with 0's
    # Now these are 4x4 because of 3 poses and a landmark
    omega = np.zeros((4,4))
    xi = np.zeros((4,1))

    # add initial pose constraint
    omega[0][0] = 1
    xi[0] = initial_pos

    # account for the first motion, dx = move1
    omega += [[1., -1., 0., 0.],
             [-1., 1., 0., 0.],
             [0., 0., 0., 0.],
             [0., 0., 0., 0.]]
    xi += [[-move1],
           [move1],
           [0.],
           [0.]]
```

---

```
    # account for the second motion
    omega += [[0., 0., 0., 0.],
             [0., 1., -1., 0.],
             [0., -1., 1., 0.],
             [0., 0., 0., 0.]]
    xi += [[0.],
           [-move2],
           [move2],
           [0.]]

    ## TODO: Include three new sensor measurements for the landmark, L
    # incorporate first sense
    omega += [[1., 0., 0., -1.],
             [0., 0., 0., 0.],
             [0., 0., 0., 0.],
             [-1., 0., 0., 1.]]
    xi += [[-Z0],
           [0.0],
           [0.0],
           [Z0]]

    # incorporate second sense
    omega += [[0., 0., 0., 0.],
             [0., 1., 0., -1.],
             [0., 0., 0., 0.],
             [0., -1., 0., 1.]]
    xi += [[0.],
           [-Z1],
           [0.],
           [Z1]]
```

---

```

## This third sense is now *very confident* and
## we multiply everything by a strength factor of 5 instead of 1
# incorporate third sense
omega += [[0., 0., 0., 0.],
          [0., 0., 0., 0.],
          [0., 0., 5., -5.],
          [0., 0., -5., 5.]]

xi += [[0.],
       [0.],
       [-Z2],
       [Z2]]

# display final omega and xi
print('Omega: \n', omega)
print('\n')
print('xi: \n', xi)
print('\n')

## TODO: calculate mu as the inverse of omega * xi
## recommended that you use: np.linalg.inv(np.matrix(omega)) to calculate the inverse
omega_inv = np.linalg.inv(np.matrix(omega))
mu = omega_inv*xi
return mu

```

```

: # call function and print out `mu`
mu = mu_from_positions(-3, 5, 3, 10, 5, 1)
print('Mu: \n', mu)

```

### Solution result

```

Omega:
[[ 3. -1.  0. -1.]
 [-1.  3. -1. -1.]
 [ 0. -1.  6. -5.]
 [-1. -1. -5.  7.]]

xi:
[[-18.]
 [-3.]
 [ 2.]
 [ 16.]]

Mu:
[[-3.          ]
 [ 2.32142857]
 [ 6.28571429]
 [ 6.67857143]]

```

## Implementing SLAM

Now you have all the information you need to complete a 2D version of SLAM. You know how to update constraint matrices and incorporate strength and noise into a model of motion and sensor measurements.

Next, you'll be given the following variable and asked to complete an implementation of SLAM for your final project, Congratulations on getting this far!