A new way of thinking about algorithms

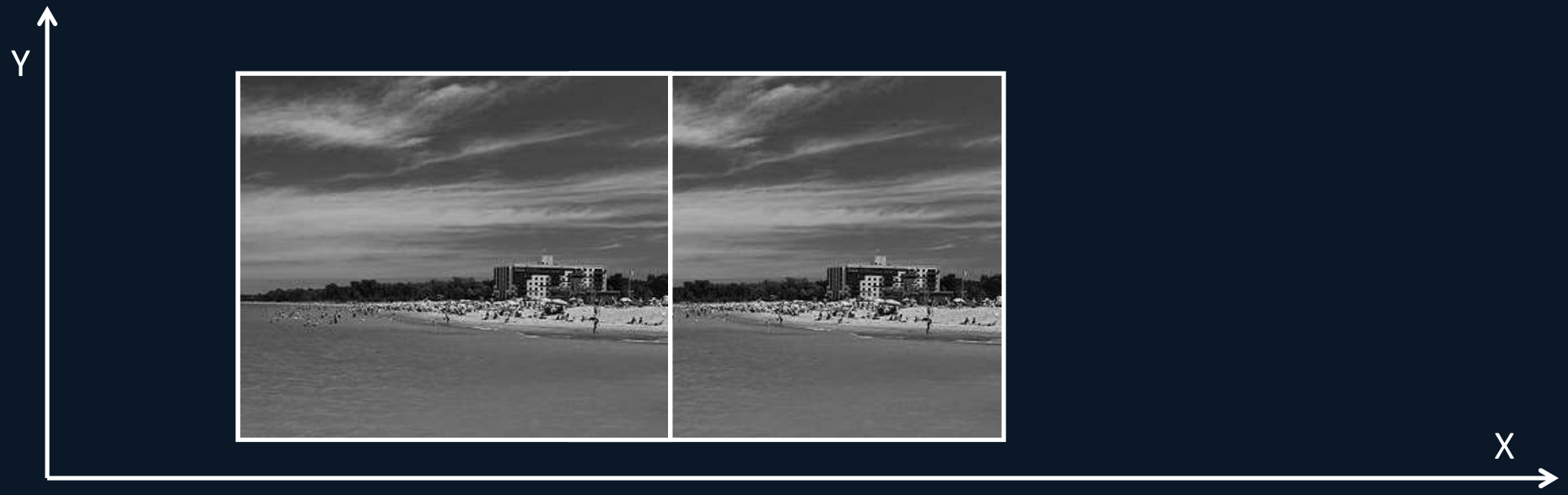
To study segmentation and registration algorithms, we need to first understand the typical conceptual viewpoint from which such algorithms are described and understood in this field.

Image registration is a good example to illustrate this conceptual viewpoint, so we will start with a registration example.

In registration, we have a moving image that we want to align with a fixed image.

In this example, the only required transformation to align the moving image with the fixed image is a translation along the X axis.

The images are already perfectly aligned along the Y axis.

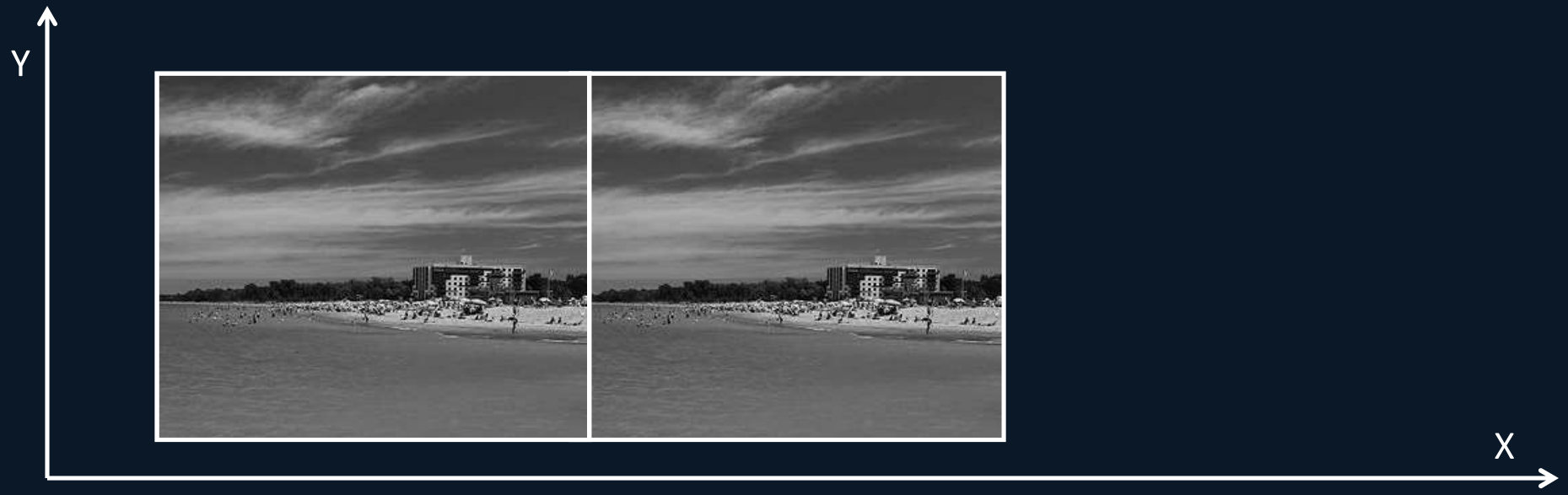


Imagine that before beginning, our two images already had a bit of overlap, like this.

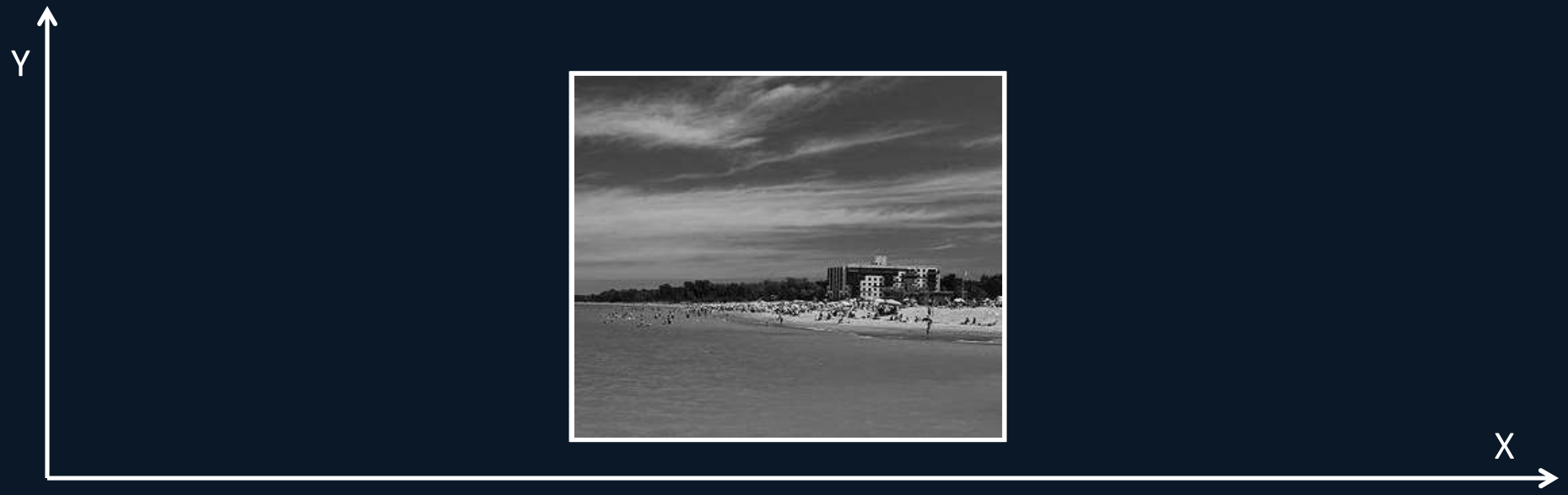
We could give this scenario some kind of “goodness” score measuring how good the alignment is.

Let’s say that for the situation above, this score is S.

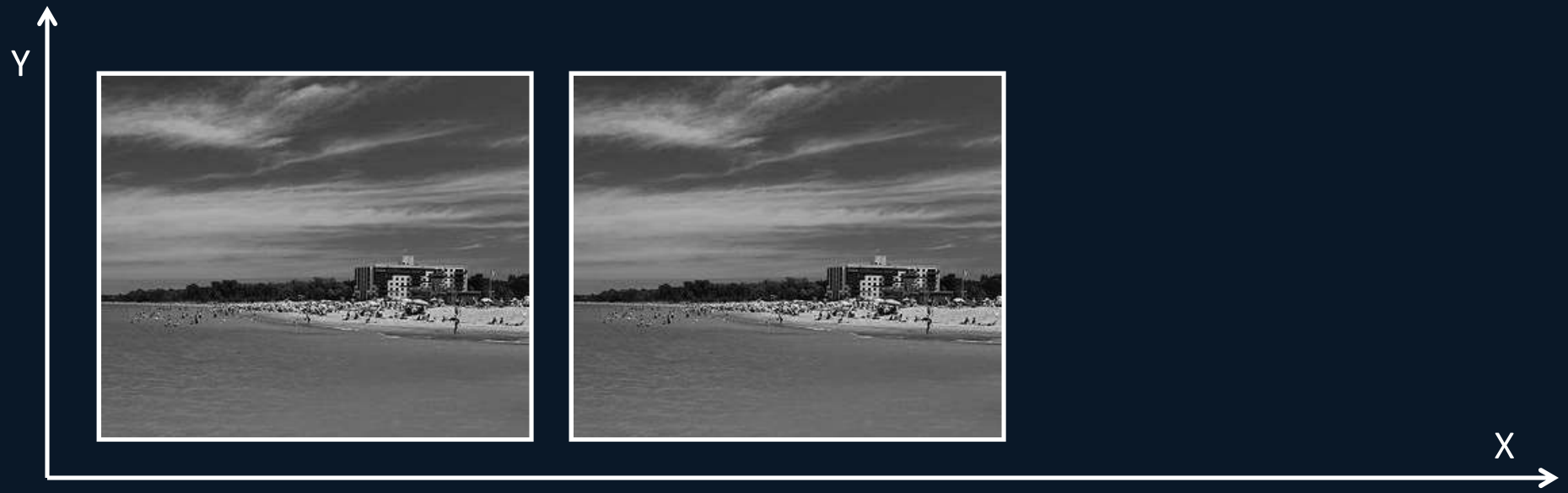
This situation would give a better score than S.



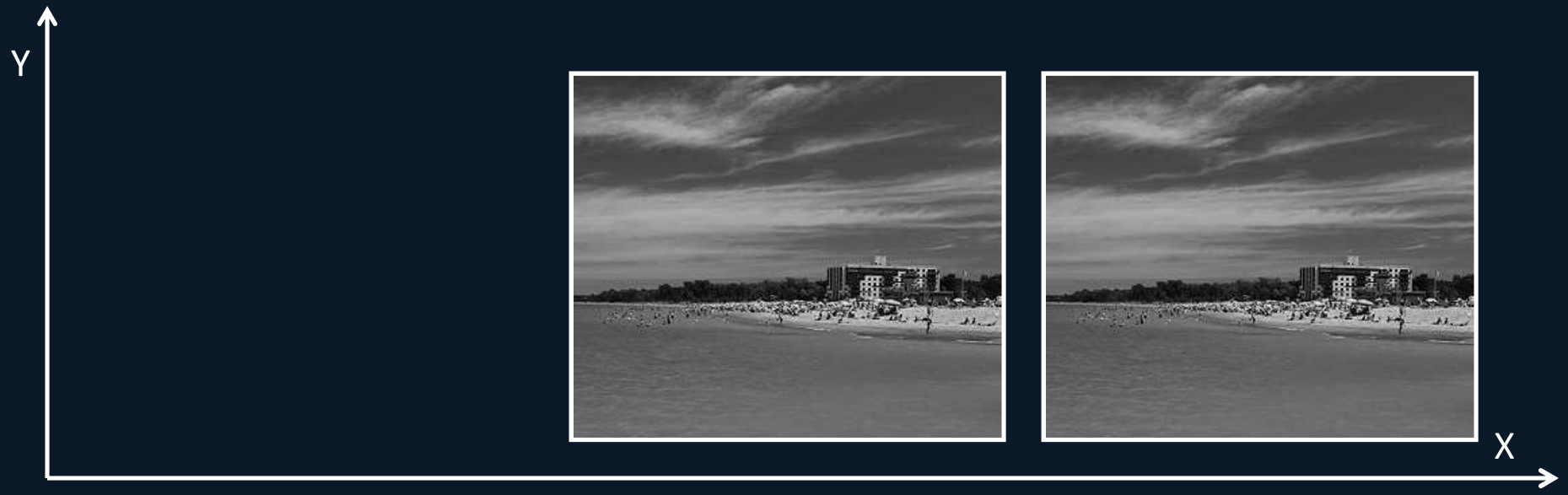
This situation would give a worse score than S.



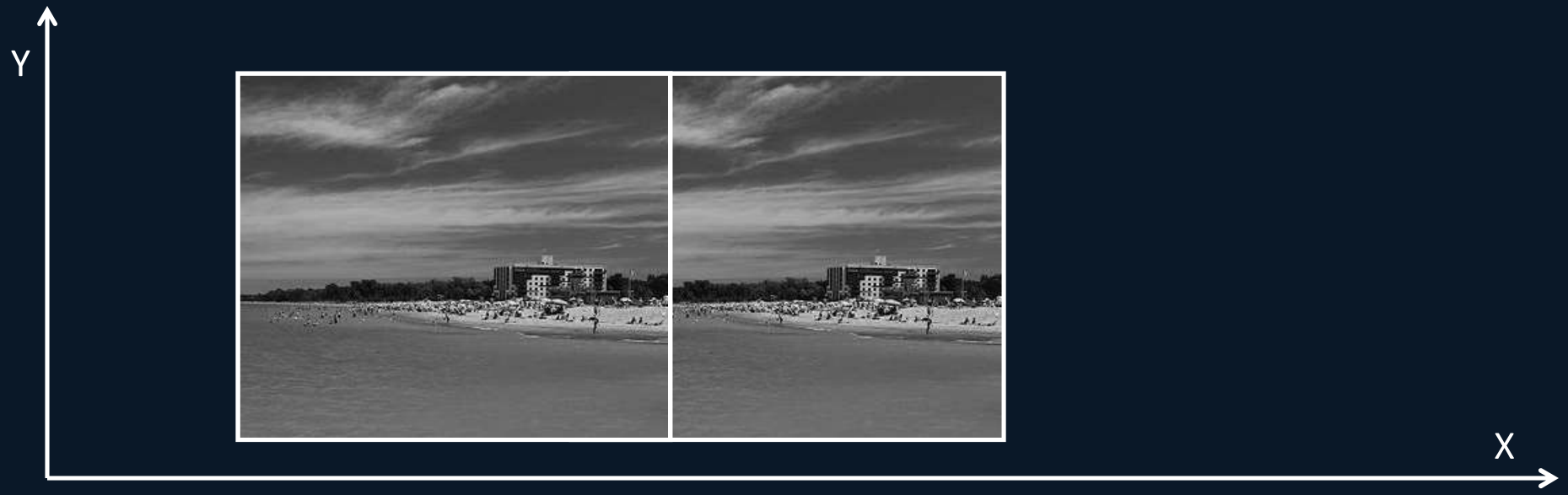
This situation (where the moving image is exactly aligned on top of the fixed image) would give the best possible score.



This situation would give the worst possible score.

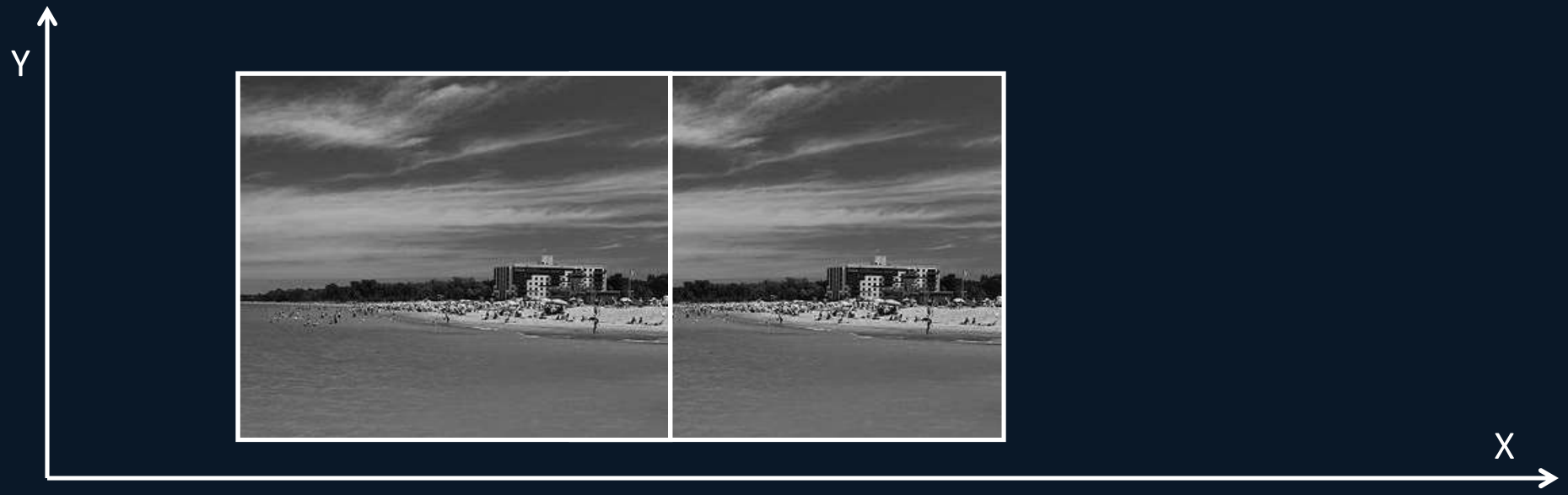


This situation would also give the same, worst possible score.

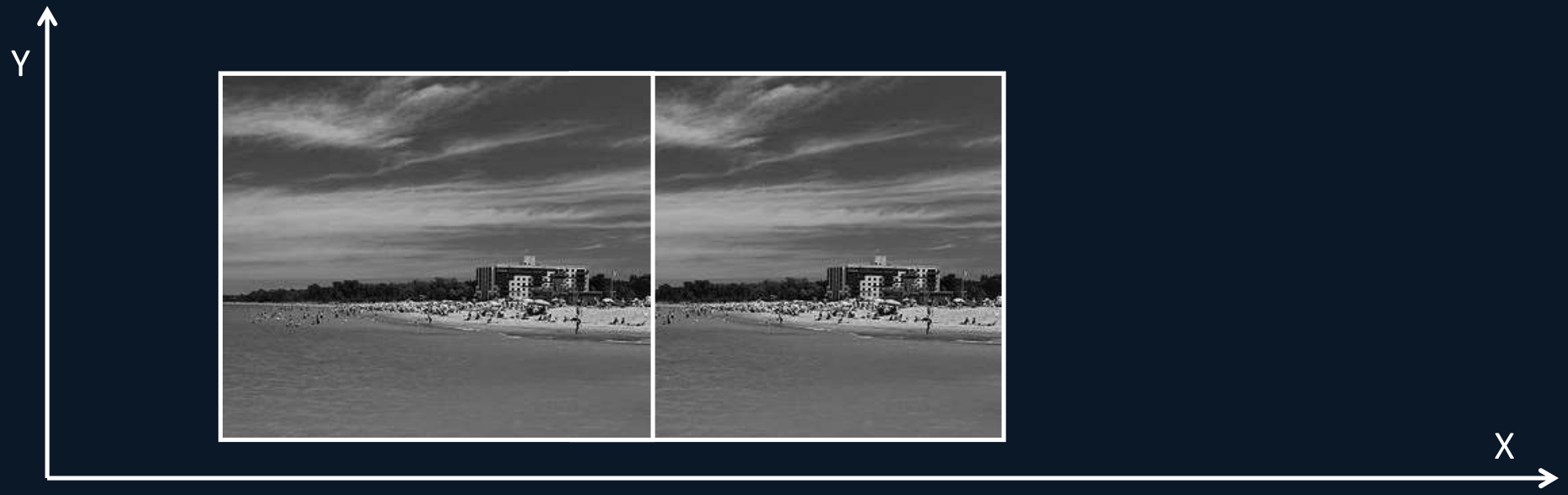


For the moment, we don't yet know how to compute such a "goodness" score. We will learn how to do that later. For now, imagine that we can compute such a score, and that large scores are good scores.

We could devise an algorithm that aligns these two images as follows.

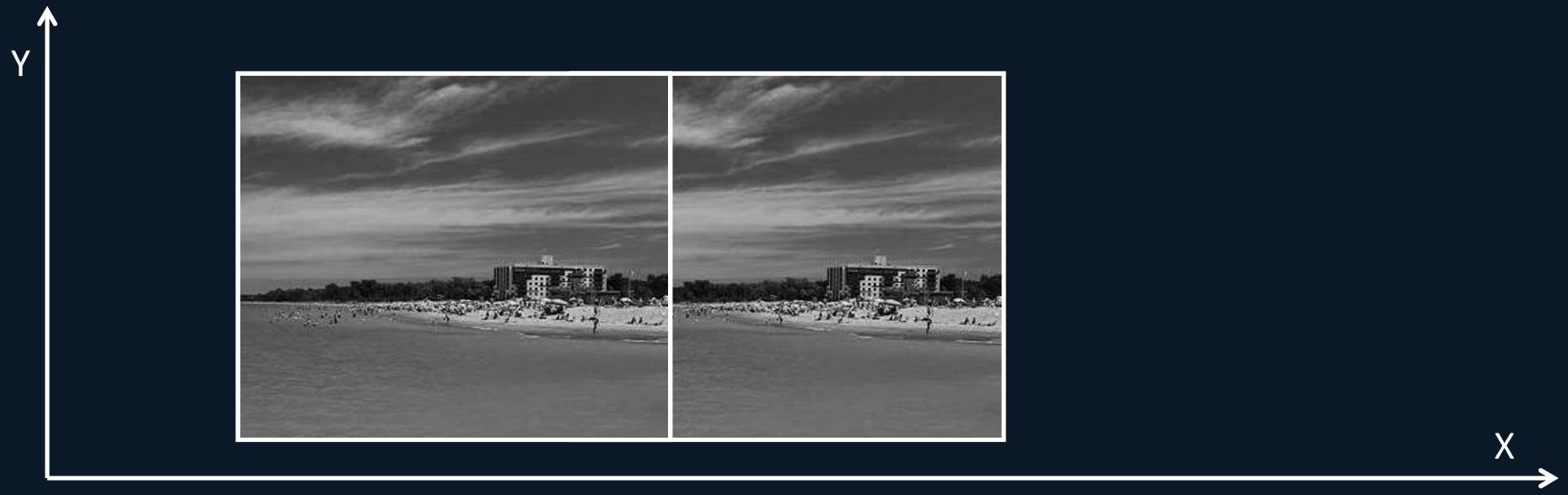


S = The score of the current alignment.



Temporarily slide the moving image a bit to the left.

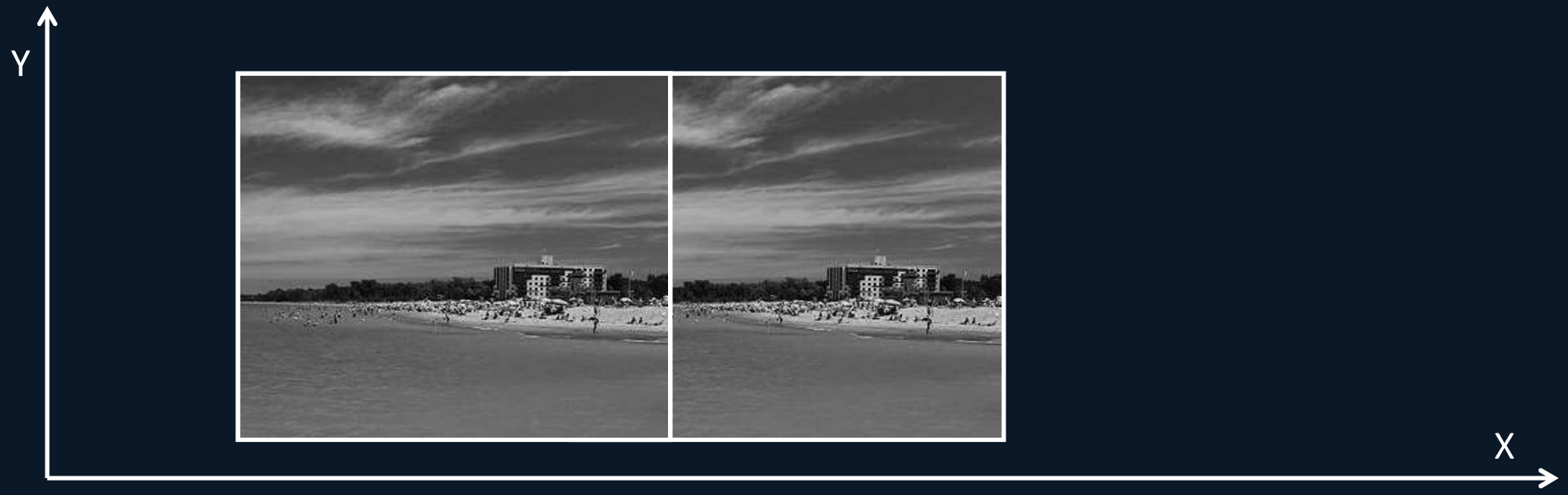
L = The score of the resulting alignment.



Put the moving image back where it was.

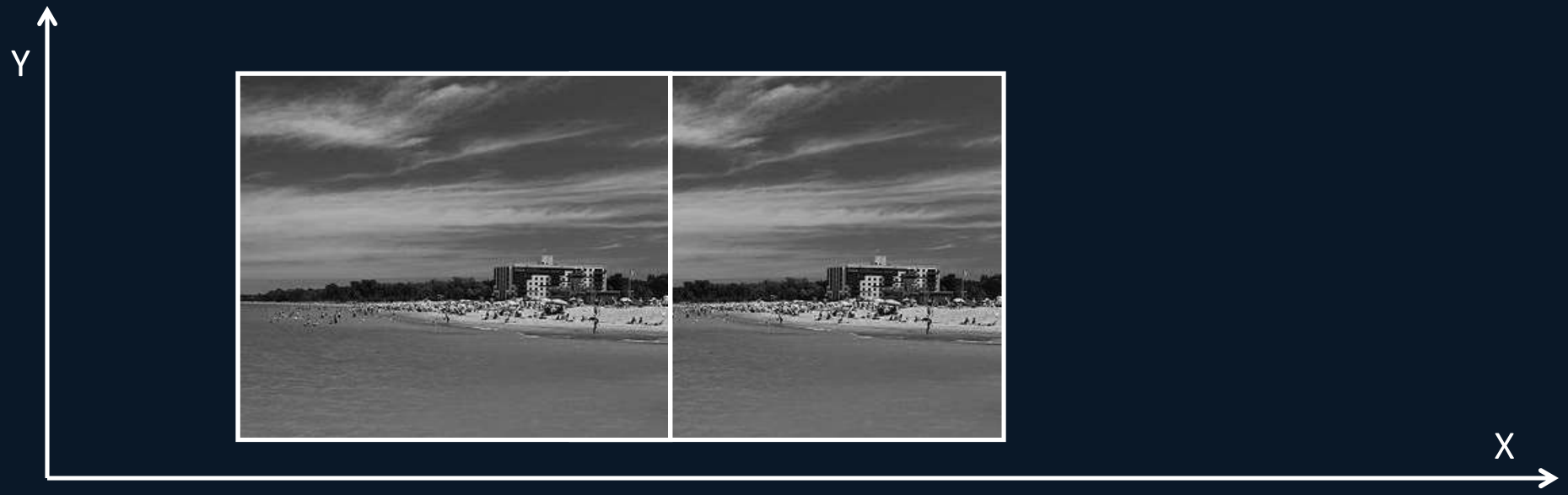
Temporarily slide the moving image a bit to the right.

R = The score of the resulting alignment.



Put the moving image back where it was.

Next, we decide which is the best way to slide the moving image based on the scores.



If L > S and R < S, then slide the image to the left.

If R > S and L < S, then slide the image to the right.

If L < S and R < S, then we have found the best spot; STOP.

If L > S and R < S, then slide the image to the left.

If R > S and L < S, then slide the image to the right.

If L < S and R < S, then we have found the best spot; STOP.

As we repeat the procedure, the moving image will continue sliding to the right one step at a time…

As we repeat the procedure, the moving image will continue sliding to the right one step at a time…

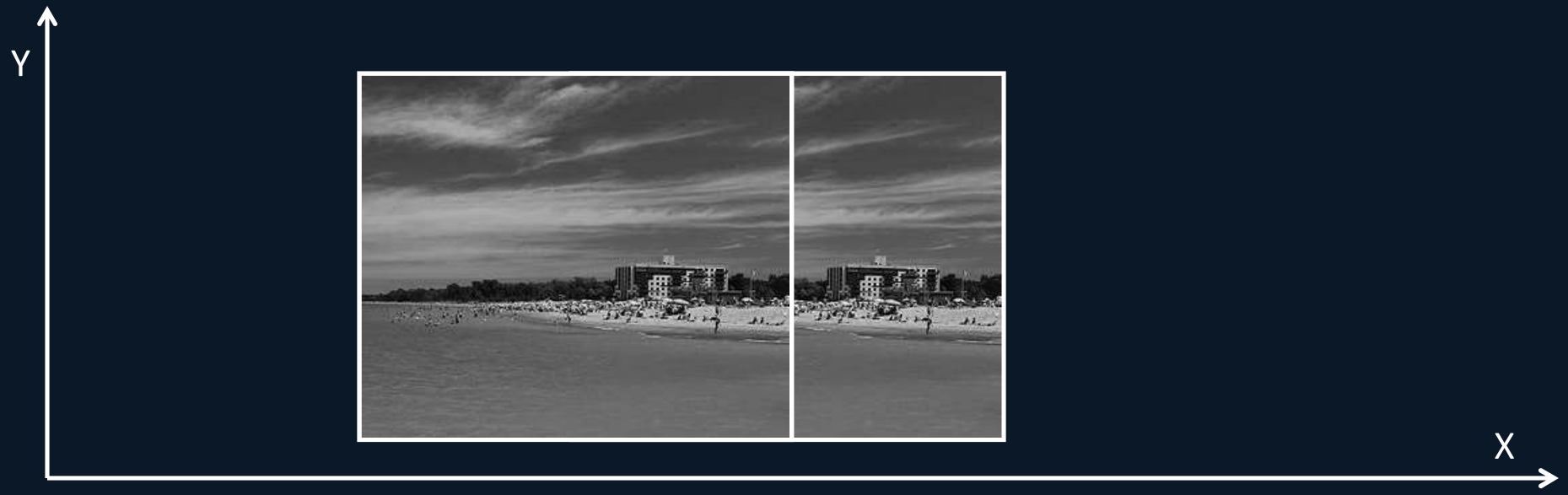
As we repeat the procedure, the moving image will continue sliding to the right one step at a time…

As we repeat the procedure, the moving image will continue sliding to the right one step at a time…

As we repeat the procedure, the moving image will continue sliding to the right one step at a time…

As we repeat the procedure, the moving image will continue sliding to the right one step at a time…

As we repeat the procedure, the moving image will continue sliding to the right one step at a time…

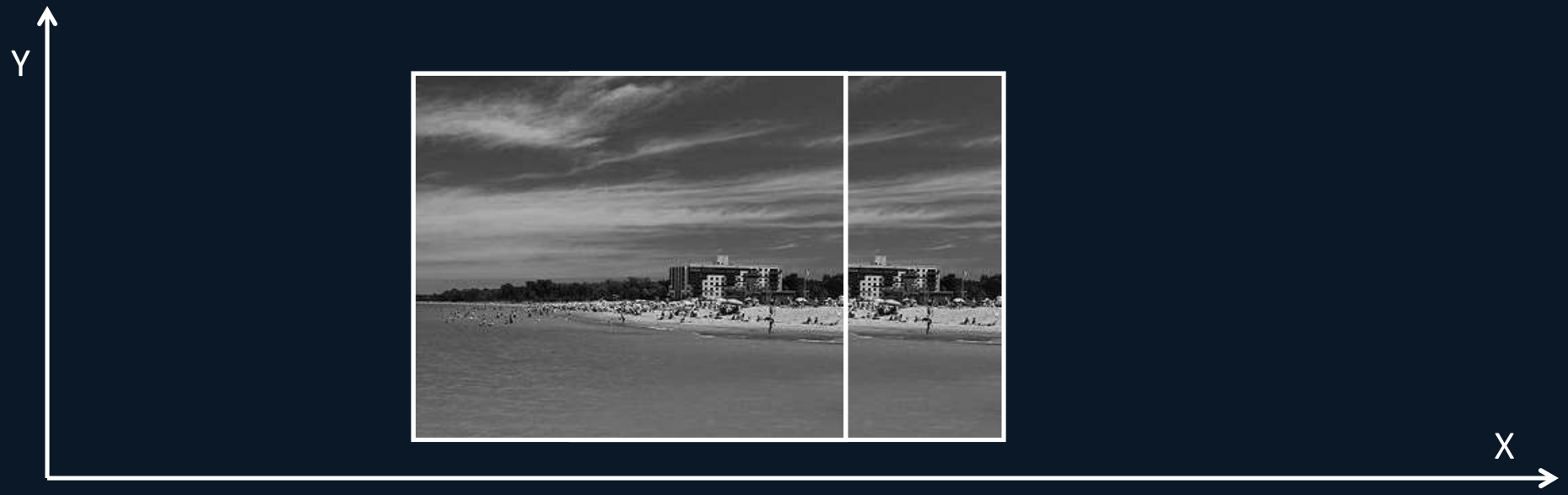


As we repeat the procedure, the moving image will continue sliding to the right one step at a time…

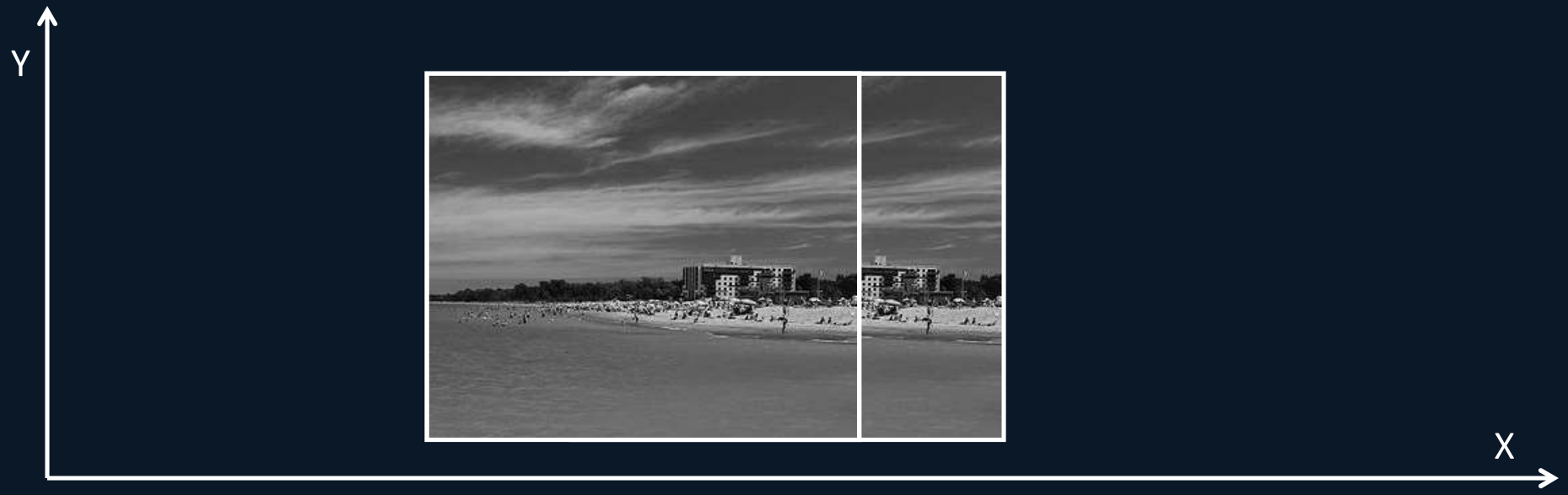
As we repeat the procedure, the moving image will continue sliding to the right one step at a time…

As we repeat the procedure, the moving image will continue sliding to the right one step at a time…

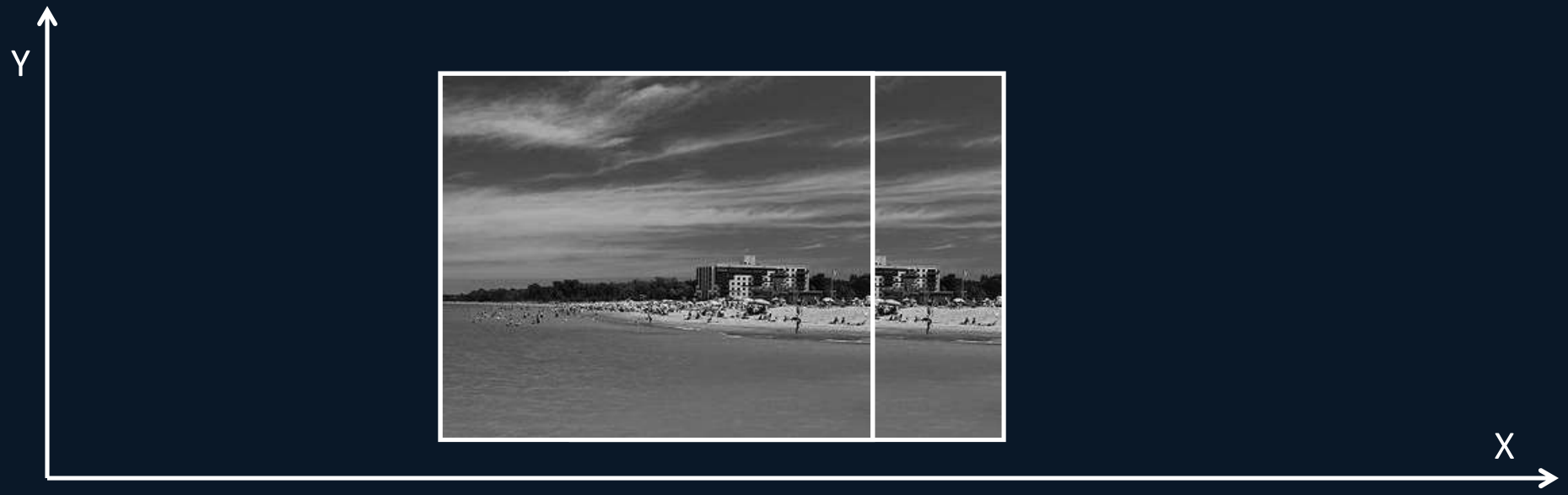
As we repeat the procedure, the moving image will continue sliding to the right one step at a time…



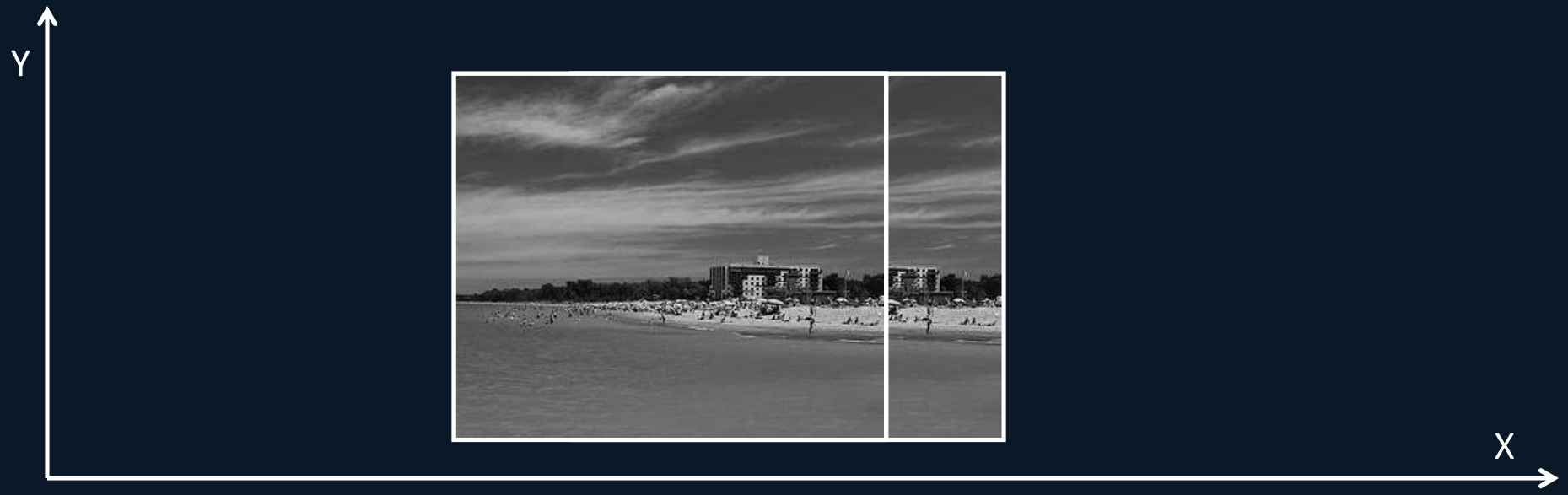
As we repeat the procedure, the moving image will continue sliding to the right one step at a time…



As we repeat the procedure, the moving image will continue sliding to the right one step at a time…



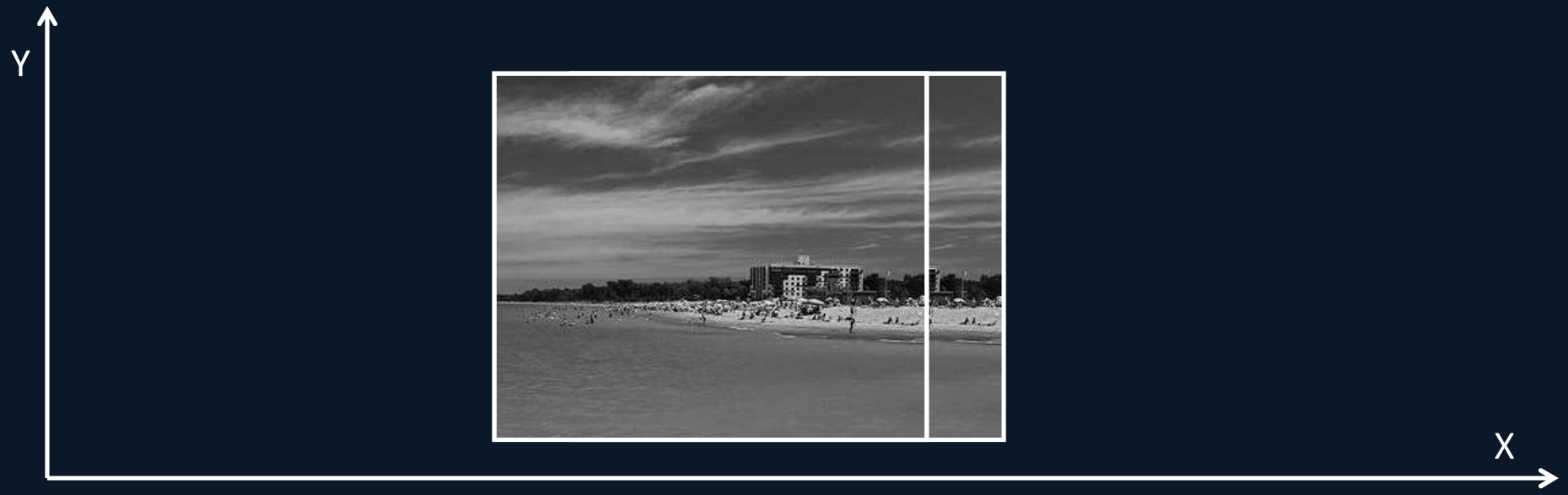
As we repeat the procedure, the moving image will continue sliding to the right one step at a time…



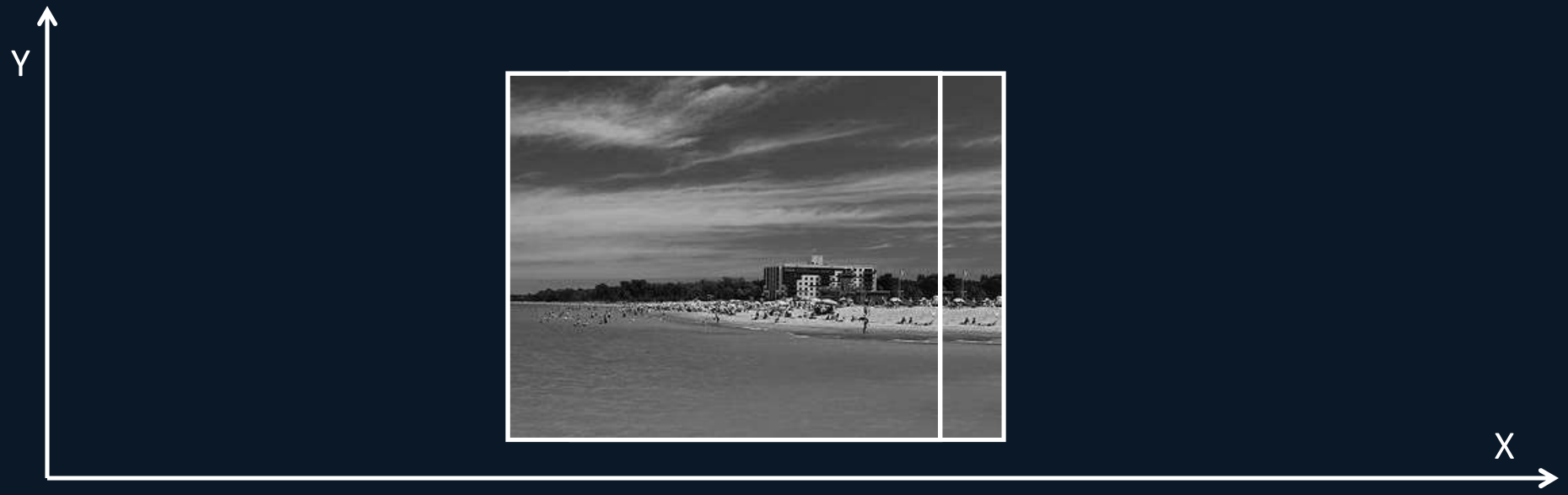
As we repeat the procedure, the moving image will continue sliding to the right one step at a time…

As we repeat the procedure, the moving image will continue sliding to the right one step at a time…

As we repeat the procedure, the moving image will continue sliding to the right one step at a time…

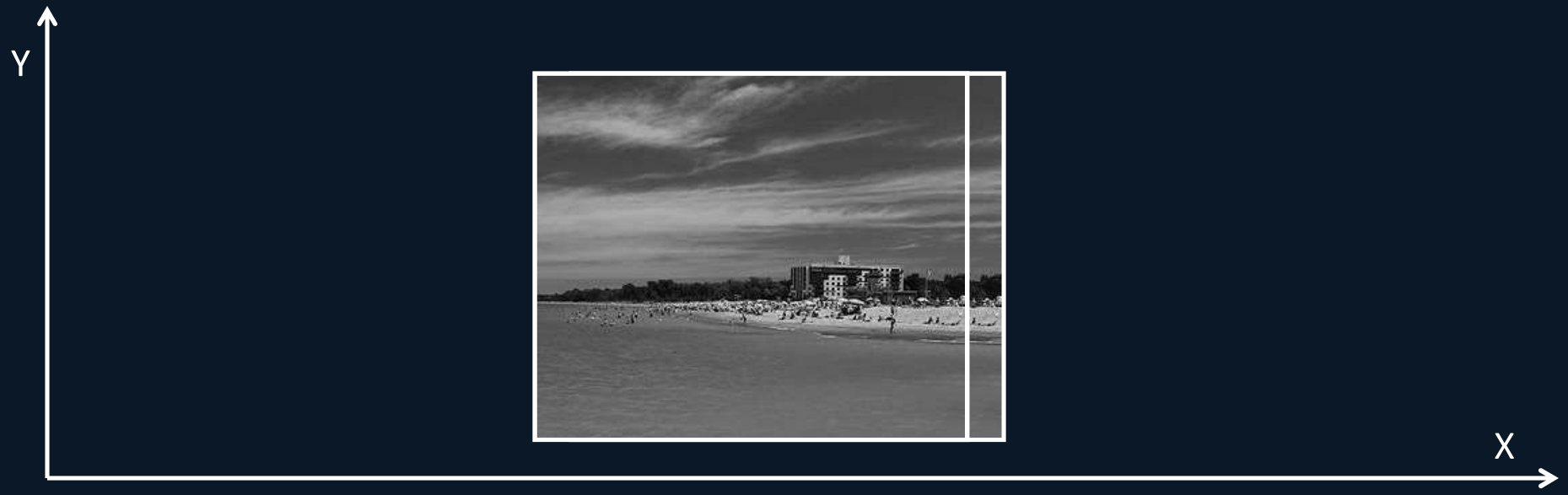


As we repeat the procedure, the moving image will continue sliding to the right one step at a time…

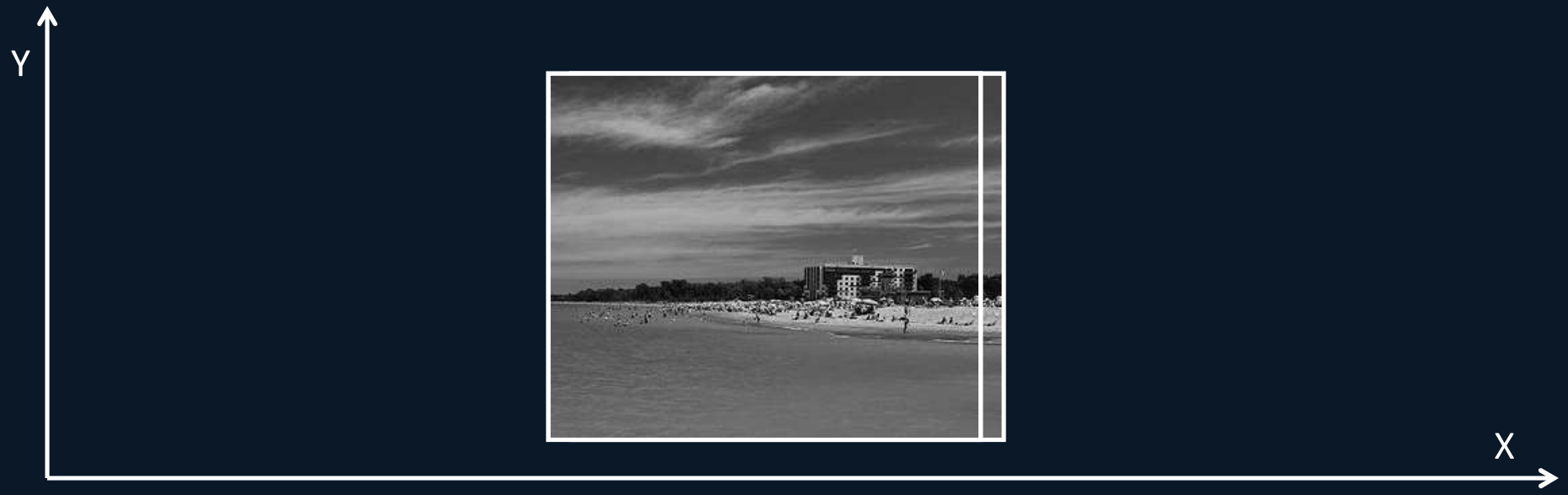


As we repeat the procedure, the moving image will continue sliding to the right one step at a time…

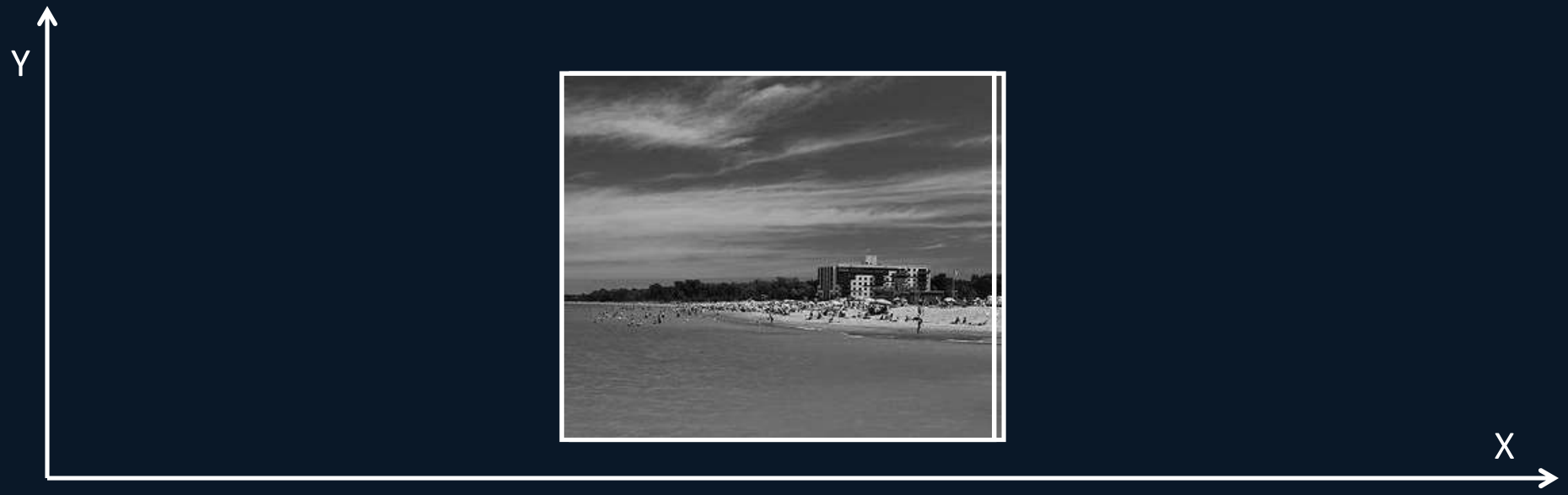
As we repeat the procedure, the moving image will continue sliding to the right one step at a time…



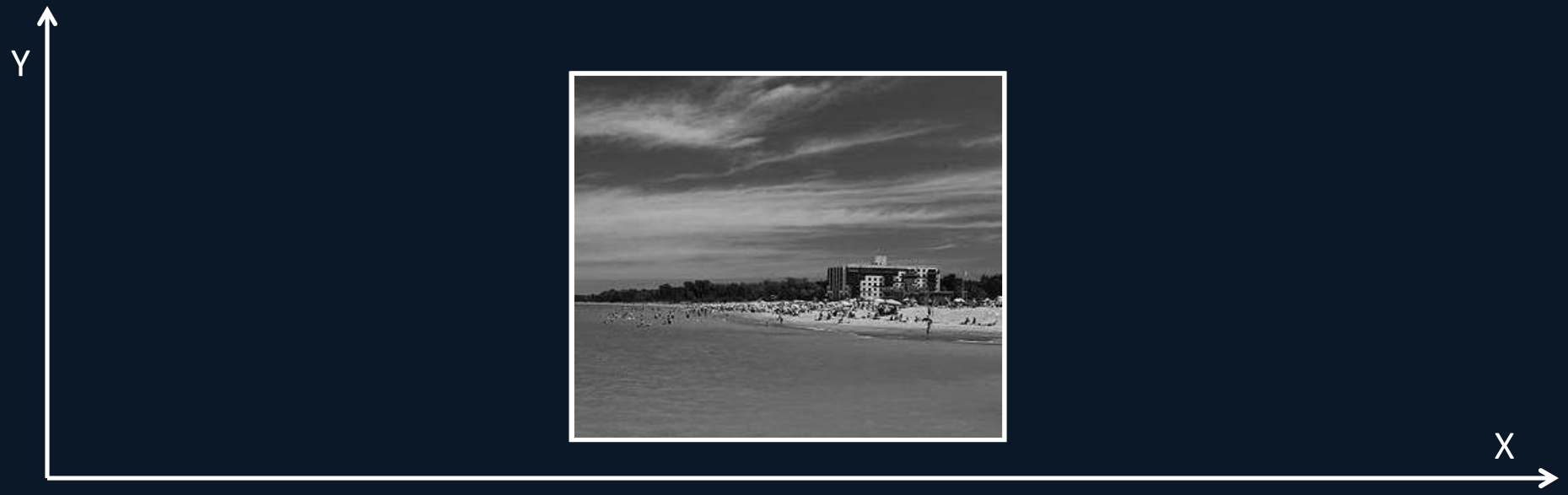
As we repeat the procedure, the moving image will continue sliding to the right one step at a time…



As we repeat the procedure, the moving image will continue sliding to the right one step at a time…

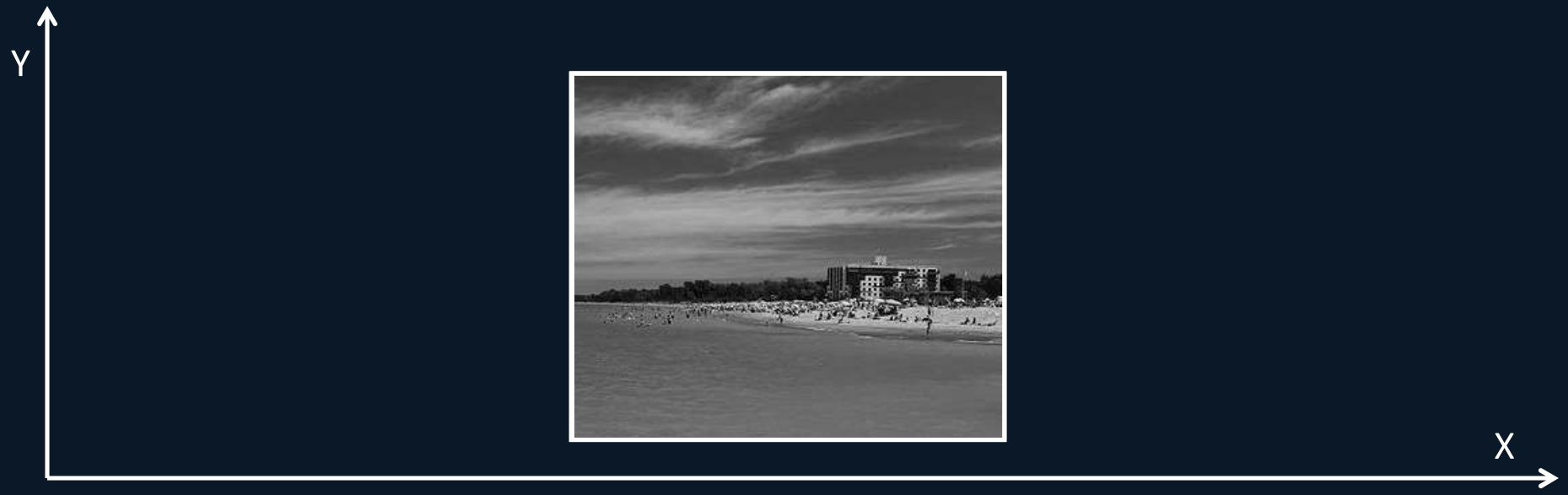


As we repeat the procedure, the moving image will continue sliding to the right one step at a time…



As we repeat the procedure, the moving image will continue sliding to the right one step at a time…
…until it comes into alignment with the fixed image.

At this point, either a move to the left or a move to the right will make the alignment worse, so the algorithm stops.



If L > S and R < S, then slide the image to the left.

If R > S and L < S, then slide the image to the right.

If L < S and R < S, then we have found the best spot; STOP.

Registration-by-X-translation algorithm

Repeat the following:

S = The score of the current alignment.

Temporarily slide the moving image a bit to the left.
L = The score of the resulting alignment.

Temporarily slide the moving image a bit to the right.
R = The score of the resulting alignment.

If L > S and R < S, then slide the image to the left.
If R > S and L < S, then slide the image to the right.
If L < S and R < S, then we have found the best spot; STOP.

Registration-by-X-translation algorithm

Repeat the following:

S = The score of the current alignment.

L = Score (left).

R = Score (right).

If L > S and R < S then go left.

If R > S and L < S then go right.

If L < S and R < S then STOP.

Now imagine that the images are vertically misaligned as well. We now need to adapt the algorithm to deal with alignment in the Y direction.

Registration-by-X-and-Y-translation algorithm

Repeat the following:

S = The score of the current alignment.

L = Score (left). R = Score (right).
U = Score (up). D = Score (down).

If L > S and R < S and U < S and D < S then go left.
If R > S and L < S and U < S and D < S then go right.
If U > S and L < S and R < S and D < S then go up.
If D > S and L < S and R < S and U < S then go down.
If L < S and R < S and U < S and D < S then STOP.

The algorithm is starting to get complicated and messy, and we haven’t even adapted it to allow for stepping in diagonal directions (as we should).

What happens when we add a rotational misalignment as well?

And a scale misalignment?

And an anisotropic scale misalignment?

And a skew?

A new way of thinking about algorithms

We are used to seeing algorithms written down as a finite series of steps to accomplish a task.

This works but quickly becomes very complicated to read and write (and these are very simple examples!).

The research community has gravitated toward an alternative way of describing registration and segmentation algorithms that is compact and consistent from one algorithm to the next.

A new way of thinking about algorithms

This viewpoint is based on the following observations:

1. The algorithm is working toward achieving some goal, or *objective*.

2. To achieve the objective, the algorithm must find a best, or *optimal* number of set of numbers; the *solution*.

3. The algorithm must have a *search strategy* for trying various candidate solutions to find the answer.

A new way of thinking about algorithms

These observations resolve to three specifications that we can use to fully define the algorithm:

1. The *objective function*, which computes the goodness of any given candidate solution.

2. The rules governing the solution – the *solution space*. How many numbers are in the solution, what ranges are legal for them, and what do they mean?

3. The *optimizer*, which itself is an algorithm for searching the solution space for the optimal solution.

A new way of thinking about algorithms

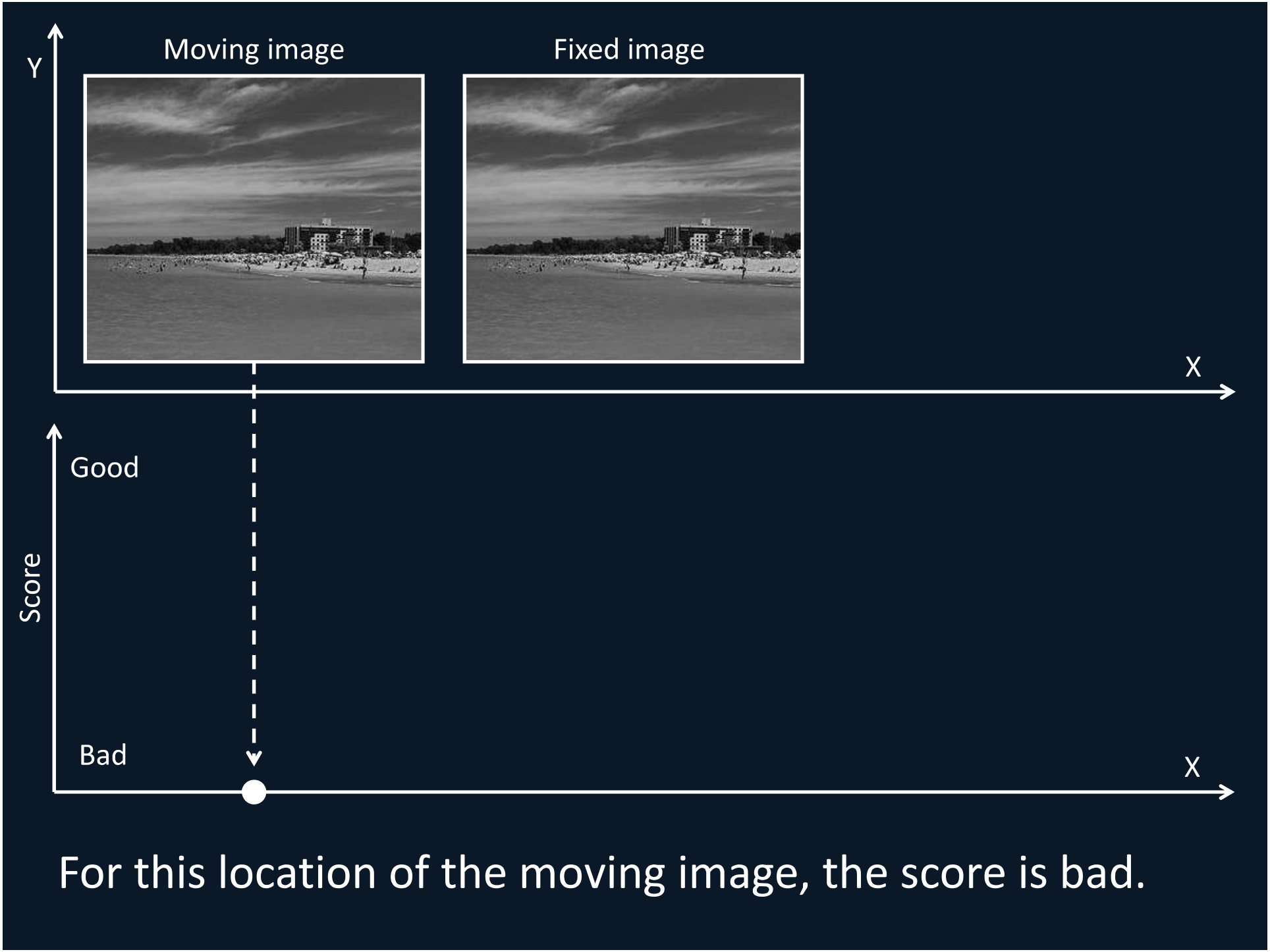
We can think of our simple registration algorithm in these terms as follows:

1. Objective function: Our “goodness” score.

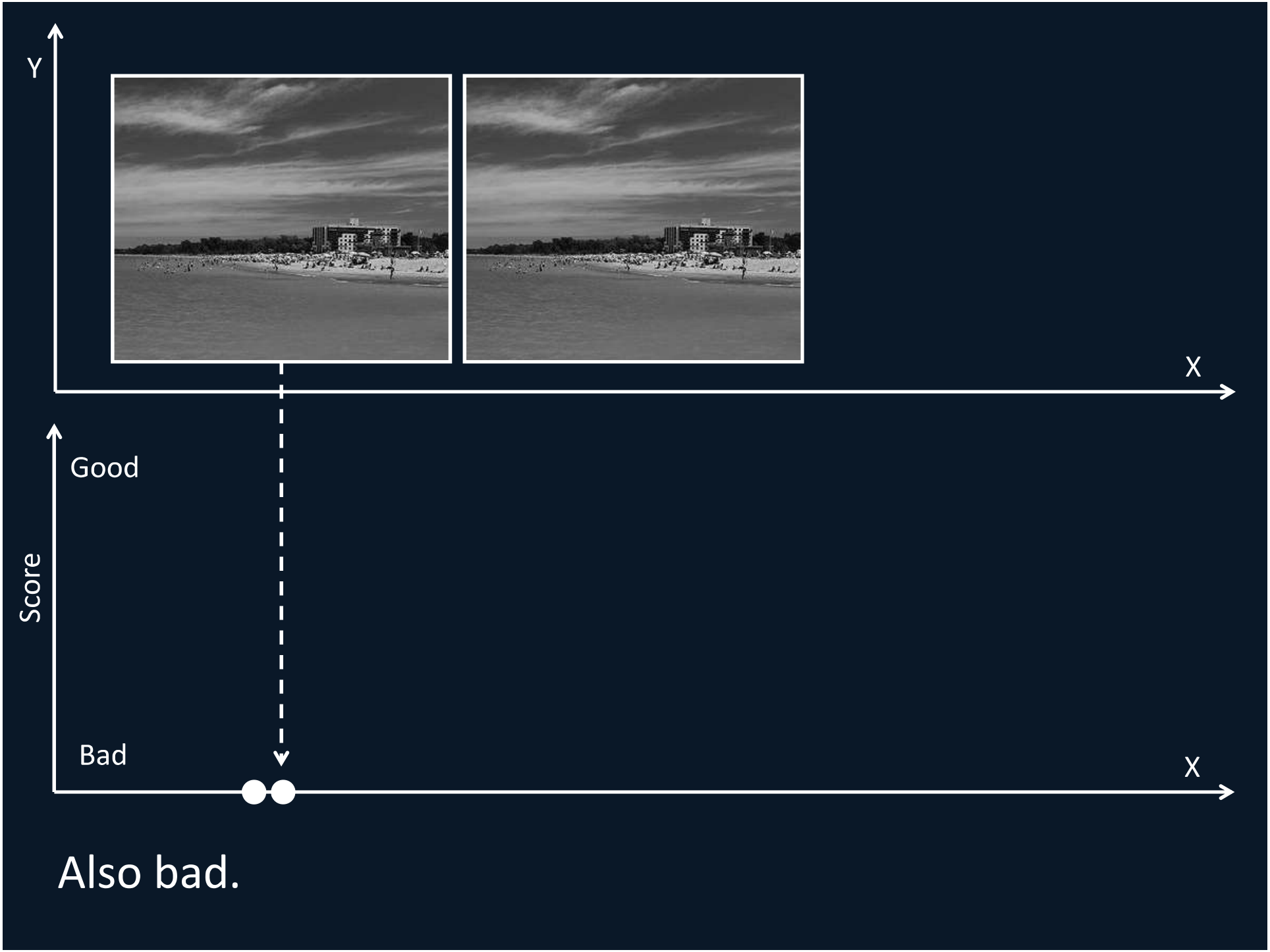
2. Solution space: One number indicating the amount of translation along the X axis. Negative values go left, positive values go right.

3. Optimizer: Gradient ascent*. ← ?

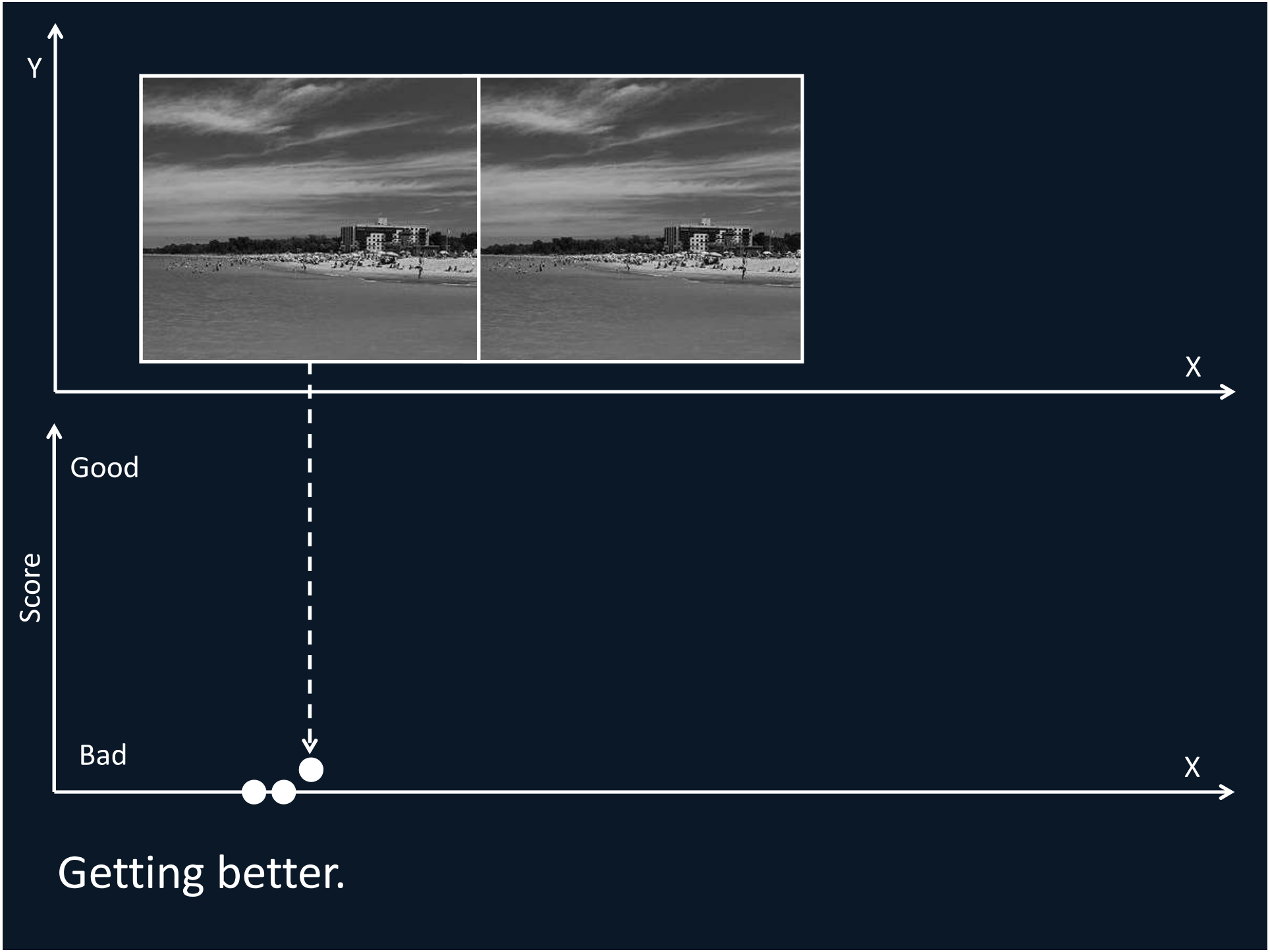
* For illustrative purposes our algorithm implemented a simplification of gradient ascent.



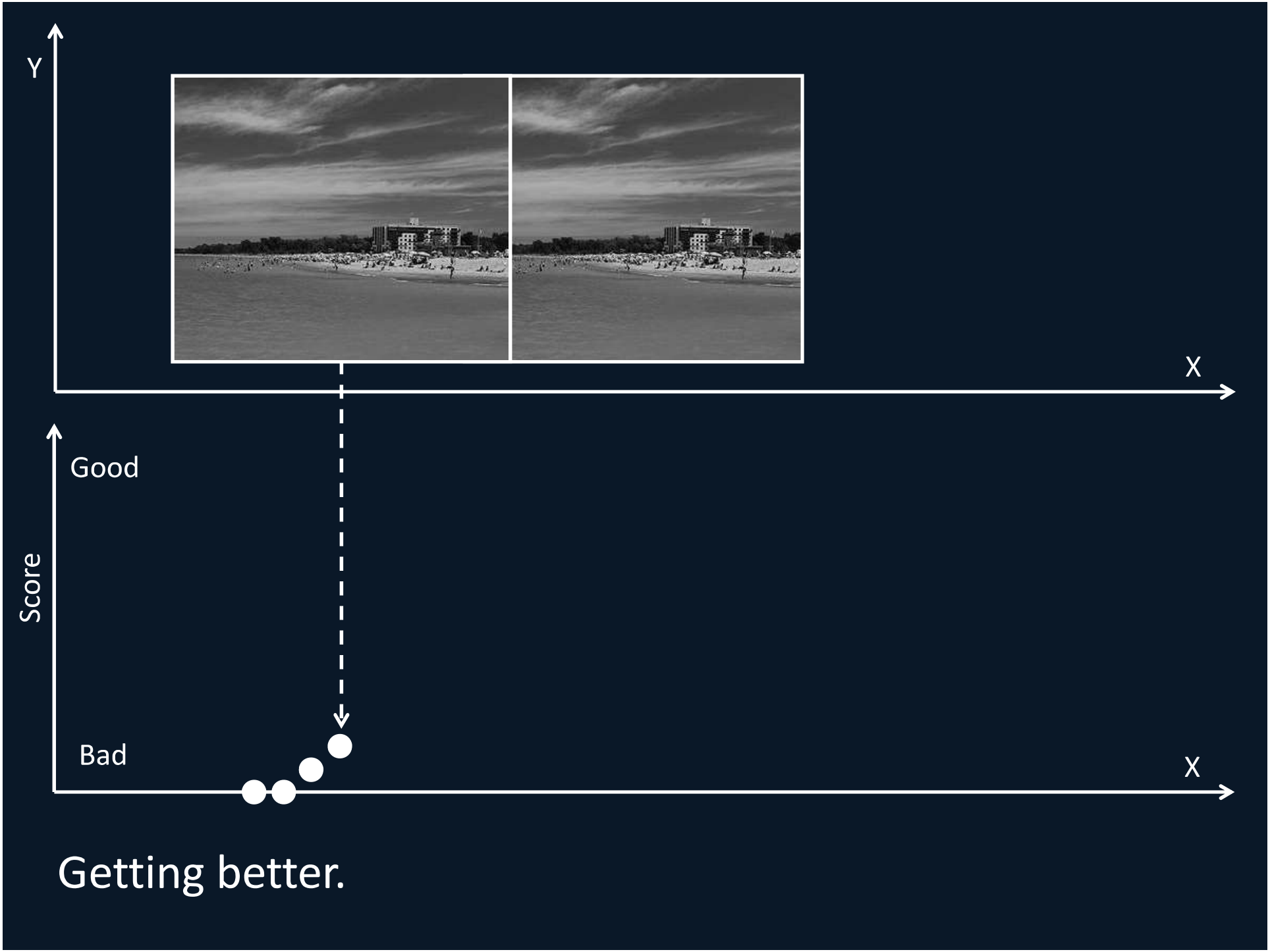
For this location of the moving image, the score is bad.

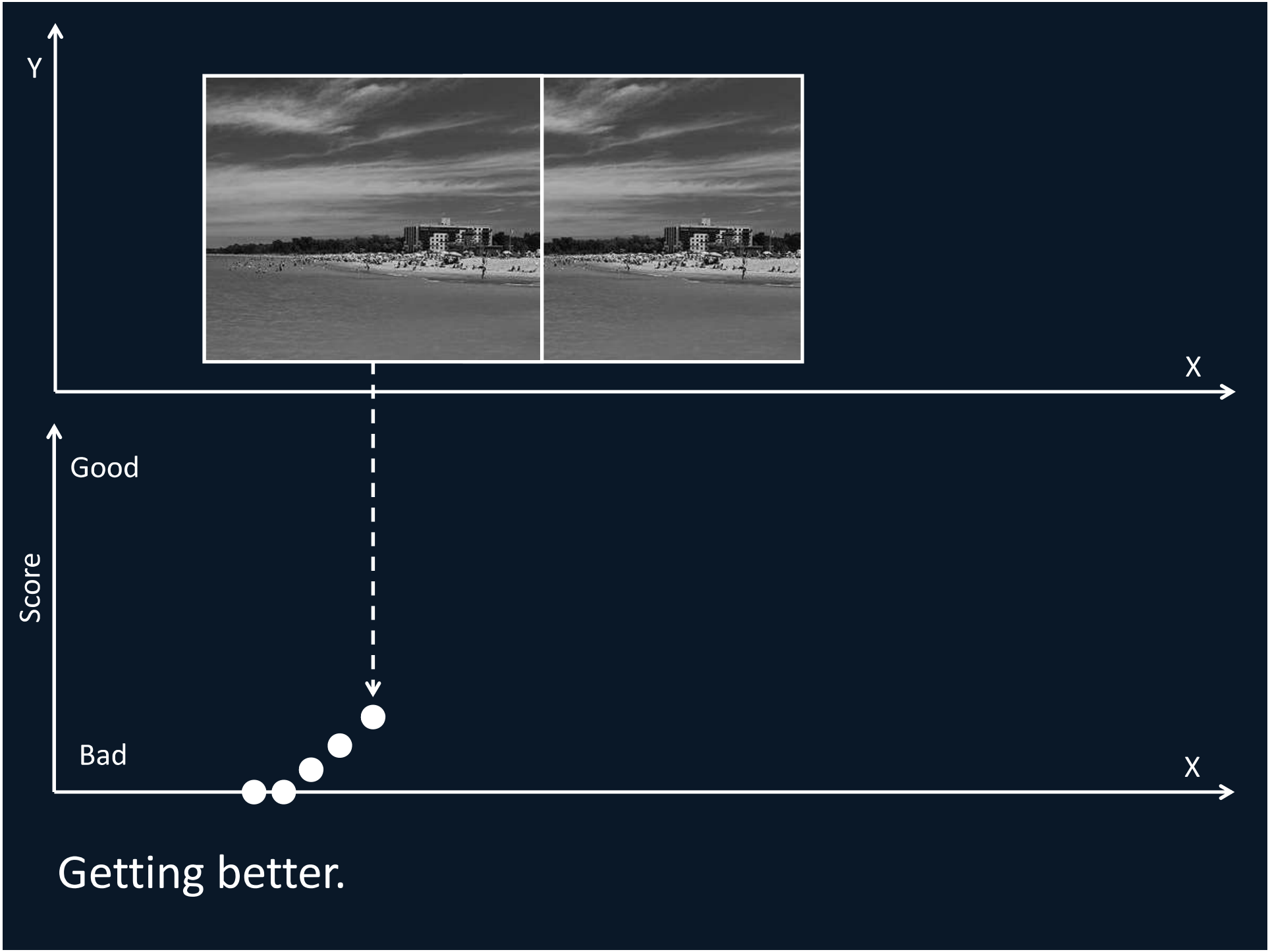


Also bad.

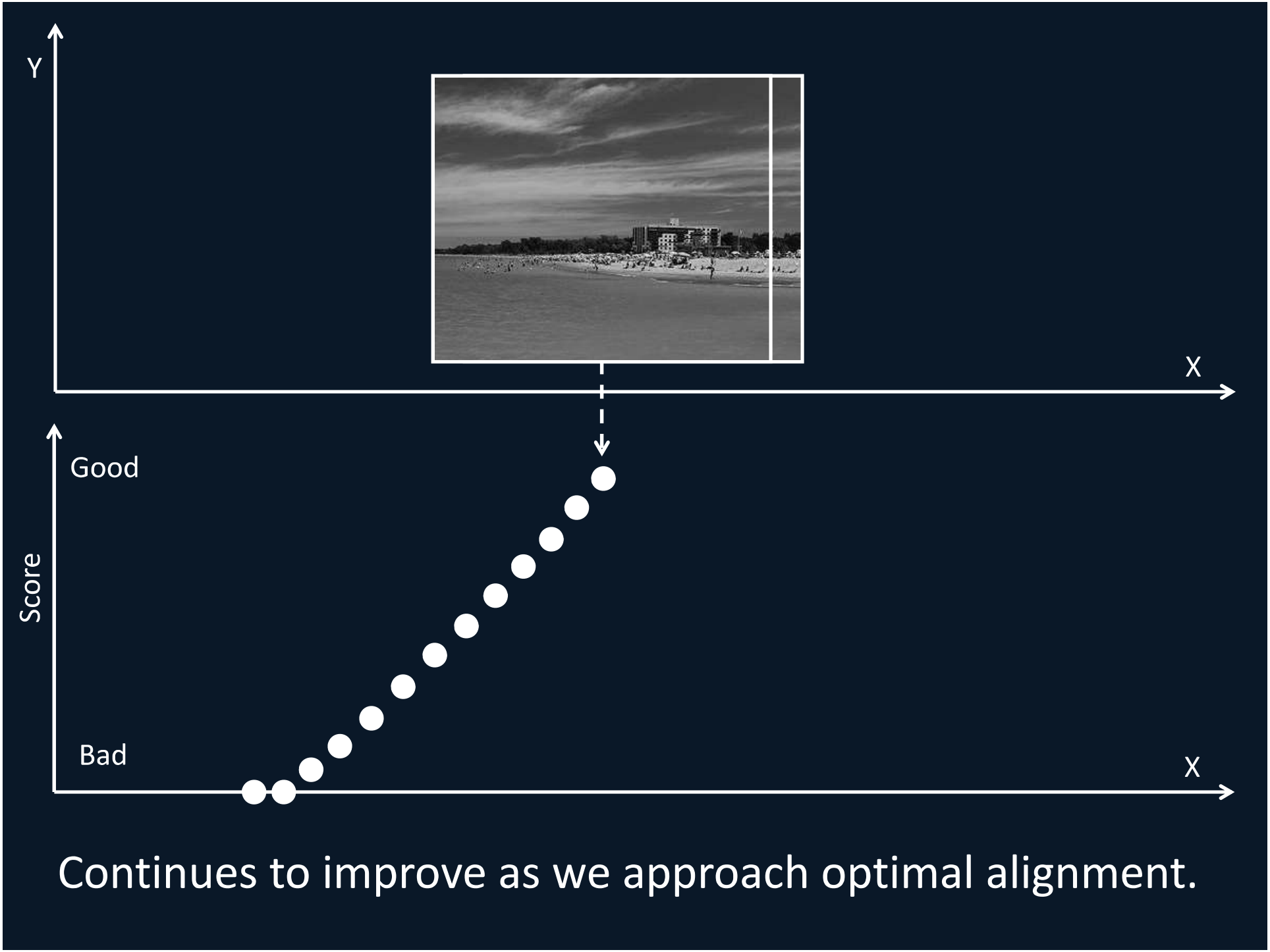


Getting better.

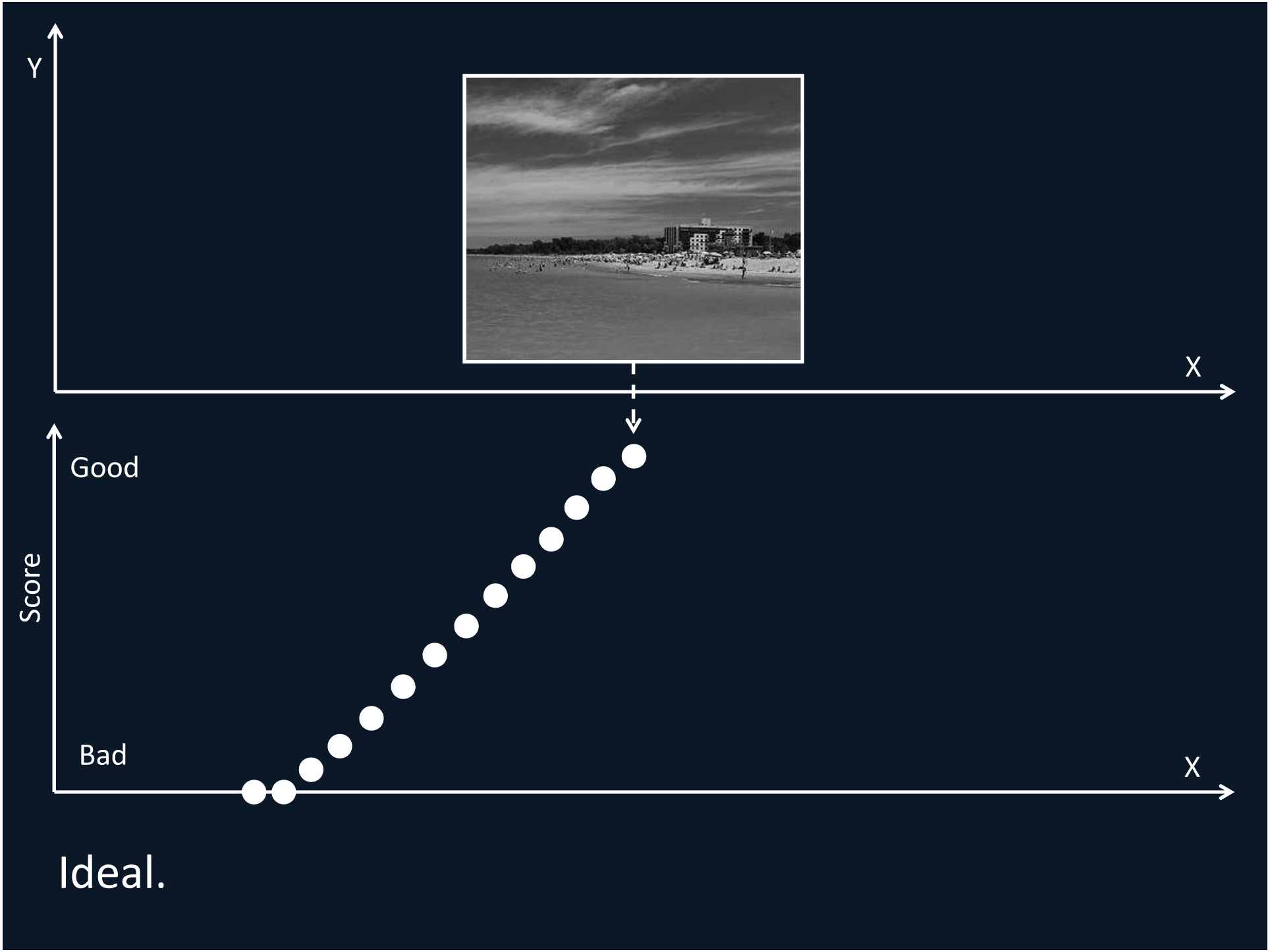

Y
X
Good
Score
Bad
X
Getting better.



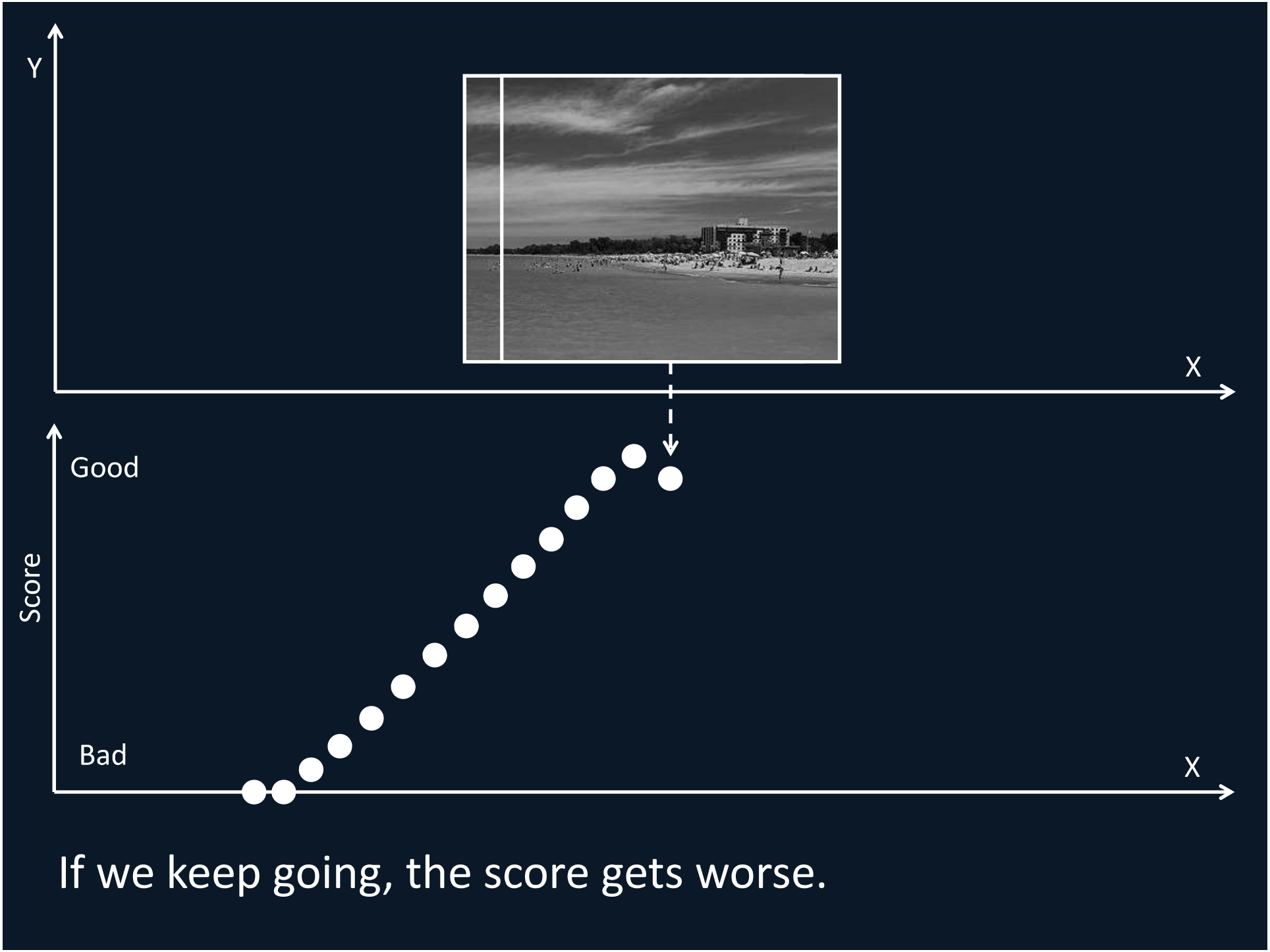
Getting better.



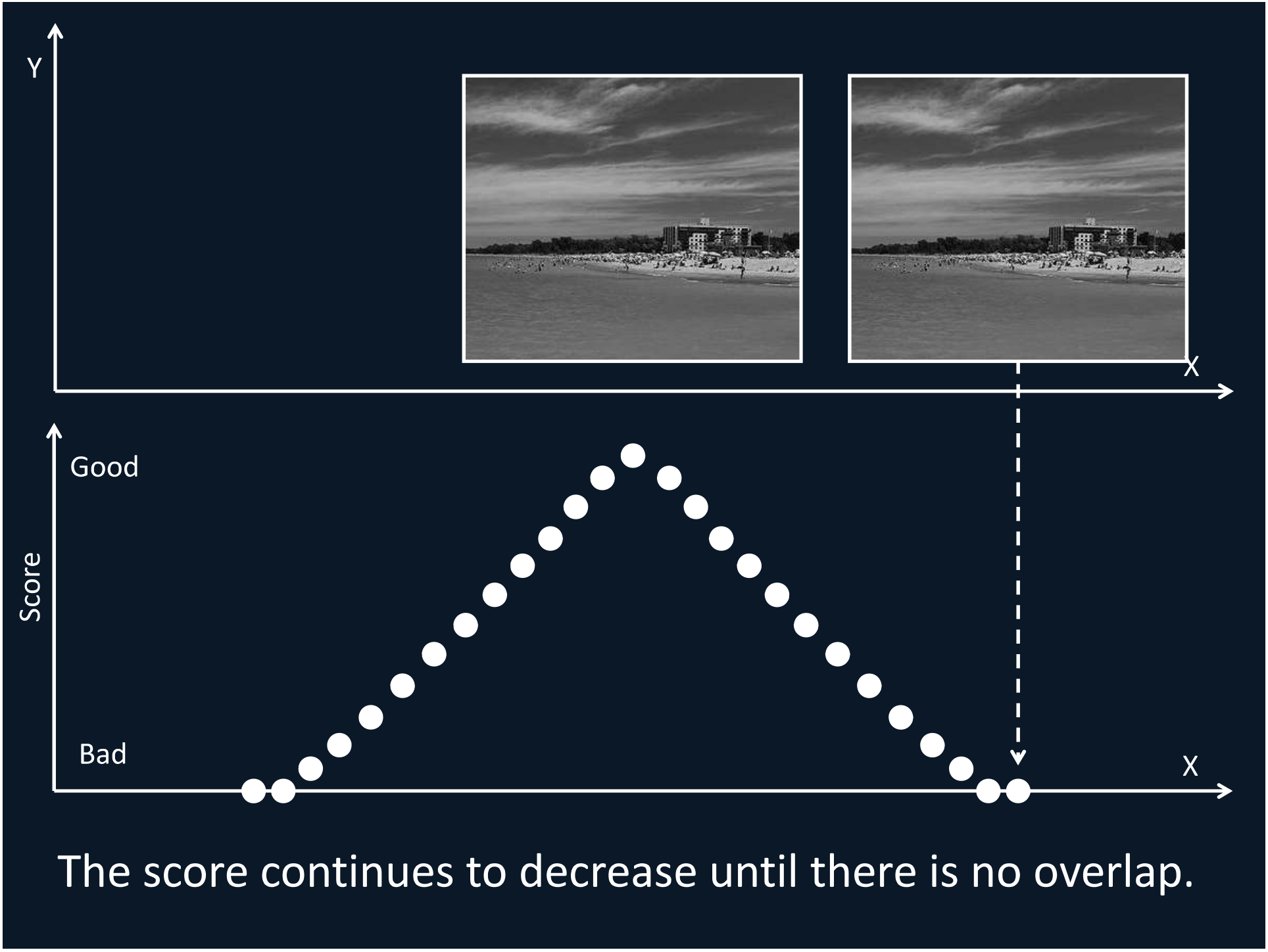
Continues to improve as we approach optimal alignment.



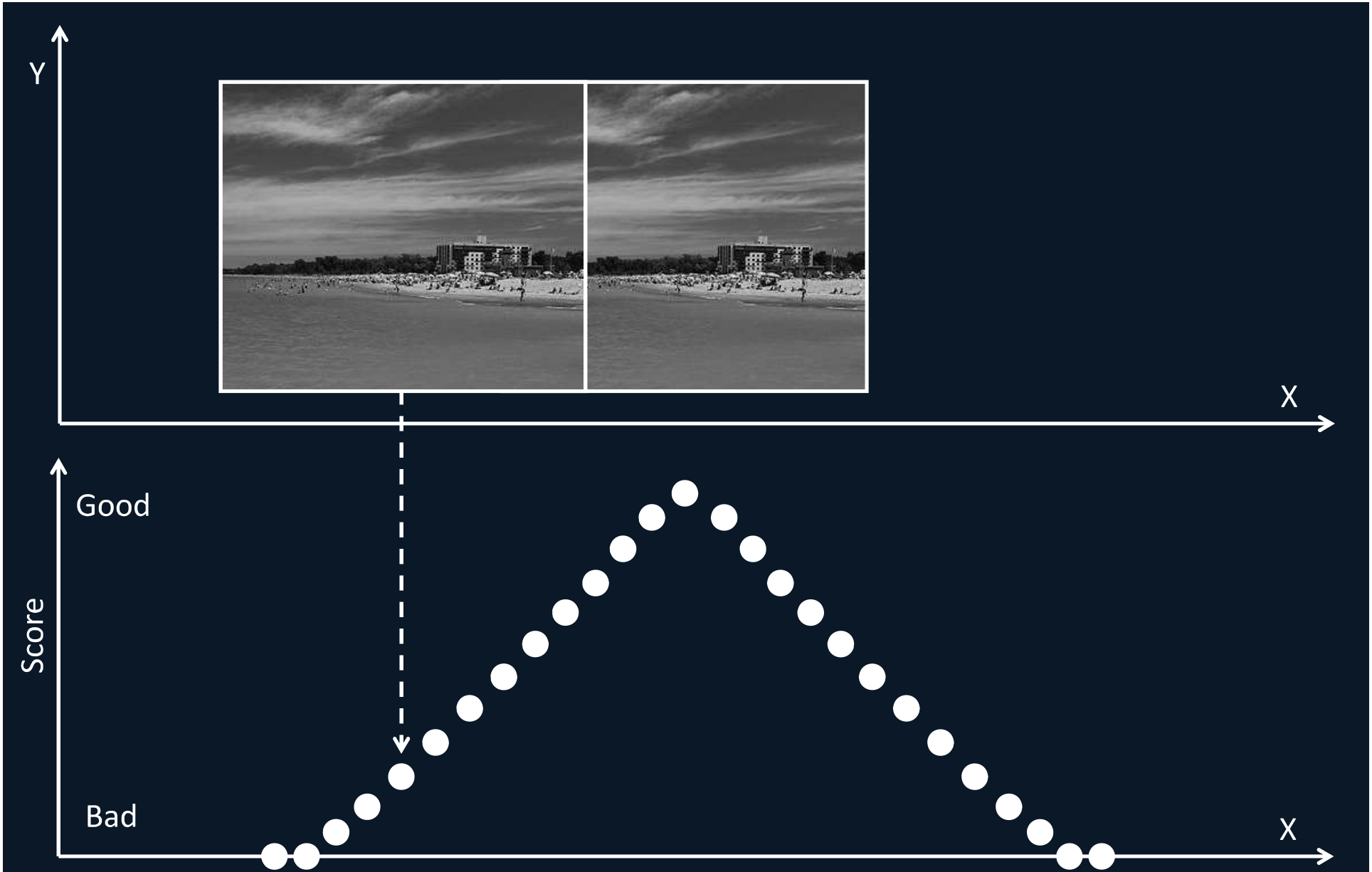
Ideal.



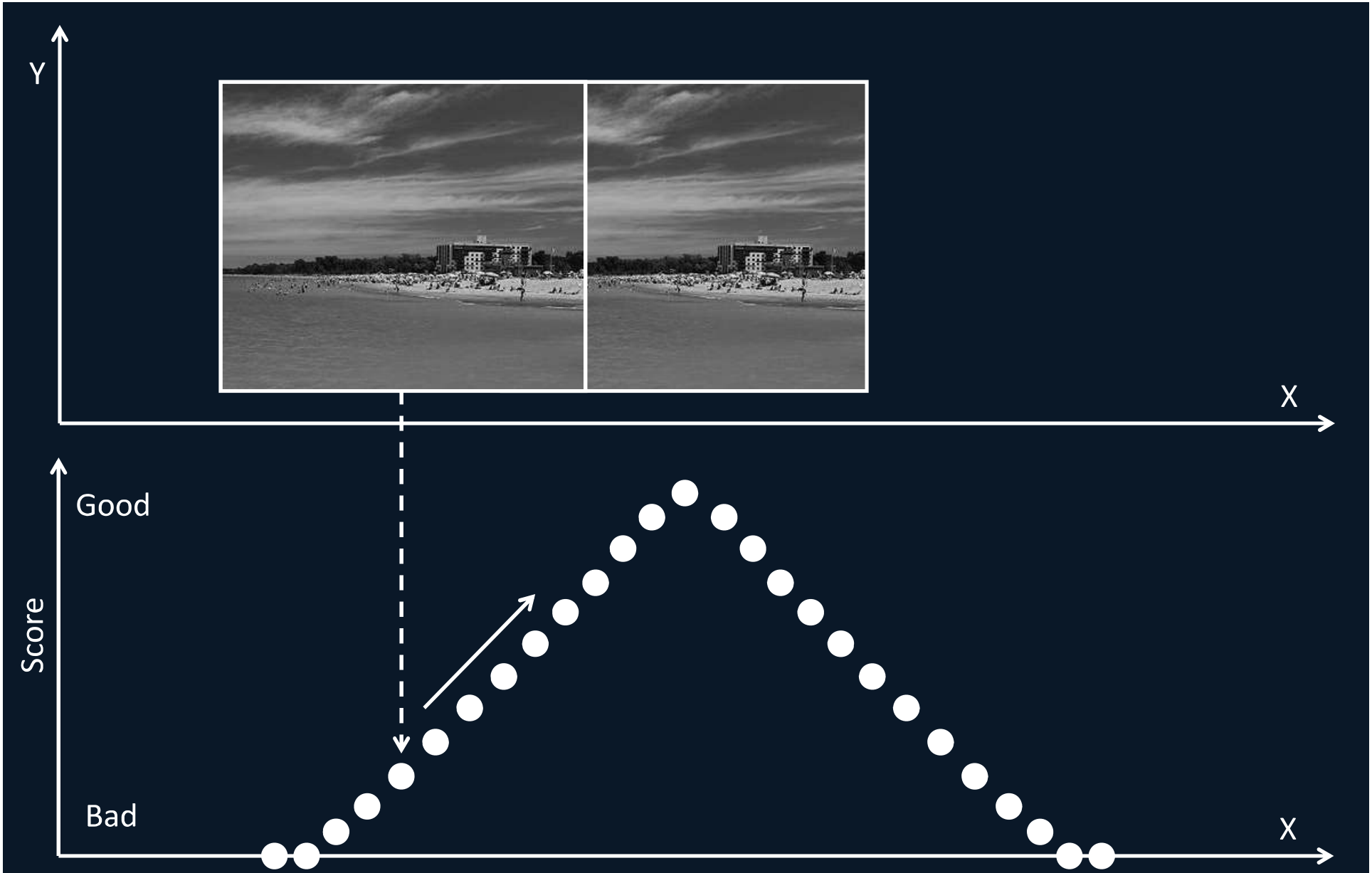
If we keep going, the score gets worse.



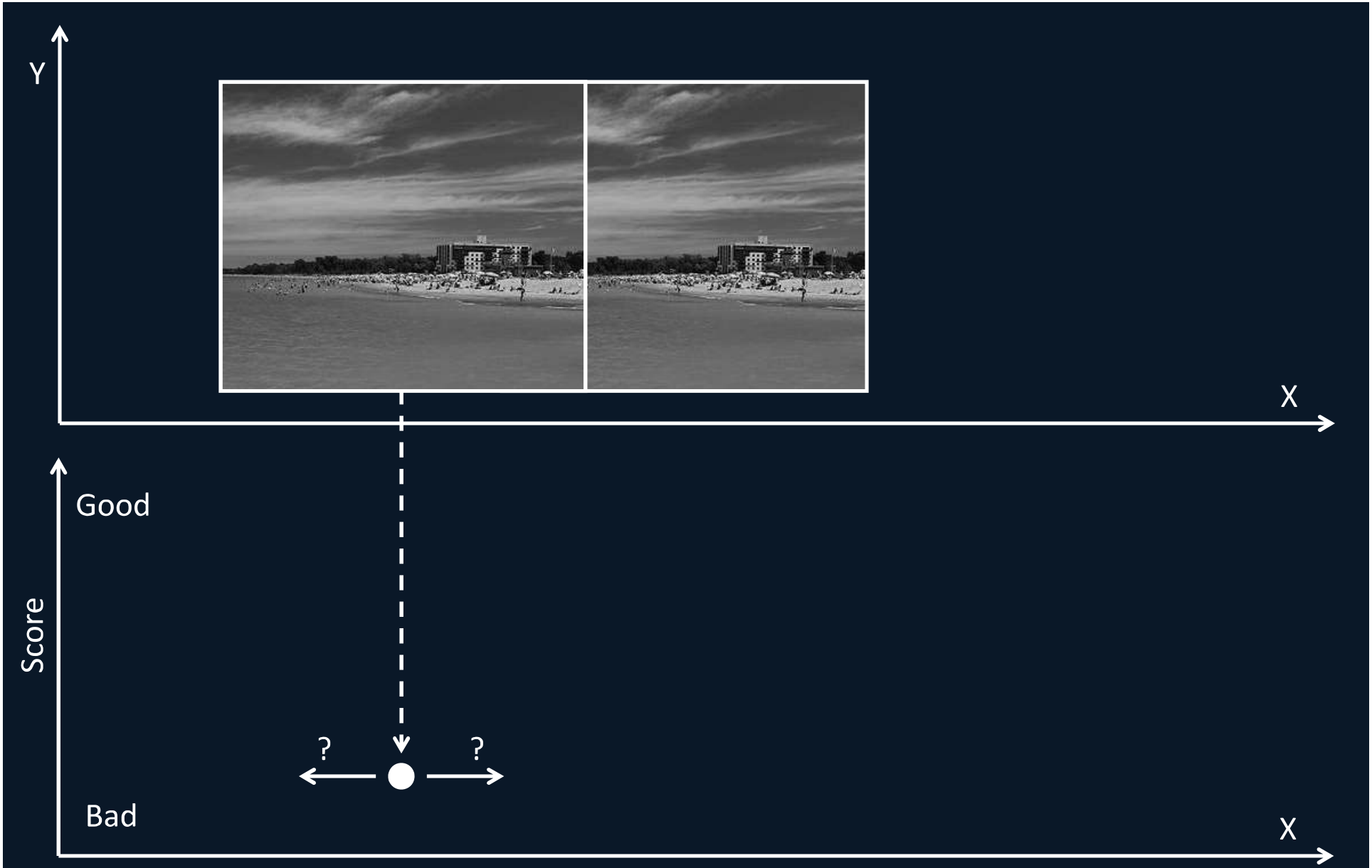
The score continues to decrease until there is no overlap.



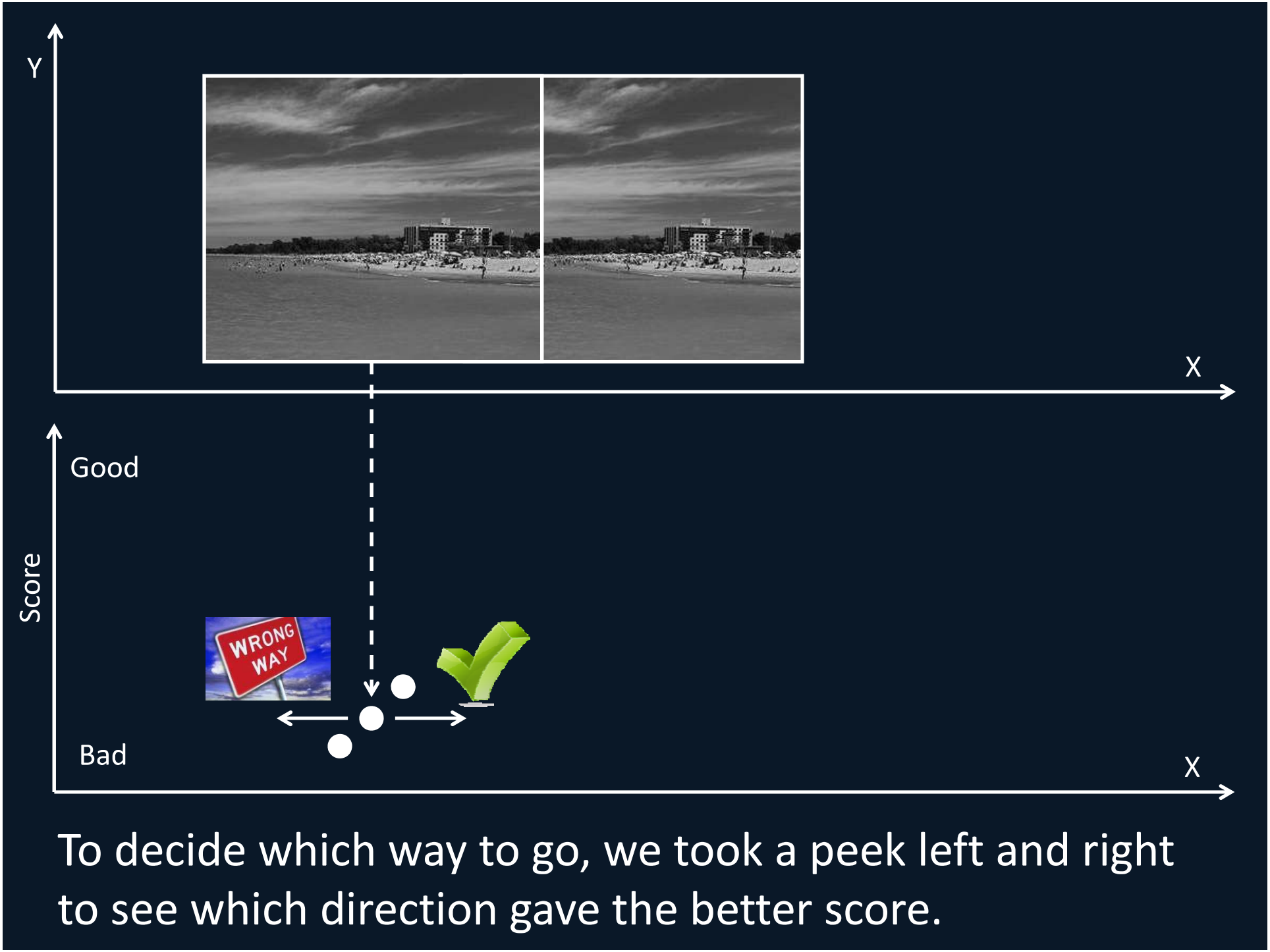
Suppose that this was the initial situation we started with before registration.



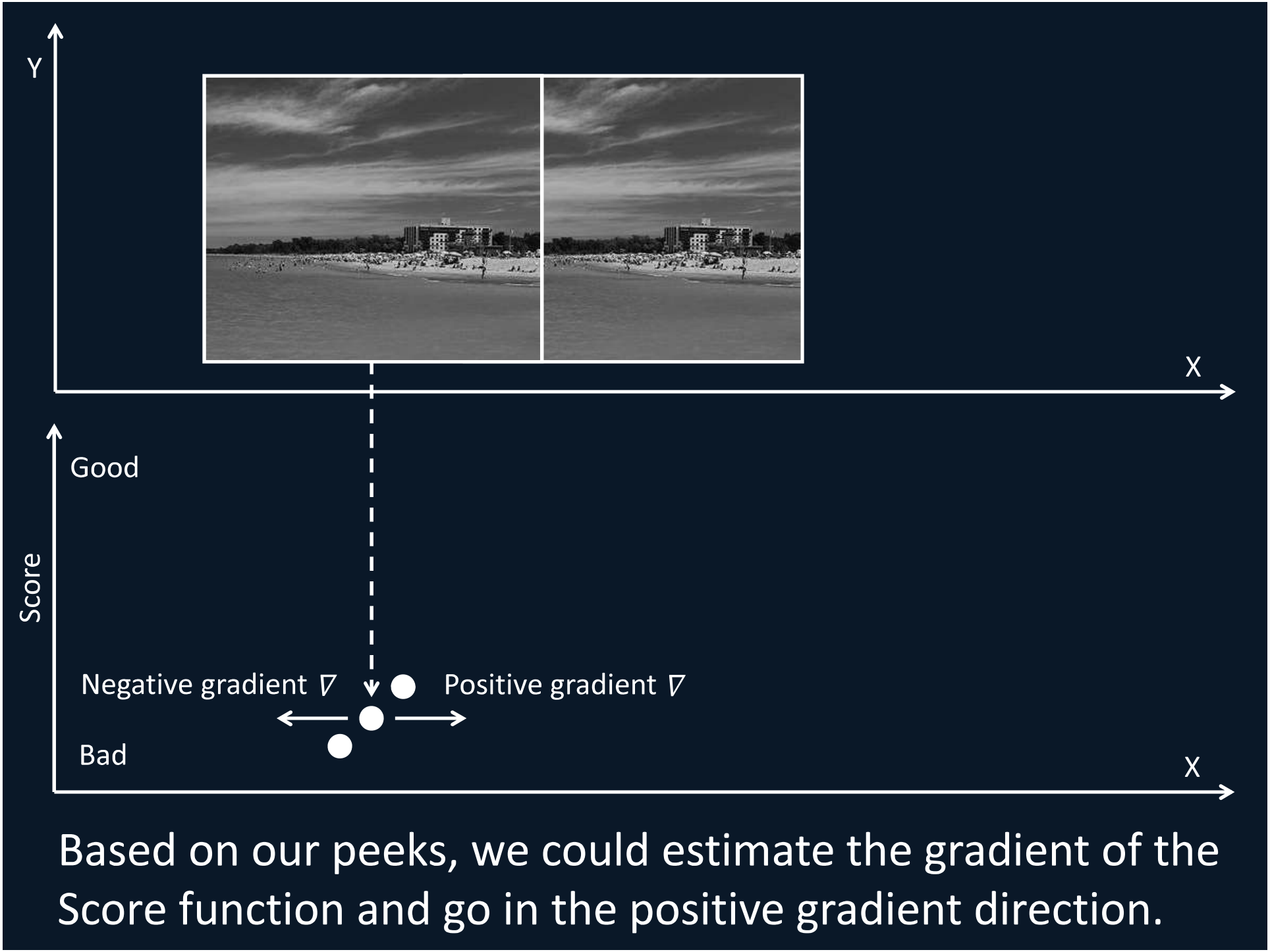
We have a hill to climb. We want to go up the gradient until we find the peak: *gradient ascent* optimization.



But remember, initially, we only have the score about our current location.



To decide which way to go, we took a peek left and right to see which direction gave the better score.



Based on our peeks, we could estimate the gradient of the Score function and go in the positive gradient direction.

A new way of thinking about algorithms

We can think of our simple registration algorithm in these terms as follows:

1. Objective function: Our “goodness” score.

2. Solution space: One number indicating the amount of translation along the X axis. Negative values go left, positive values go right.

3. Optimizer: Gradient ascent. 

Gradient ascent optimization

3. Optimizer: Gradient ascent.

So, gradient ascent is an optimization scheme that computes the “next best guess” solution with the following two steps:

I. Estimate the gradient of the objective function at the current point.

II. Take a step in the solution space in the direction of greatest positive gradient.

Gradient ascent optimization

3. Optimizer: Gradient ascent.

This is typically written down like this:

$$\hat{x} = x + \nabla F(x)$$

Next step in solution space

Current location in solution space

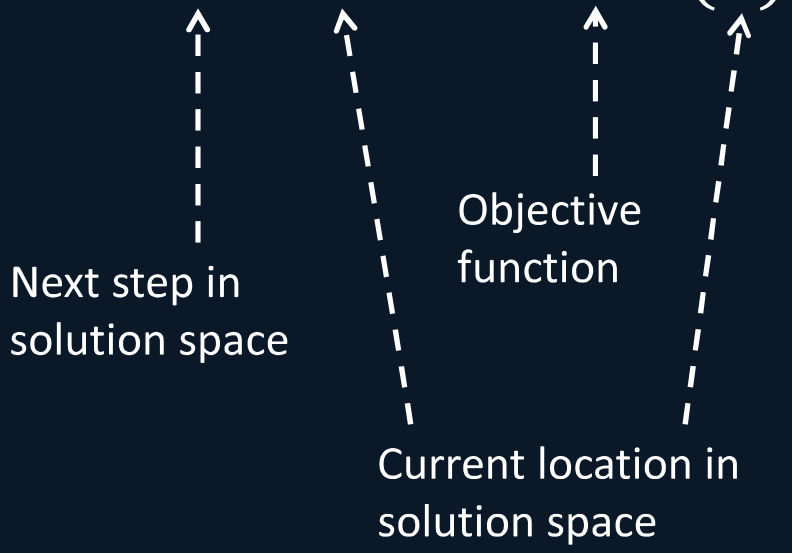
Objective function

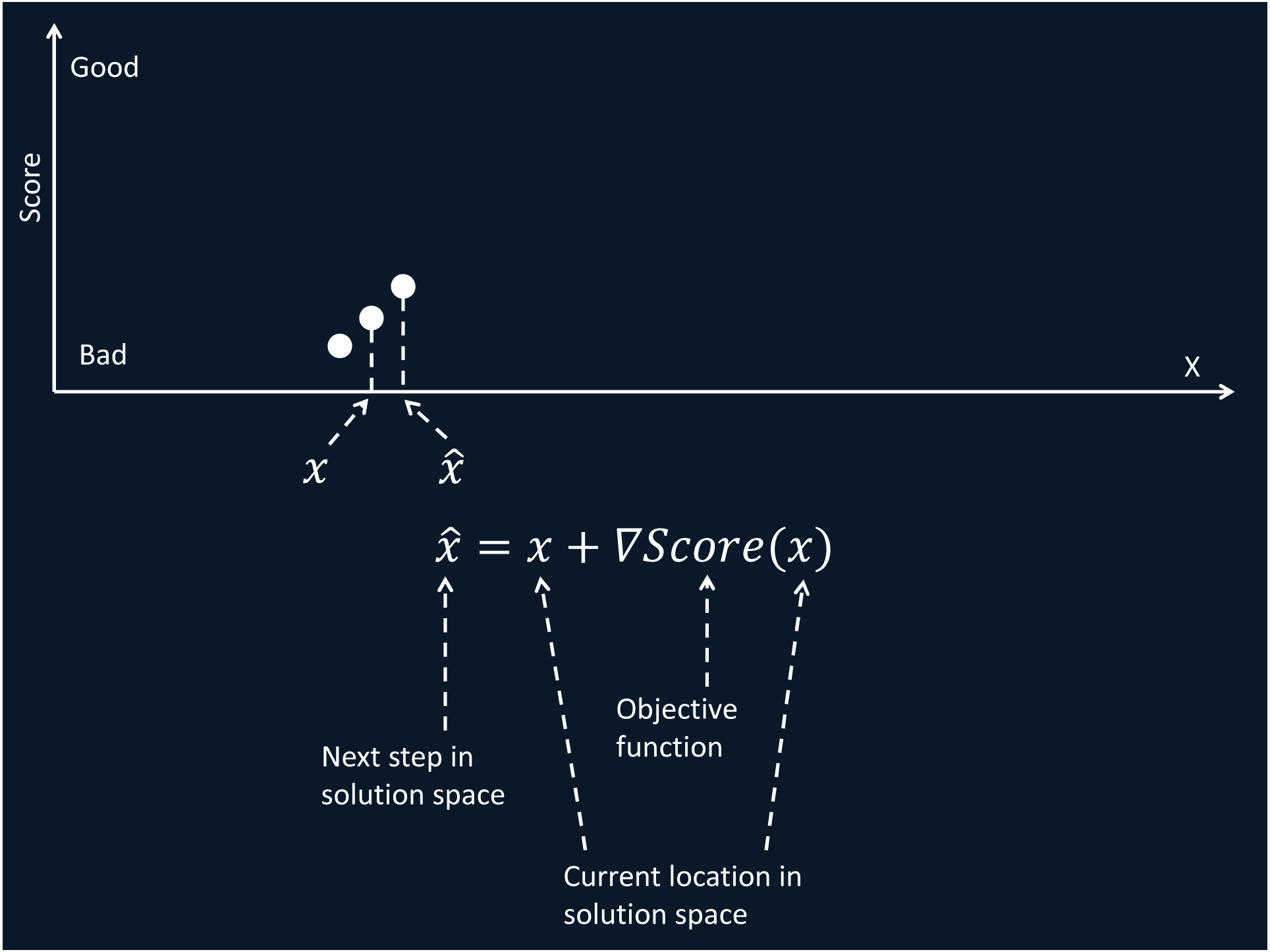
Gradient ascent optimization

3. Optimizer: Gradient ascent.

In our example, the objective function was the Score function:

$$\hat{x} = x + \nabla Score(x)$$




Good
Score
Bad
X
x
$\hat{x}$
$\hat{x} = x + \nabla Score(x)$
Next step in solution space
Objective function
Current location in solution space

A new way of thinking about algorithms

We can think of our simple registration algorithm in these terms as follows:

1. Objective function: Our “goodness” score.

2. Solution space: One number indicating the amount of translation along the X axis. Negative values go left, positive values go right.

3. Optimizer: Gradient ascent. 

A new way of thinking about algorithms

1. Objective function: Our “goodness” score.

Let’s write it down:

$$Score(x)$$

It takes a value along the x-axis and gives us back a score.

A new way of thinking about algorithms

1. Objective function: Our “goodness” score.

We want to find the x value that maximizes this score:

$$\dot{x} = \underset{x}{\arg max}\, Score(x)$$

This means: for all possible values of the x argument, find the one that gives the largest value of the (Score) function.

A new way of thinking about algorithms

We can think of our simple registration algorithm in these terms as follows:

1. Objective function: Our “goodness” score. 
2. Solution space: One number indicating the amount of translation along the X axis. Negative values go left, positive values go right.
3. Optimizer: Gradient ascent. 

A new way of thinking about algorithms

2. Solution space: One number indicating the amount of translation along the X axis. Negative values go left, positive values go right.

We need to indicate that x is an element of the real numbers:

$$\dot{x} = \underset{x \in \mathbb{R}}{\arg max}\, Score(x)$$

Registration-by-X-translation: the new way

Now, we can write down our entire algorithm as follows.

Objective function:

$$\dot{x} = \underset{x \in \mathbb{R}}{\arg max}\, Score(x)$$

Optimizer is gradient ascent:

$$\hat{x} = x + \nabla Score(x)$$

Registration-by-X-translation: the old way

Repeat the following:

S = The score of the current alignment.

Temporarily slide the moving image a bit to the left.
L = The score of the resulting alignment.

Temporarily slide the moving image a bit to the right.
R = The score of the resulting alignment.

If L > S and R < S, then slide the image to the left.
If R > S and L < S, then slide the image to the right.
If L < S and R < S, then we have found the best spot; STOP.

Now imagine that the images are vertically misaligned as well. How to write this down using the new approach?

Registration-by-X-and-Y-translation: the new way

Easy:

Objective function:

$$\dot{\boldsymbol{t}} = \underset{\boldsymbol{t} \in \mathbb{R}^2}{\arg max}\, Score(\boldsymbol{t})$$

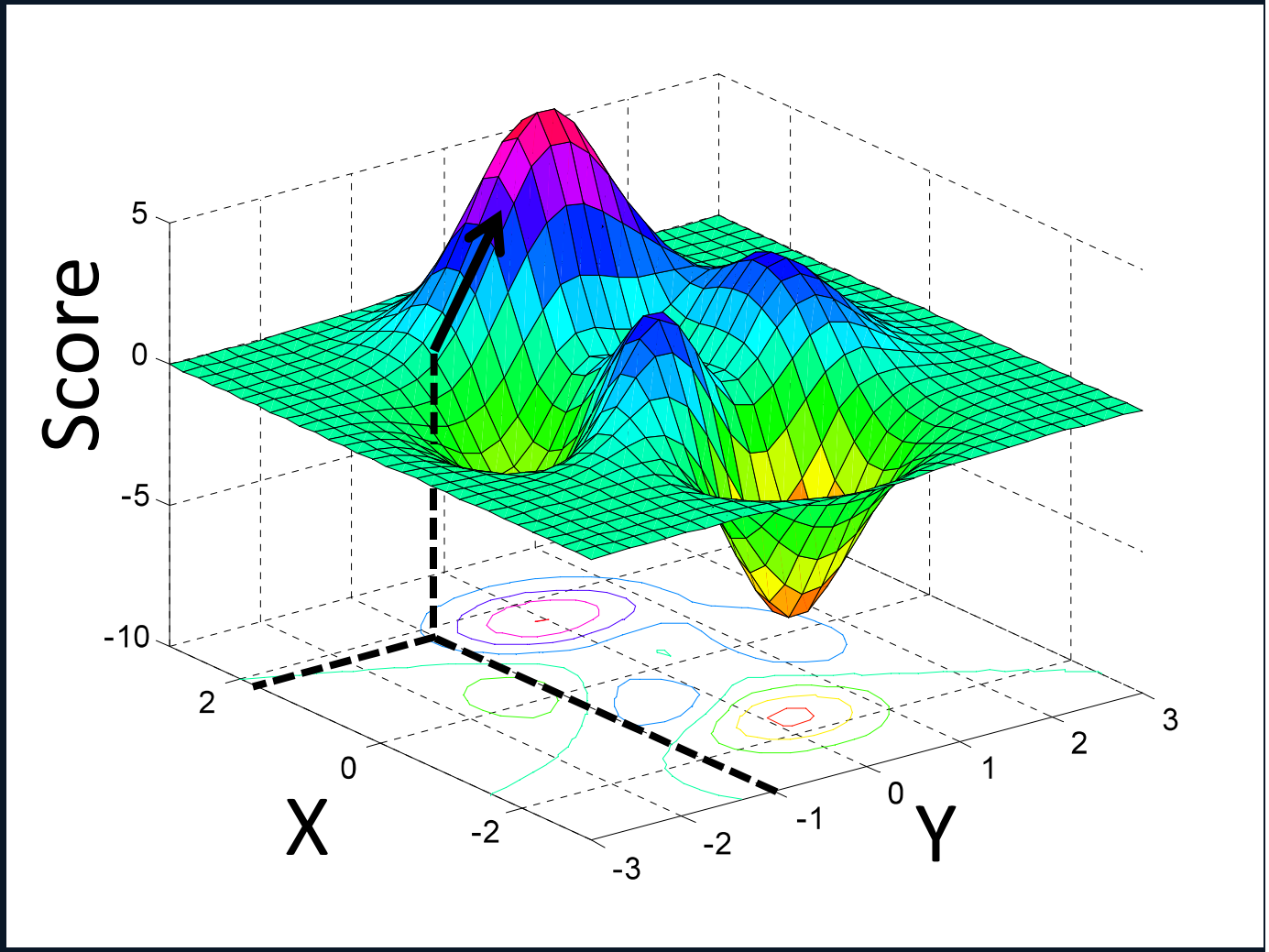
Optimizer is gradient ascent:

$$\hat{\boldsymbol{t}} = \boldsymbol{t} + \nabla Score(\boldsymbol{t})$$

The gradient ascent optimizer now takes “next best guess” steps in a 2-dimensional space.

Registration-by-X-and-Y-translation: the new way

Starting from any given (x,y) point, the gradient ascent optimizer will travel in the steepest upward direction along x and y simultaneously.



What happens when we add a rotational misalignment as well?

Registration by translation and rotation: the new way

Easy:

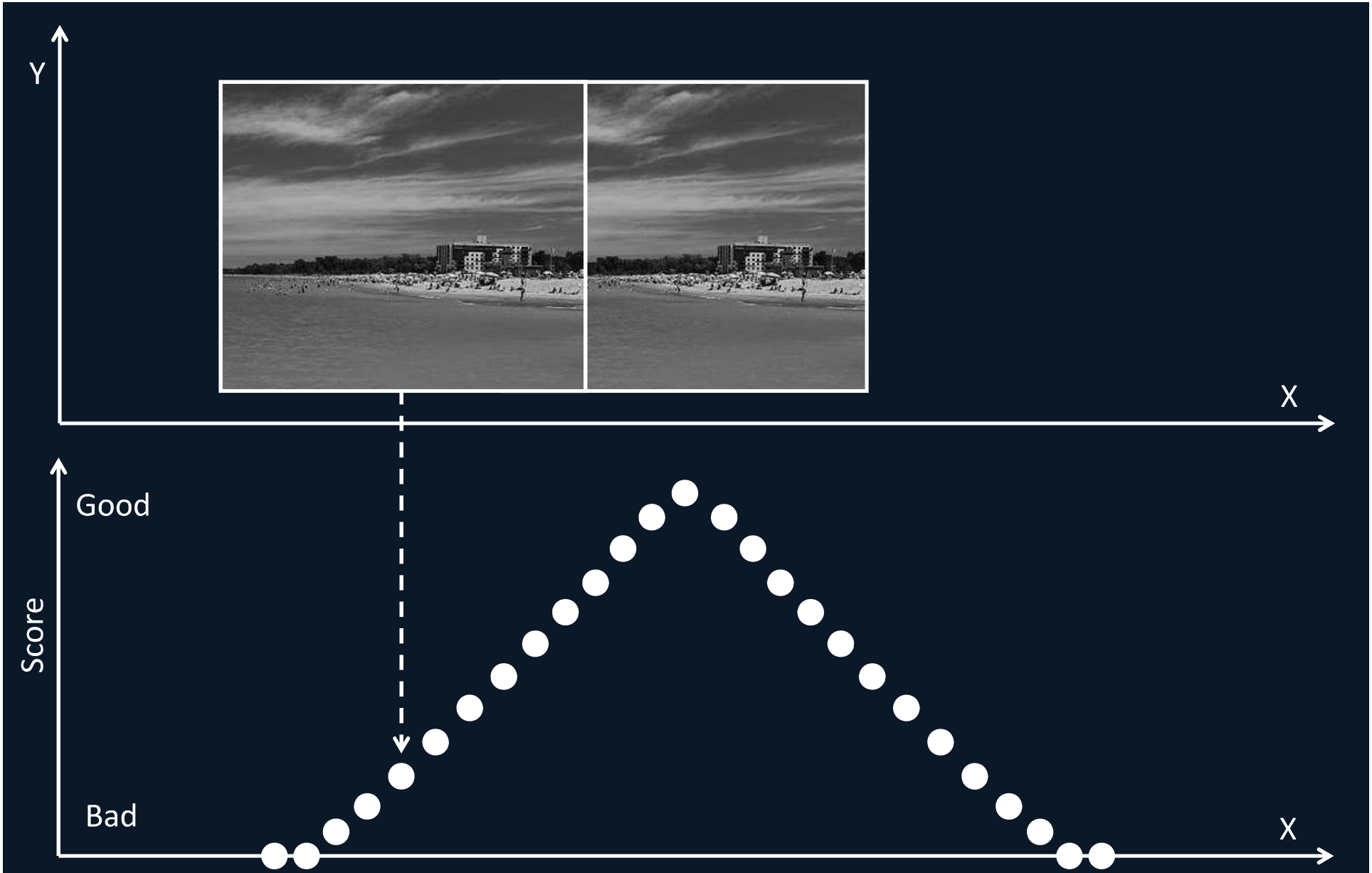
Objective function:

$$\dot{\boldsymbol{t}} = \underset{\boldsymbol{t} \in \mathbb{R}^2 \times \mathbb{S}^1}{\arg max} Score(\boldsymbol{t})$$

Optimizer is gradient ascent:

$$\hat{\boldsymbol{t}} = \boldsymbol{t} + \nabla Score(\boldsymbol{t})$$

The gradient ascent optimizer now takes “next best guess” steps in a 3-dimensional space: two of the dimensions are x and y, and the third is the rotation angle.



For every possible x-translation value, we need to compute a score indicating how good the image match is.

A way to picture image alignment

To understand how to calculate an image similarity metric, it is valuable to have a convenient mental picture of the agreement of pixel/voxel intensities when images match.

We will explore this mental picture first with a simplified example, and then with some real images.

In the simplified example, our 2D images are 3 x 3 pixels in size, and each pixel can have an intensity value ranging from 1 to 5.

Image A

1	2	2
5	3	1
1	4	4

Image B

3	2	2
5	3	1
1	4	4

Image B intensities

Image A intensities

	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

This is an accumulator table.

Initially, we put 0 everywhere in the table.

Image A

1	2	2
5	3	1
1	4	4

Image B

3	2	2
5	3	1
1	4	4

Image B intensities

Image A intensities

	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

We look at the first two corresponding pixels in the two images.

They correspond in the spatial sense – both are in the top-left at (1,1).

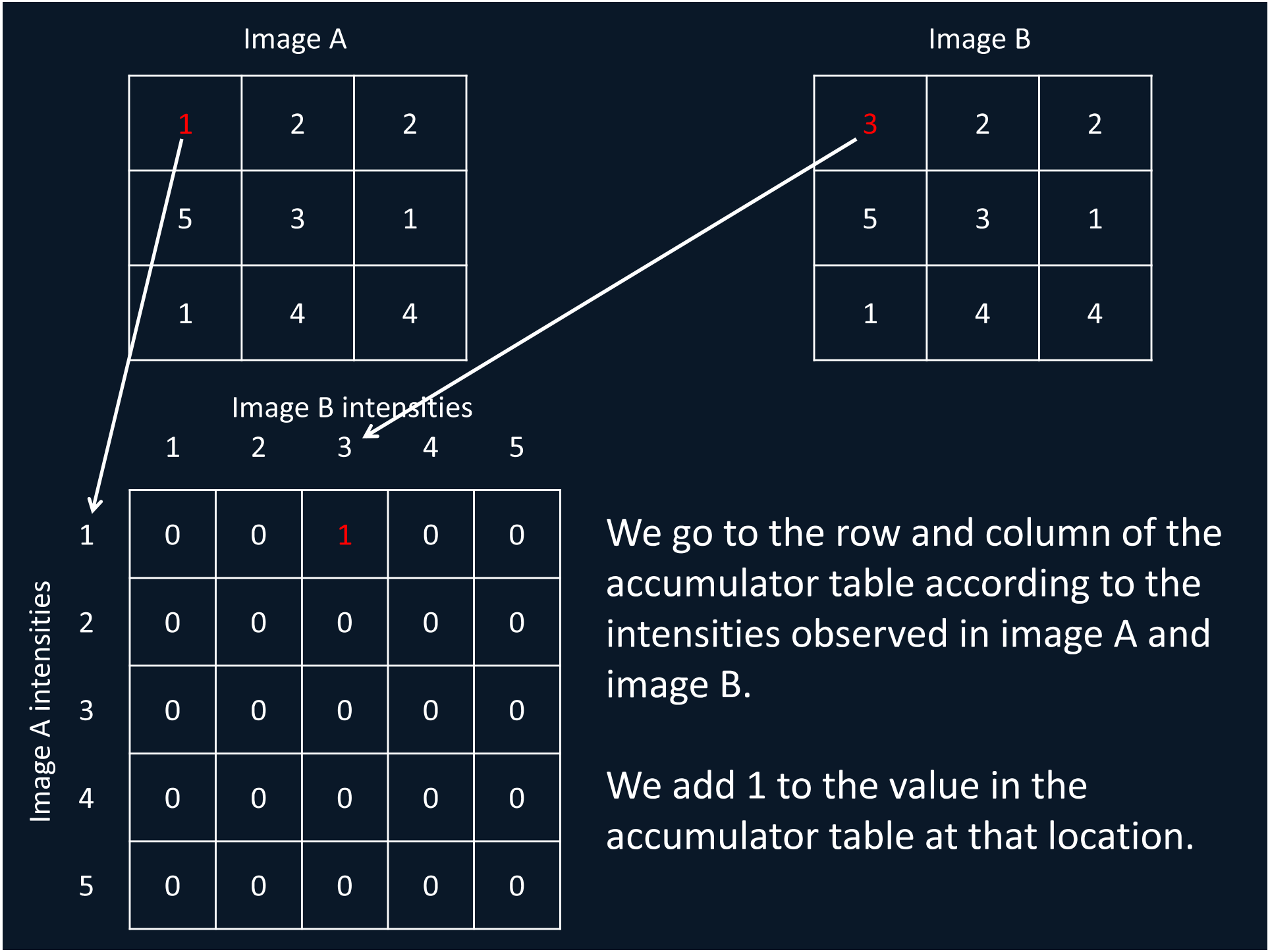


Image A
1 2 2
5 3 1
1 4 4
Image B
3 2 2
5 3 1
1 4 4
Image B intensities
1 2 3 4 5
Image A intensities
1 0 0 1 0 0
2 0 0 0 0 0
3 0 0 0 0 0
4 0 0 0 0 0
5 0 0 0 0 0
We go to the row and column of the accumulator table according to the intensities observed in image A and image B.
We add 1 to the value in the accumulator table at that location.

Image A

1	2	2
5	3	1
1	4	4

Image B

3	2	2
5	3	1
1	4	4

Image B intensities

Image A intensities

	1	2	3	4	5
1	0	0	1	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

We go to the next pair of spatially corresponding pixels…

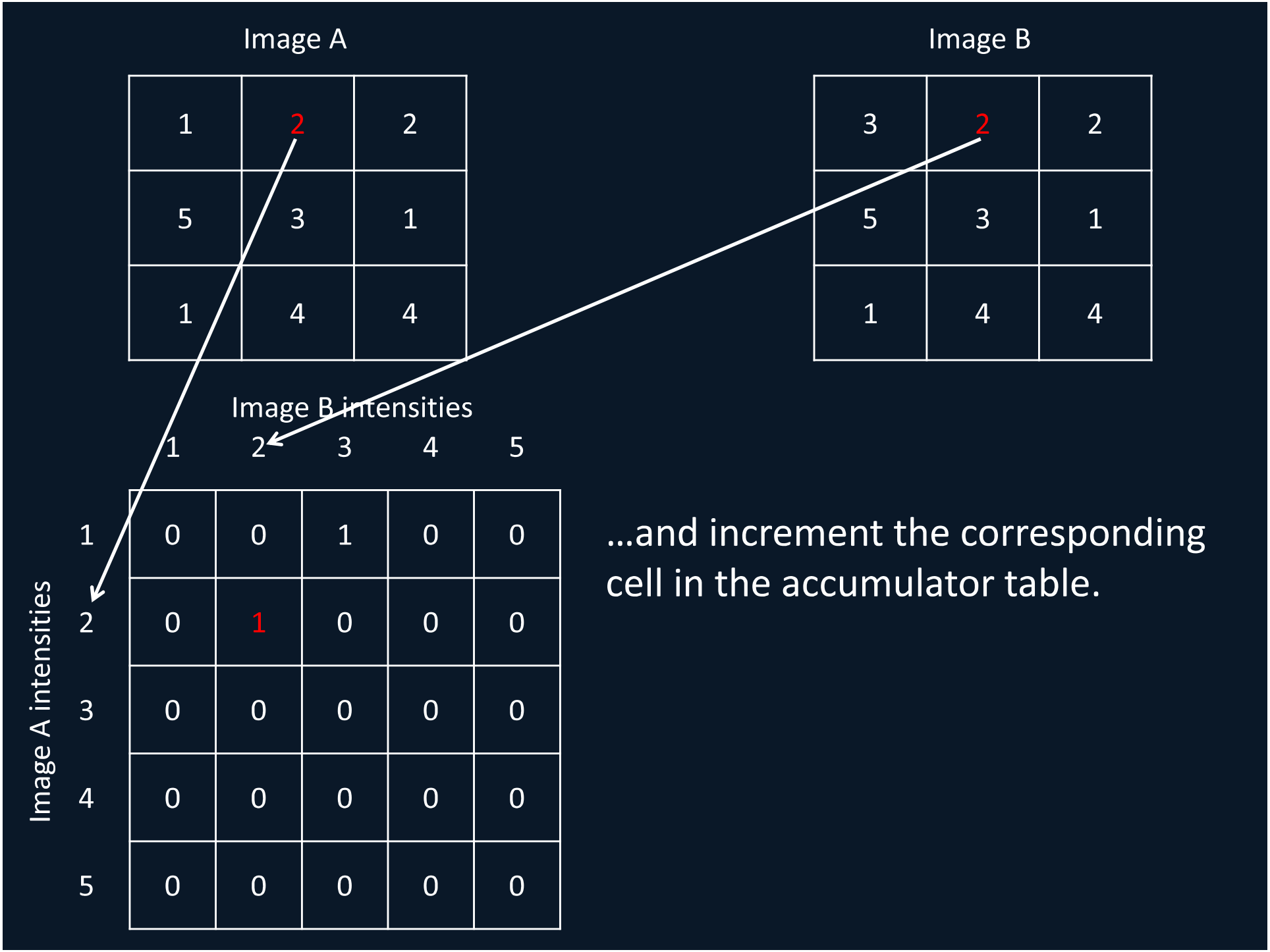


Image A
1 2 2
5 3 1
1 4 4
Image B
3 2 2
5 3 1
1 4 4
Image B intensities
1 2 3 4 5
Image A intensities
1 0 0 1 0 0
2 0 1 0 0 0
3 0 0 0 0 0
4 0 0 0 0 0
5 0 0 0 0 0
…and increment the corresponding cell in the accumulator table.

Image A

1	2	2
5	3	1
1	4	4

Image B

3	2	2
5	3	1
1	4	4

Image B intensities

Image A intensities

	1	2	3	4	5
1	0	0	1	0	0
2	0	2	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

We continue in the same fashion for all spatially corresponding pixels.

Image A

1	2	2
5	3	1
1	4	4

Image B

3	2	2
5	3	1
1	4	4

Image B intensities

Image A intensities

	1	2	3	4	5
1	0	0	1	0	0
2	0	2	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	1

We continue in the same fashion for all spatially corresponding pixels.

Image A

1	2	2
5	3	1
1	4	4

Image B

3	2	2
5	3	1
1	4	4

Image B intensities

Image A intensities

	1	2	3	4	5
1	0	0	1	0	0
2	0	2	0	0	0
3	0	0	1	0	0
4	0	0	0	0	0
5	0	0	0	0	1

We continue in the same fashion for all spatially corresponding pixels.

Image A

1	2	2
5	3	1
1	4	4

Image B

3	2	2
5	3	1
1	4	4

Image B intensities

Image A intensities

	1	2	3	4	5
1	1	0	1	0	0
2	0	2	0	0	0
3	0	0	1	0	0
4	0	0	0	0	0
5	0	0	0	0	1

We continue in the same fashion for all spatially corresponding pixels.

Image A

1	2	2
5	3	1
1	4	4

Image B

3	2	2
5	3	1
1	4	4

Image B intensities

Image A intensities

	1	2	3	4	5
1	2	0	1	0	0
2	0	2	0	0	0
3	0	0	1	0	0
4	0	0	0	0	0
5	0	0	0	0	1

We continue in the same fashion for all spatially corresponding pixels.

Image A

1	2	2
5	3	1
1	4	4

Image B

3	2	2
5	3	1
1	4	4

Image B intensities

Image A intensities

	1	2	3	4	5
1	2	0	1	0	0
2	0	2	0	0	0
3	0	0	1	0	0
4	0	0	0	1	0
5	0	0	0	0	1

We continue in the same fashion for all spatially corresponding pixels.

Image A

1	2	2
5	3	1
1	4	4

Image B

3	2	2
5	3	1
1	4	4

Image B intensities

Image A intensities

	1	2	3	4	5
1	2	0	1	0	0
2	0	2	0	0	0
3	0	0	1	0	0
4	0	0	0	2	0
5	0	0	0	0	1

We continue in the same fashion for all spatially corresponding pixels.

Image A

1	2	2
5	3	1
1	4	4

Image B

3	2	2
5	3	1
1	4	4

Image B intensities

Image A intensities

	1	2	3	4	5
1	2	0	1	0	0
2	0	2	0	0	0
3	0	0	1	0	0
4	0	0	0	2	0
5	0	0	0	0	1

How to interpret the contents of the accumulator table?

Image A

1	2	2
5	3	1
1	4	4

Image B

3	2	2
5	3	1
1	4	4

Image B intensities

Image A intensities	1	2	3	4	5
1	2	0	1	0	0
2	0	2	0	0	0
3	0	0	1	0	0
4	0	0	0	2	0
5	0	0	0	0	1

“There were two spatial locations where Image A had an intensity of 2 and Image B also had an intensity of 2.”

Image A

1	2	2
5	3	1
1	4	4

Image B

3	2	2
5	3	1
1	4	4

Image B intensities

Image A intensities

	1	2	3	4	5
1	2	0	1	0	0
2	0	2	0	0	0
3	0	0	1	0	0
4	0	0	0	2	0
5	0	0	0	0	1

“There was one spatial location where Image A had an intensity of 1 and Image B had an intensity of 3.”

Image A

1	2	2
5	3	1
1	4	4

Image B

3	2	2
5	3	1
1	4	4

Image B intensities

Image A intensities	1	2	3	4	5
1	2	0	1	0	0
2	0	2	0	0	0
3	0	0	1	0	0
4	0	0	0	2	0
5	0	0	0	0	1

Image A and Image B are almost a perfect match (except for one pixel).

Let's fix that.

Image A

1	2	2
5	3	1
1	4	4

Image B

1	2	2
5	3	1
1	4	4

Image B intensities

Image A intensities

	1	2	3	4	5
1	3	0	0	0	0
2	0	2	0	0	0
3	0	0	1	0	0
4	0	0	0	2	0
5	0	0	0	0	1

Image A and Image B are almost a perfect match (except for one pixel).

Let's fix that.

Image A

1	2	2
5	3	1
1	4	4

Image B

1	2	2
5	3	1
1	4	4

Image B intensities

Image A intensities

	1	2	3	4	5
1	3	0	0	0	0
2	0	2	0	0	0
3	0	0	1	0	0
4	0	0	0	2	0
5	0	0	0	0	1

Notice that when Image A and Image B are a perfect match, the only nonzero elements in the accumulator table are on the diagonal.

Image A

1	2	2
5	3	1
1	4	4

Image B

1	2	2
5	3	1
1	4	4

Image B intensities

Image A intensities

	1	2	3	4	5
1	3	0	0	0	0
2	0	2	0	0	0
3	0	0	1	0	0
4	0	0	0	2	0
5	0	0	0	0	1

Notice that when Image A and Image B are a perfect match, the only nonzero elements in the accumulator table are on the diagonal.

Image A

1	2	2
5	3	1
1	4	4

Image B

3	4	1
1	5	4
2	3	5

Image B intensities

Image A intensities

	1	2	3	4	5
1	0	1	1	1	0
2	1	0	0	1	0
3	0	0	0	0	1
4	0	0	1	0	1
5	1	0	0	0	0

When we modify image B to be totally different from image A, the accumulator table has 0s in the diagonal and the nonzeros have "spread out".

This "smearing" of the accumulator table is an important concept.

Image A

1	2	2
5	3	1
1	4	4

Image B

1	2	2
5	3	1
1	4	4

Image B intensities

Image A intensities	1	2	3	4	5
1	3	0	0	0	0
2	0	2	0	0	0
3	0	0	1	0	0
4	0	0	0	2	0
5	0	0	0	0	1

I.e. for the perfect match we have a compact “ridge” of relatively large numbers along the diagonal..

Image A

1	2	2
5	3	1
1	4	4

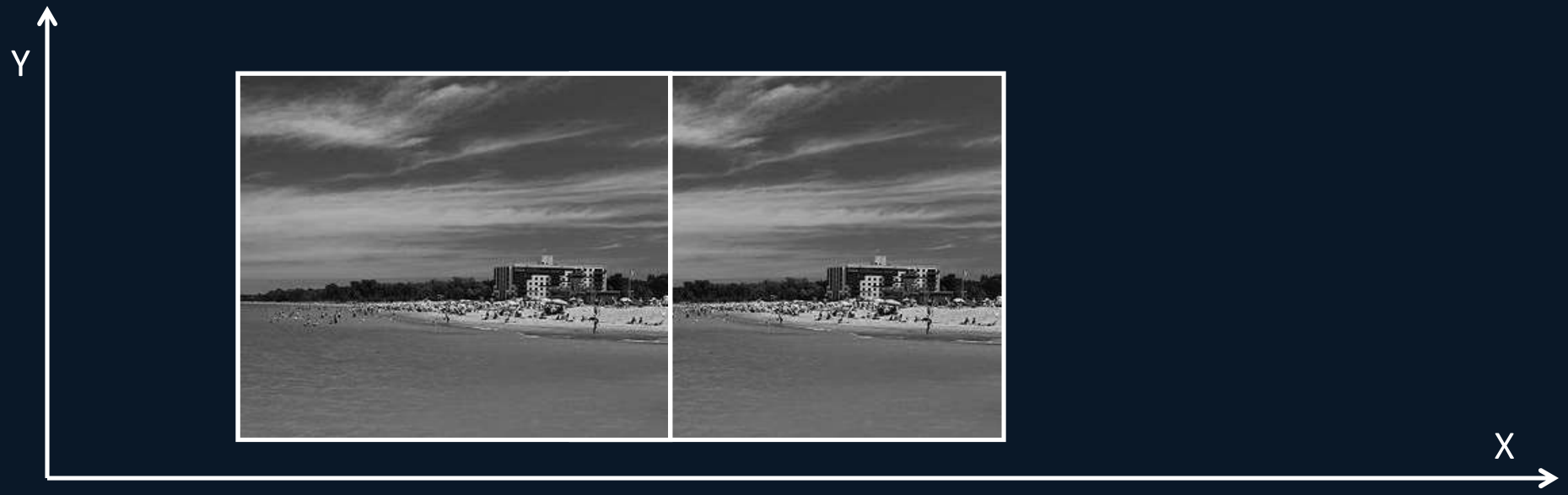
Image B

3	4	1
1	5	4
2	3	5

Image B intensities

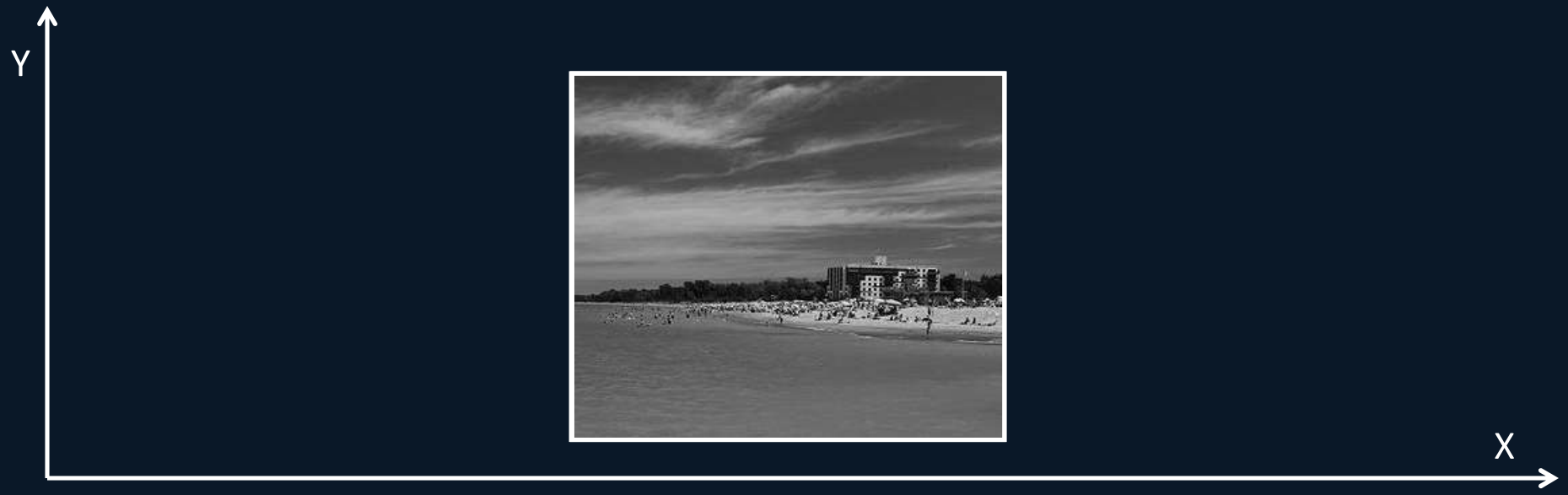
Image A intensities	1	2	3	4	5
1	0	1	1	1	0
2	1	0	0	1	0
3	0	0	0	0	1
4	0	0	1	0	1
5	1	0	0	0	0

..and for the bad match we have a diffuse “plateau” of relatively small numbers in the off-diagonal areas.

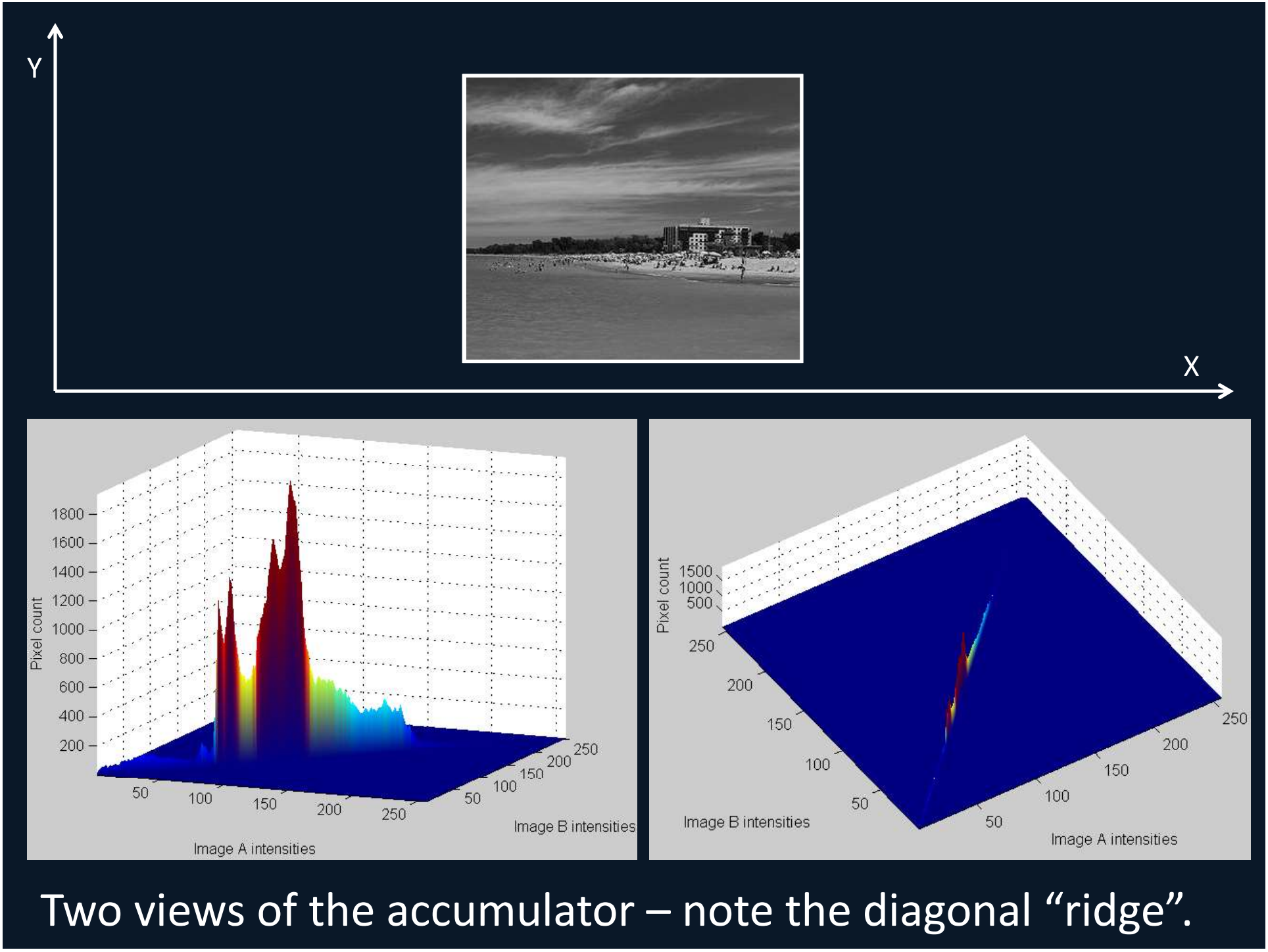


Let’s look at the accumulator table for real images.

These images have 256 different gray levels so to show this as a table we’d need a 256 x 256 table. This is easier to see as a 2D histogram, where large numbers are shown as large peaks, and small numbers as small peaks.

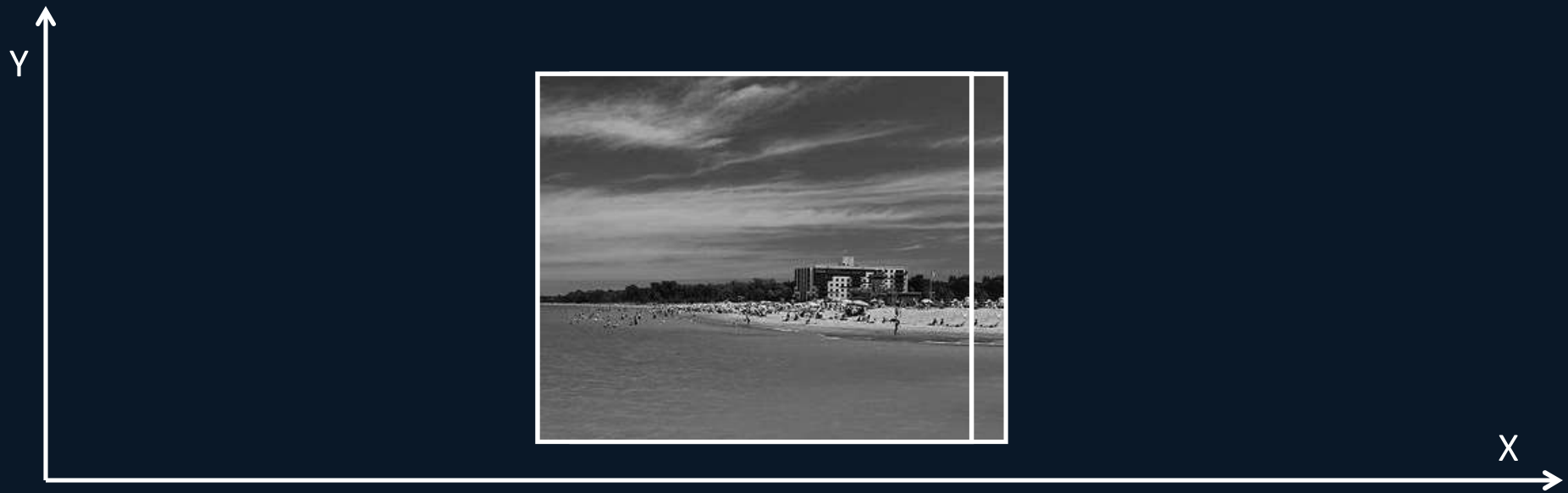


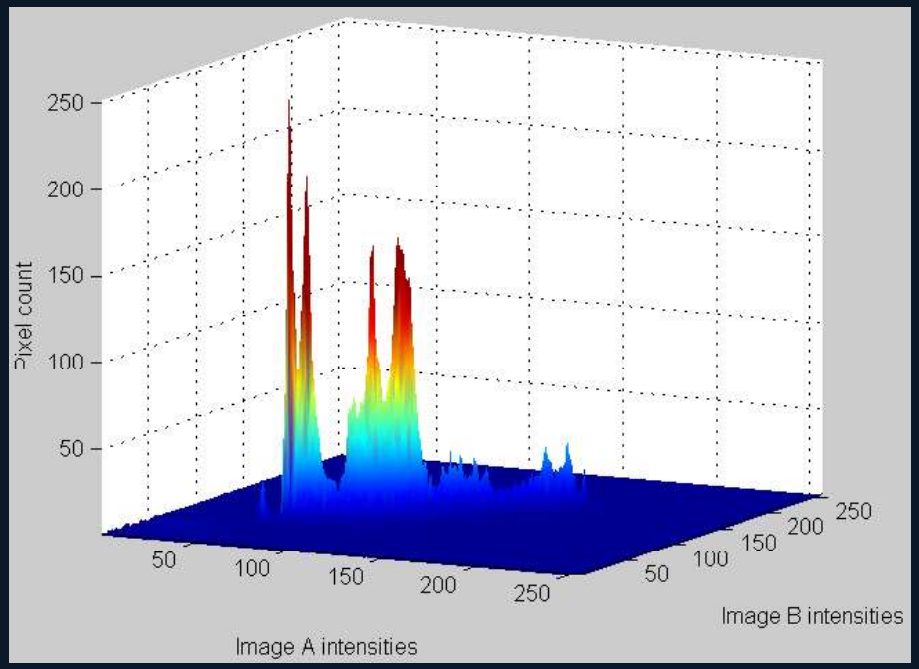
We begin with the situation where the images are perfectly aligned.

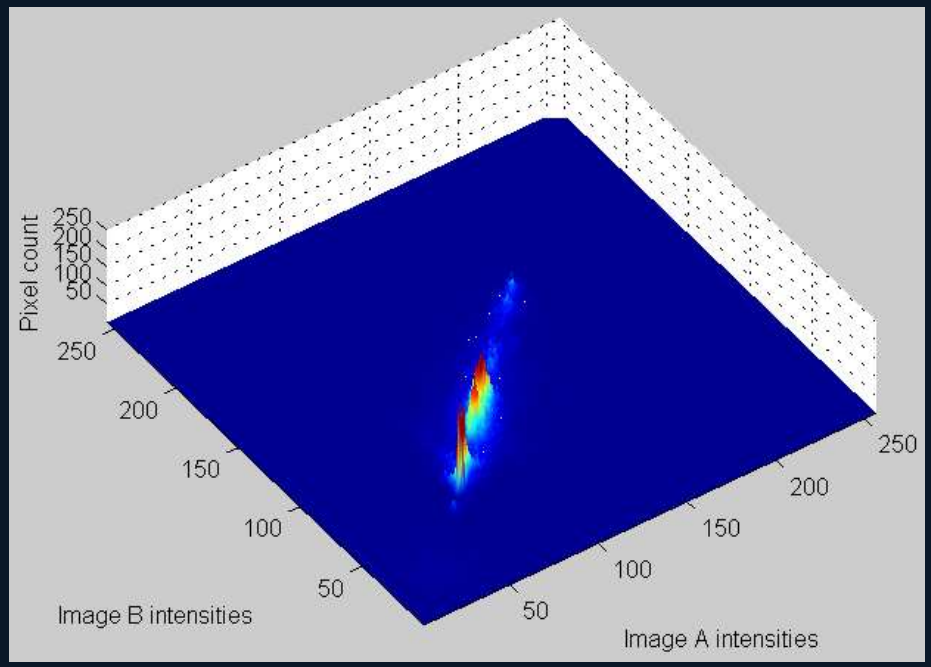


Two views of the accumulator – note the diagonal “ridge”.

Now let’s move the images apart a bit and look at the accumulator table only in the region where they overlap.

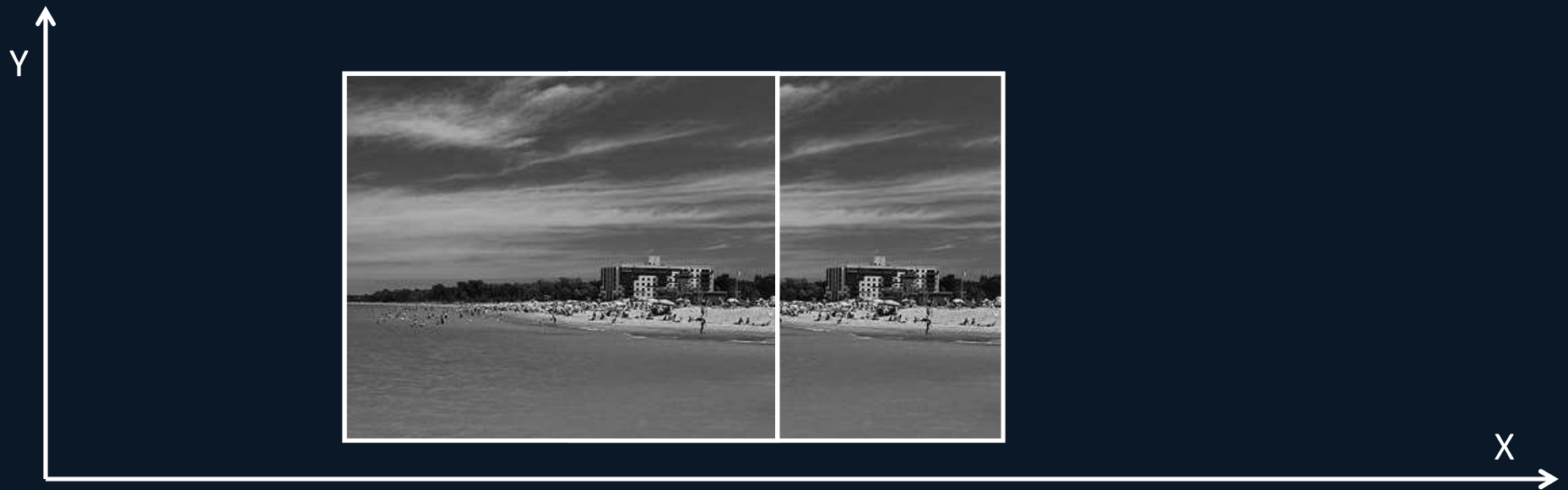


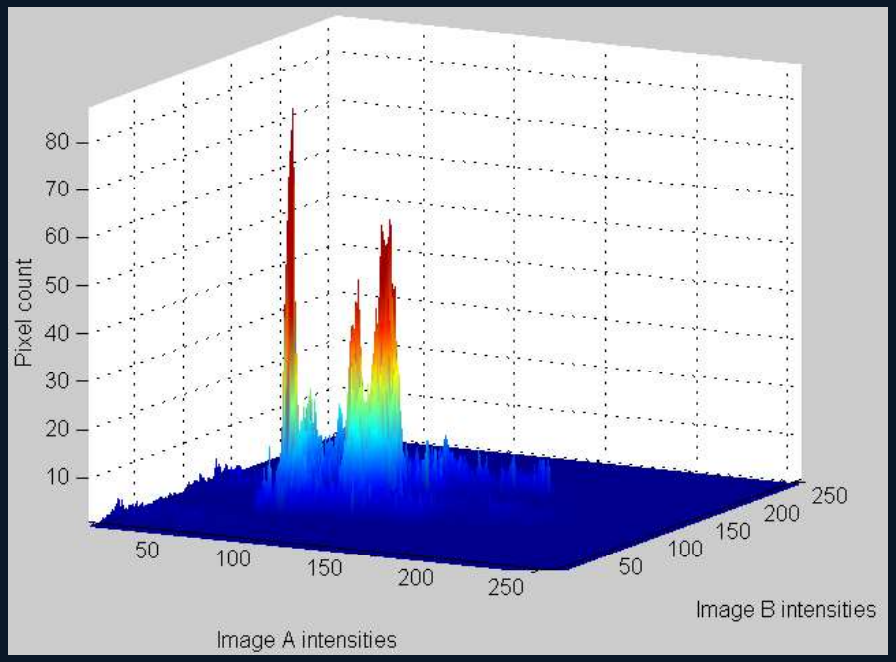


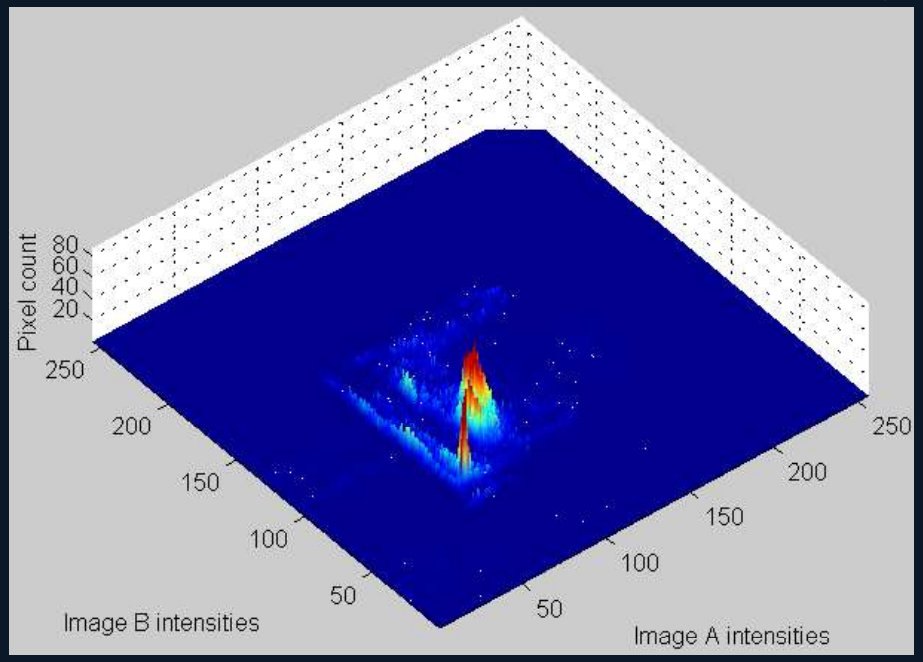


Notice that the ridge is starting to flatten/"smear".

Now let’s move the images apart so they only overlap by about 50% and look at the accumulator table only in the region where they overlap.

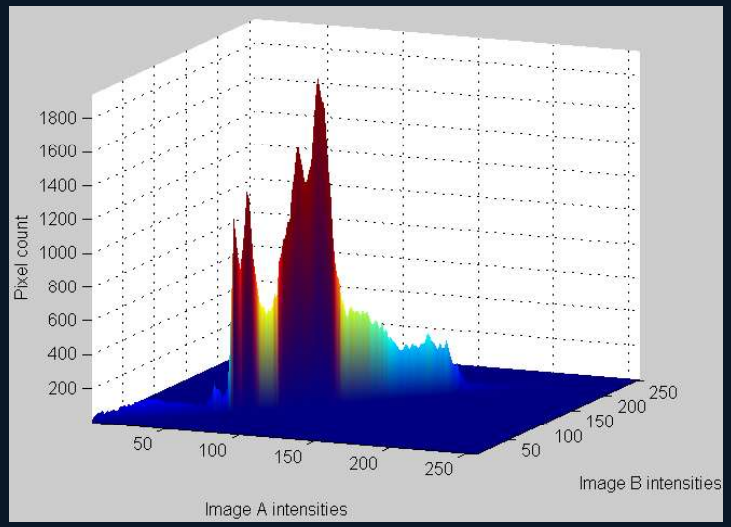


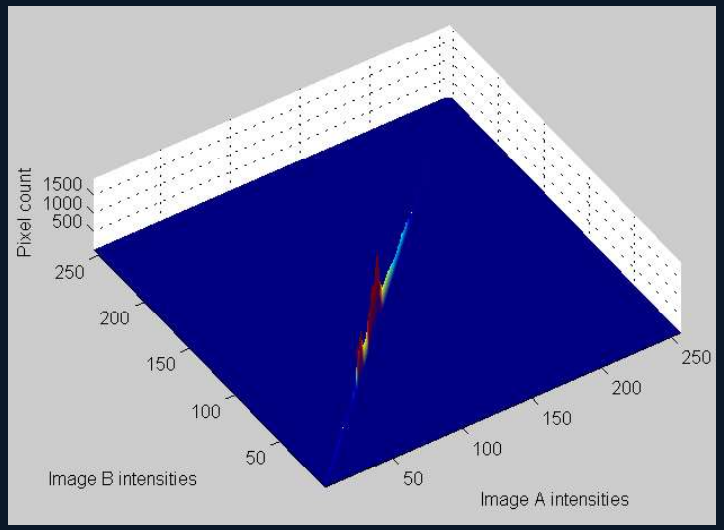




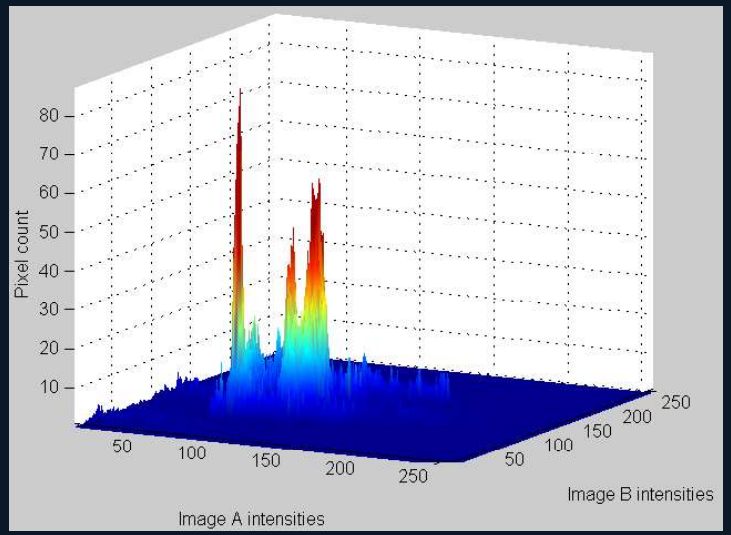
The accumulator table values have become much more distributed, and the heights of the peaks are lower.

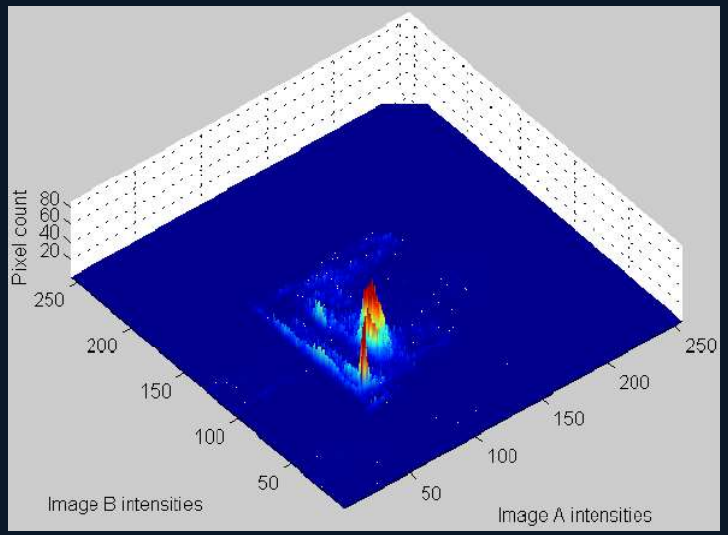
Good alignment





Bad alignment



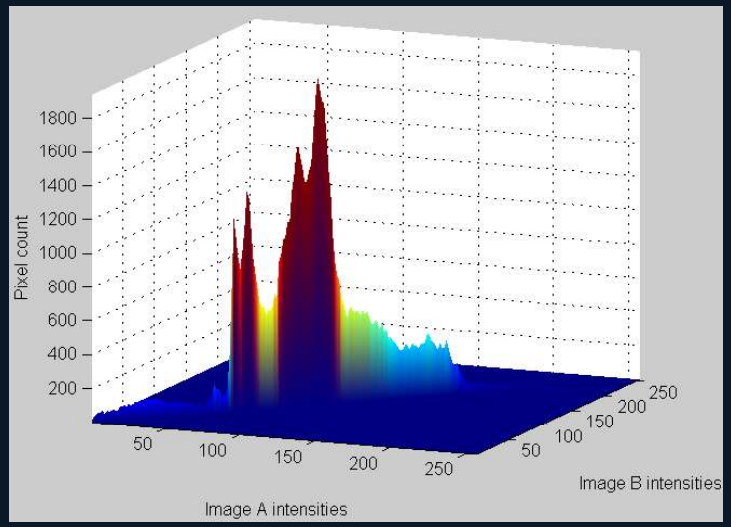


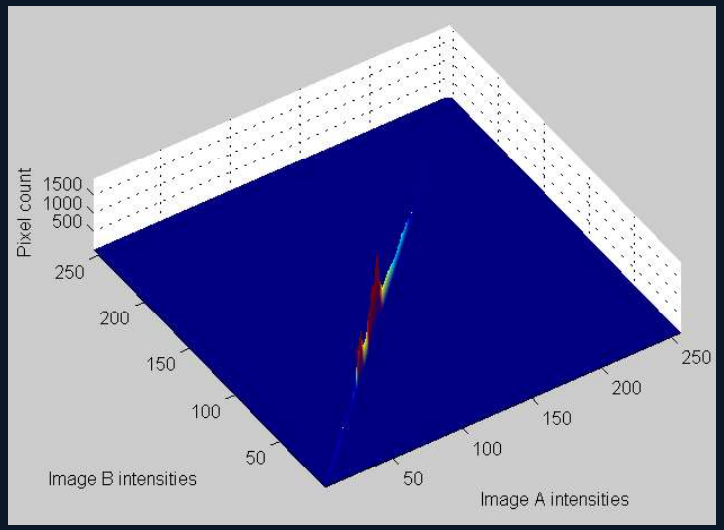
Key observation:

Good alignment = histogram with narrow, tall peaks.

Bad alignment = flatter histogram.

Good alignment





How to calculate a scalar “goodness” score based on this observation?

For these images, note that when they are in perfect alignment, the intensity values of corresponding pixels are always exactly the same.

This observation is due to the fact that these two images are identical – approximately analogous to e.g. two CT scans taken of the same patient on different days, where the patient is positioned differently.

In this type of situation, we can use an intensity-difference based approach to calculating the score.

Image similarity metric: Mean-squared error

$$MSE(I1, I2) = \frac{1}{N}\sum_{x}\sum_{y}\left(I1(x,y) - I2(x,y)\right)^2$$

N: # pixels/voxels
$I1(x,y)$: intensity of pixel in $I1$ at location (x,y)
$I2(x,y)$: intensity of pixel in $I2$ at location (x,y)

The ideal value of MSE is 0; thus gradient *descent* could be used to optimize this.

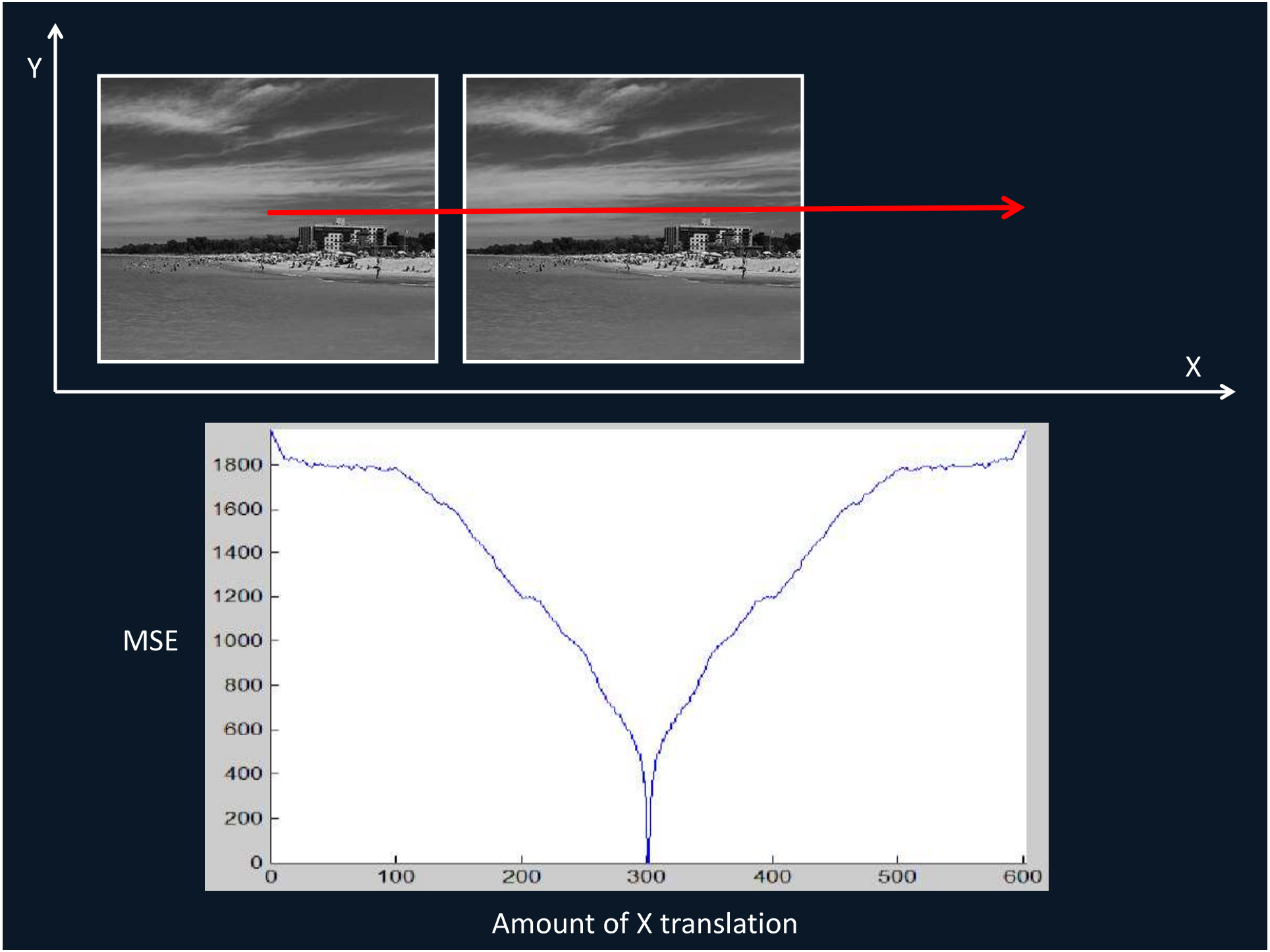
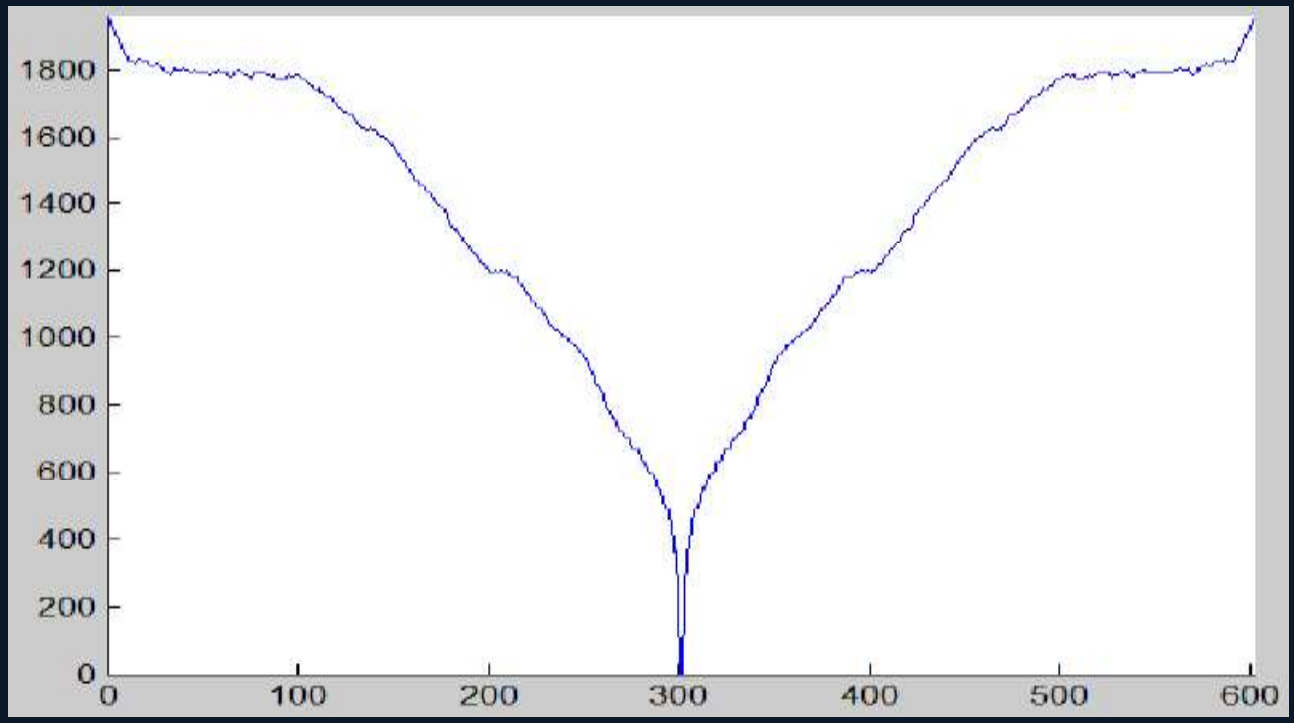

Y
X
MSE
1800
1600
1400
1200
1000
800
600
400
200
0
0
100
200
300
400
500
600
Amount of X translation

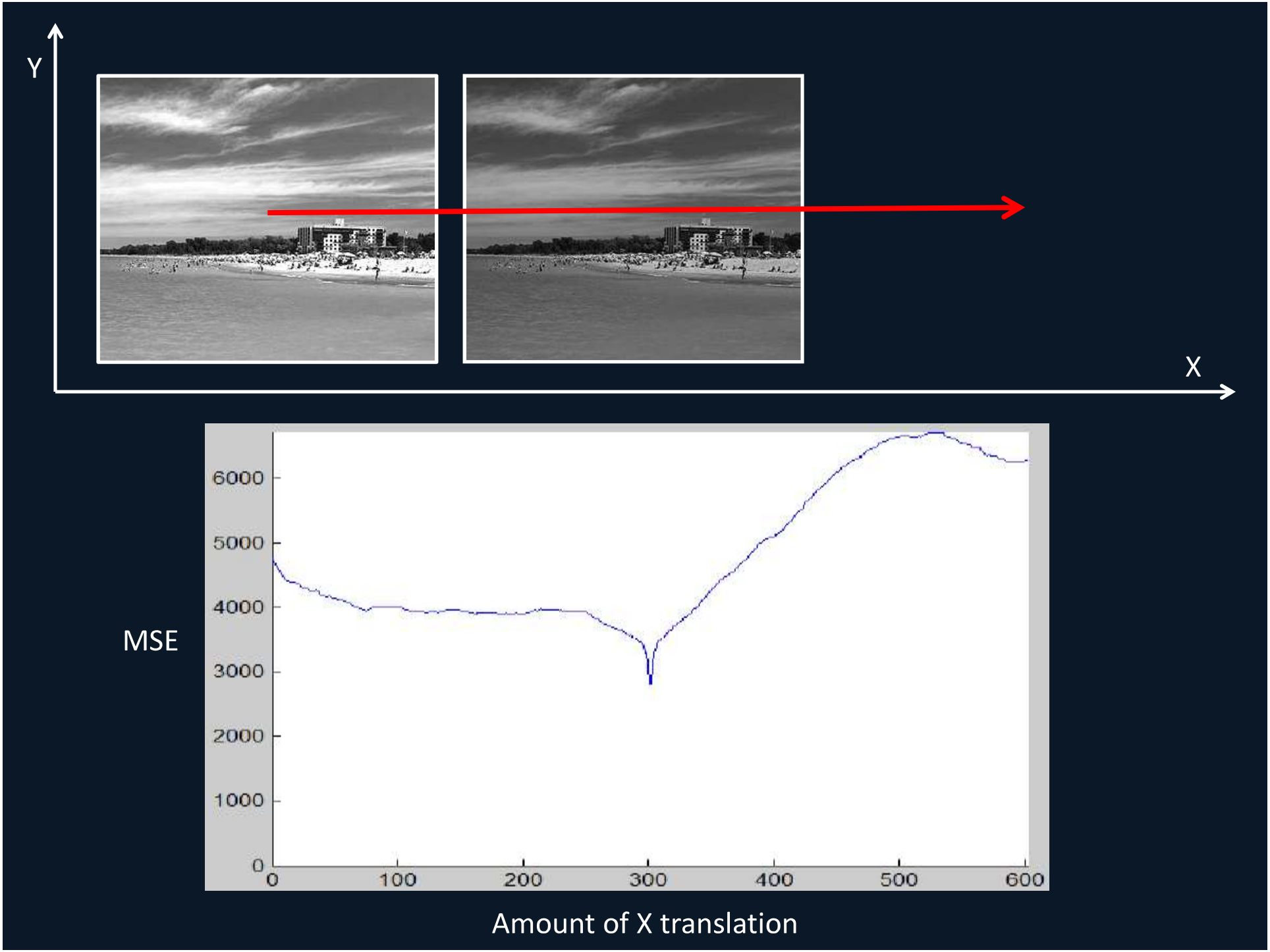
Image similarity metric: Mean-squared error

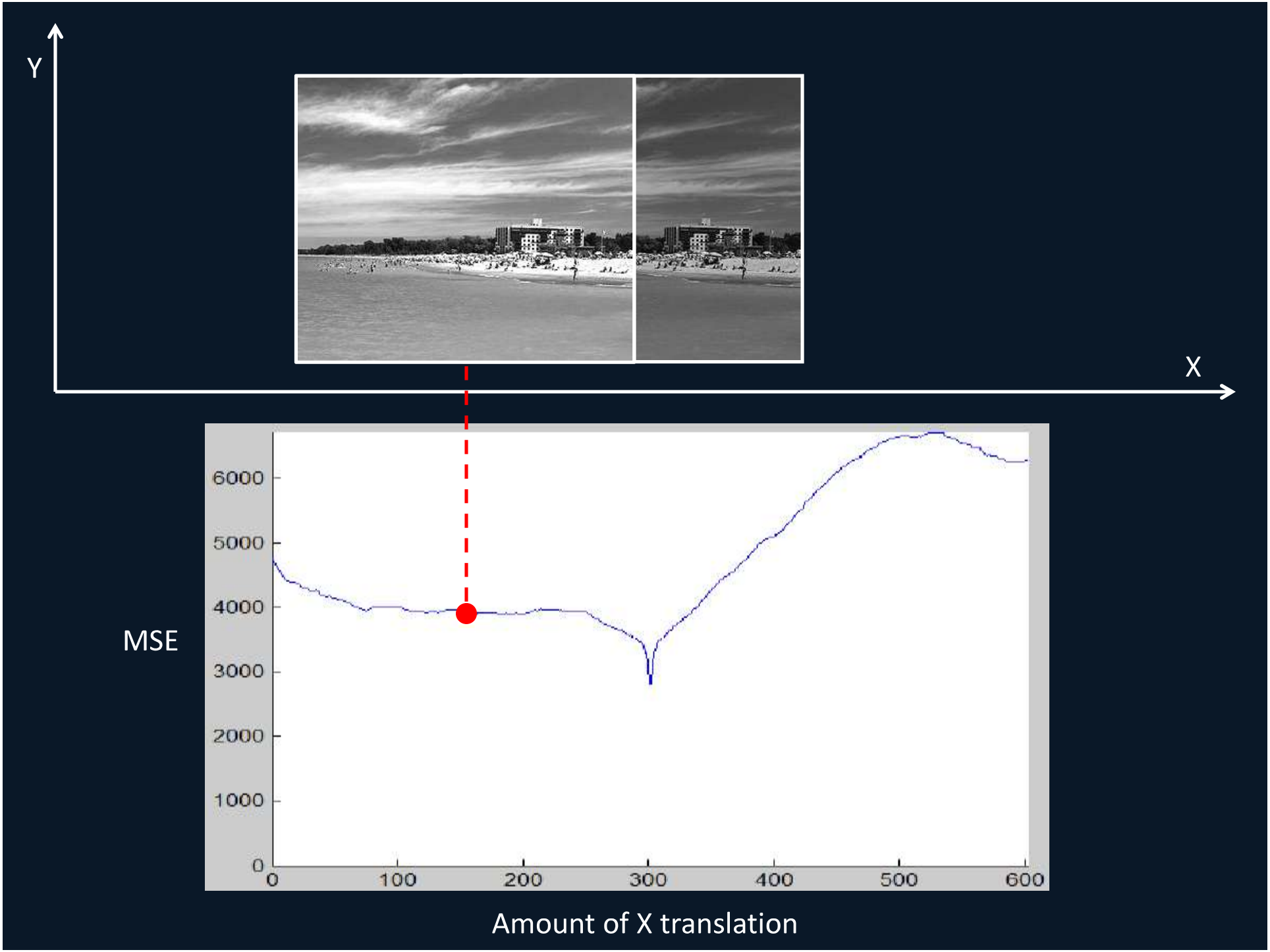
This is good behaviour for an image similarity metric. There are few local minima in which a gradient descent optimizer could get stuck, and there is a downward slope toward the correct answer even from far away.

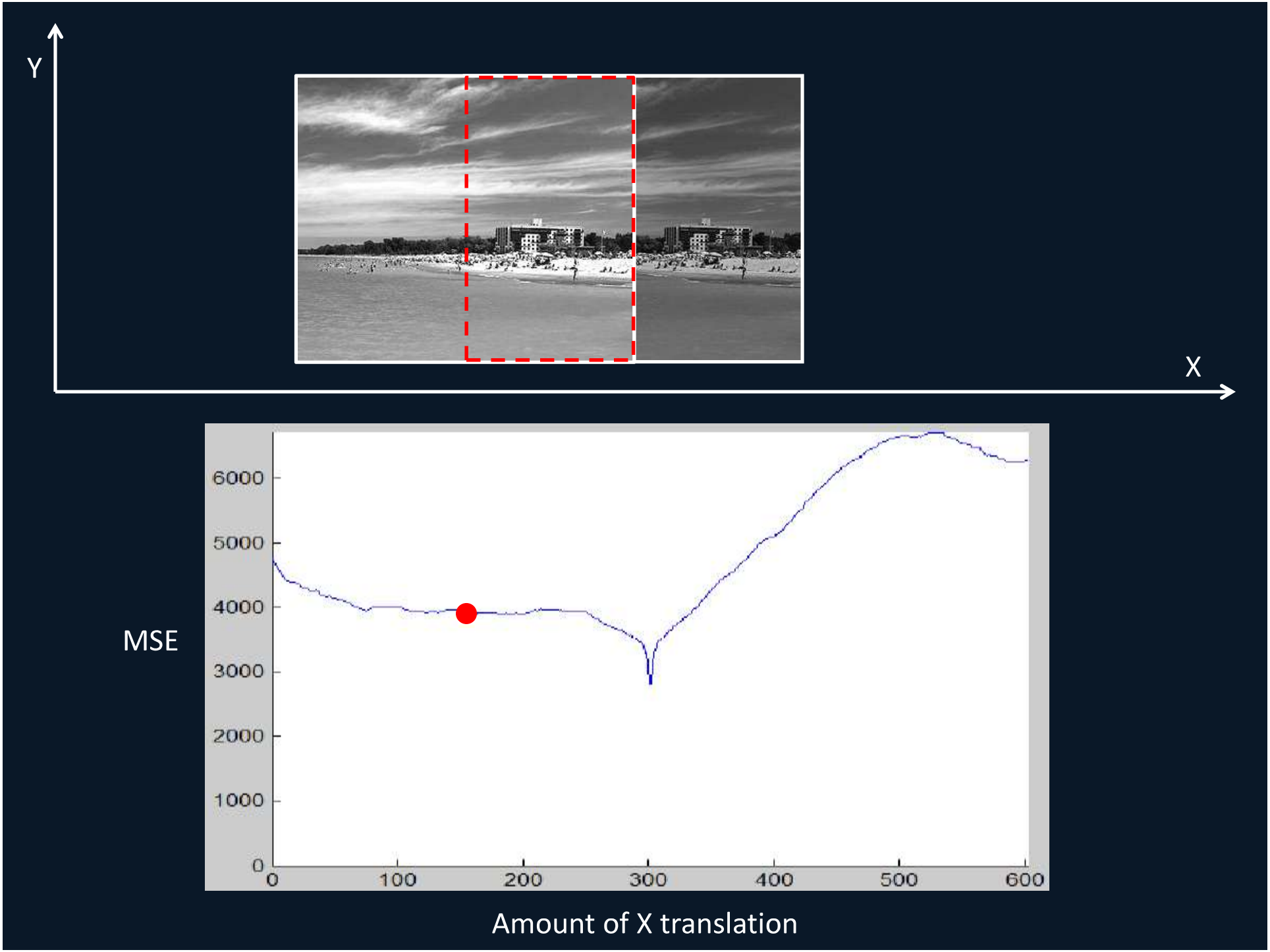


In this example, the intensity of the moving image has been scaled by 150%. This could be analogous to two patient MRIs taken on different days with different scanners.

Let's observe the behaviour of the MSE image similarity metric for this situation.


Y
X
MSE
6000
5000
4000
3000
2000
1000
0
0
100
200
300
400
500
600
Amount of X translation


Y
X
MSE
6000
5000
4000
3000
2000
1000
0
0
100
200
300
400
500
600
Amount of X translation

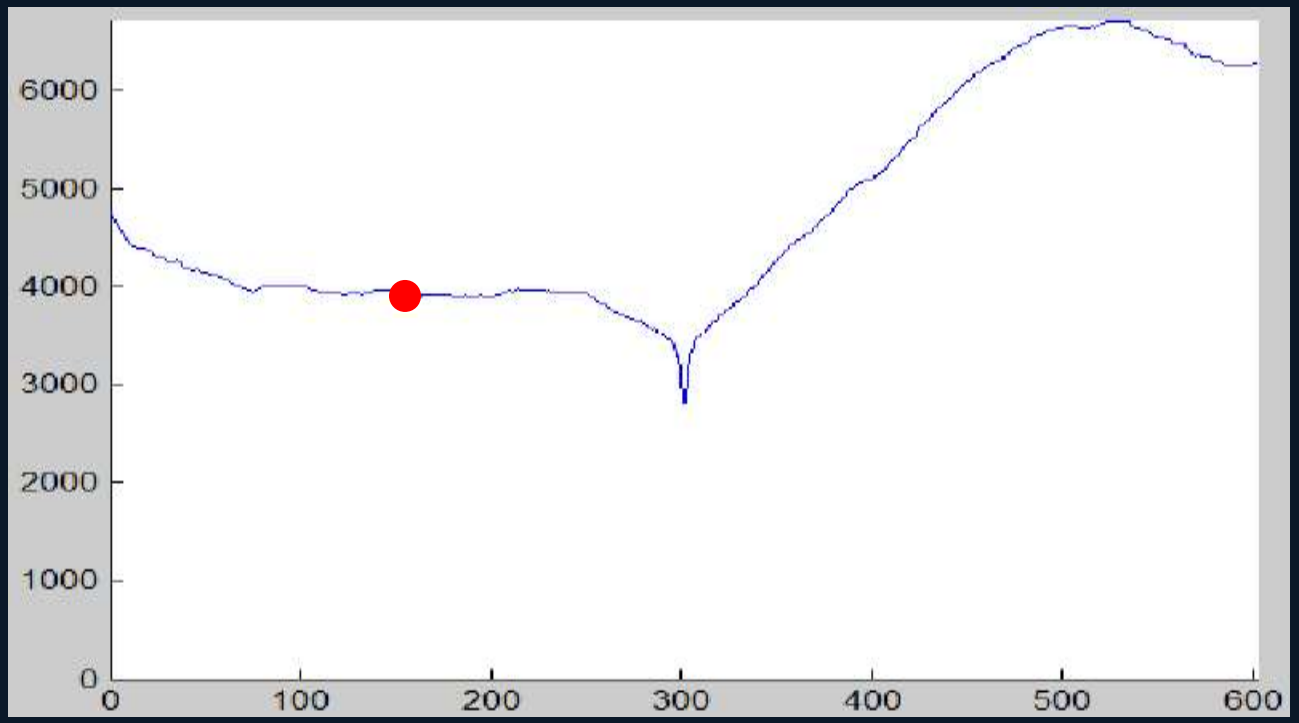

Y
X
MSE
6000
5000
4000
3000
2000
1000
0
0
100
200
300
400
500
600
Amount of X translation

Moving image

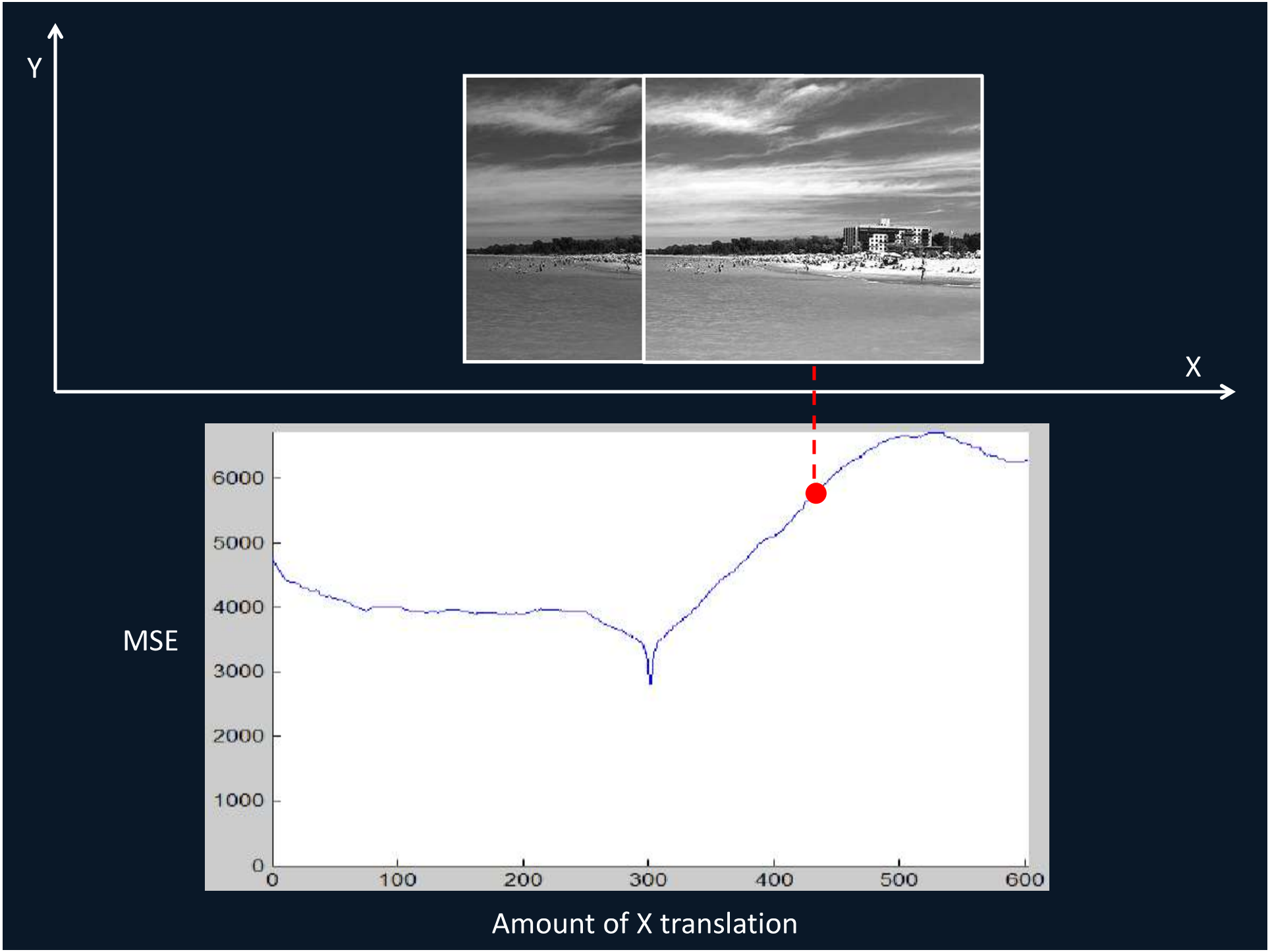
Fixed image

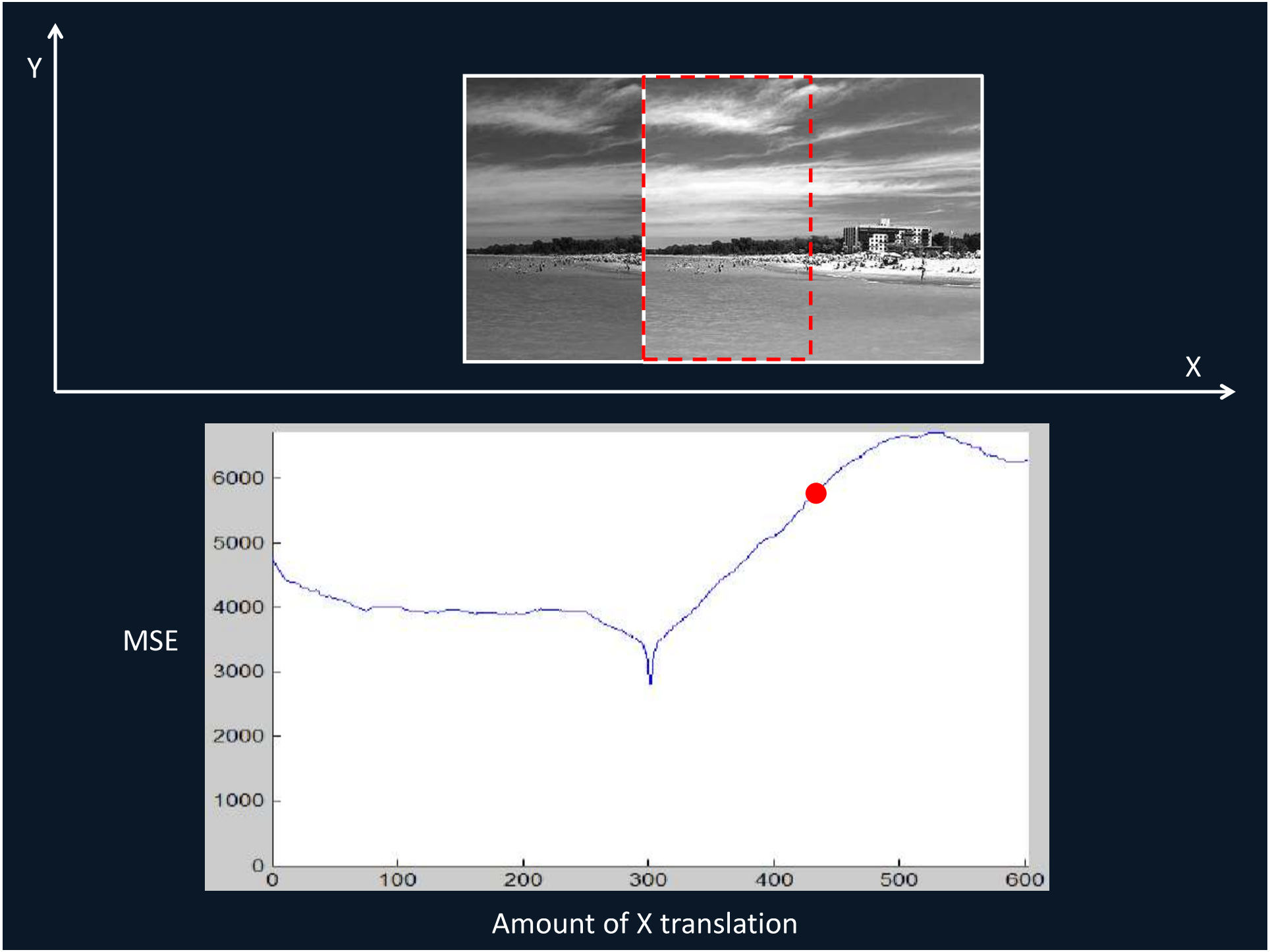
"Overlap regions"
to be compared

MSE



Amount of X translation

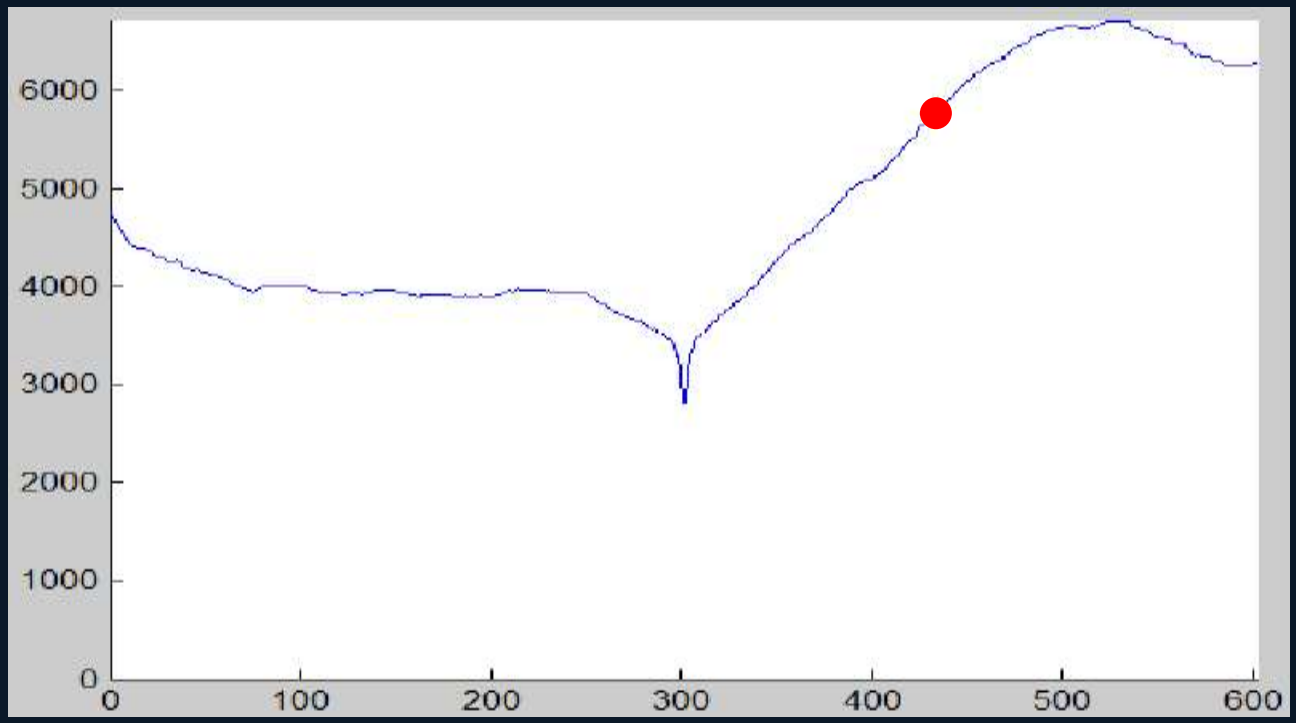

Y
X
MSE
6000
5000
4000
3000
2000
1000
0
0
100
200
300
400
500
600
Amount of X translation


Y
X
MSE
6000
5000
4000
3000
2000
1000
0
0
100
200
300
400
500
600
Amount of X translation

Fixed image

Moving image

“Overlap regions” to be compared

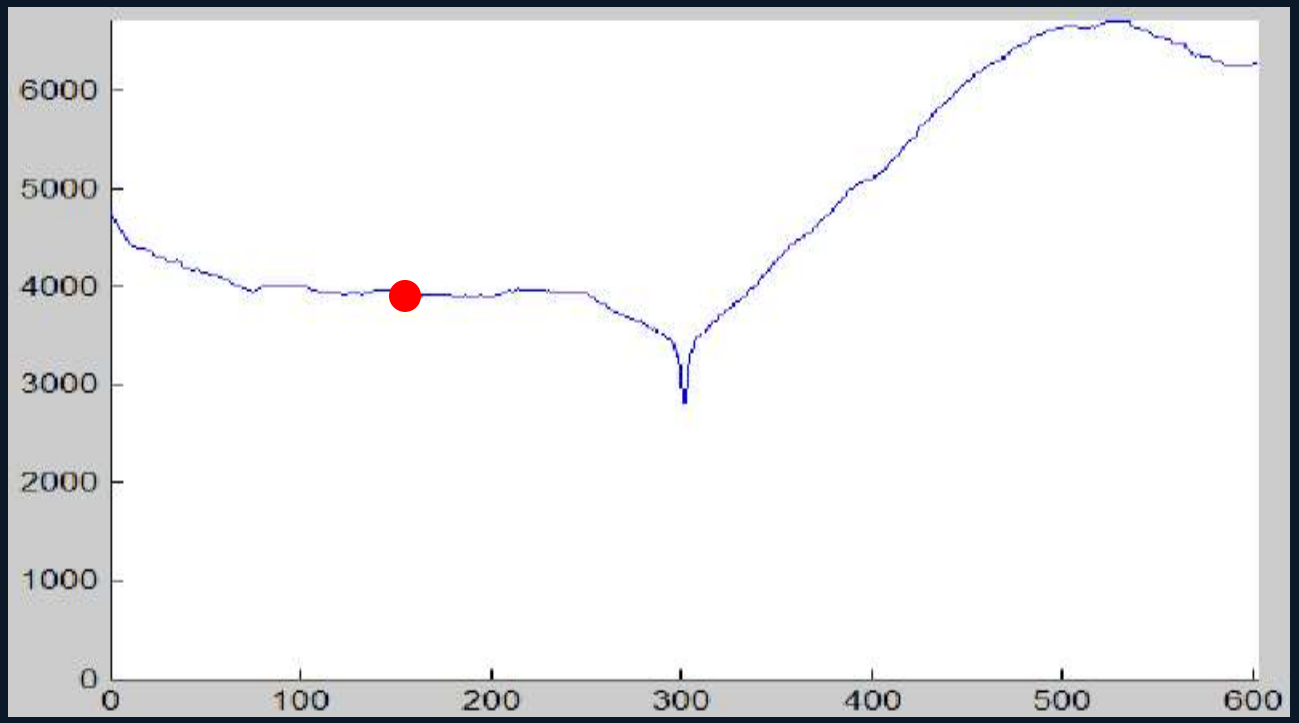


Moving image

Fixed image

"Overlap regions"
to be compared

MSE



Amount of X translation

Image similarity metric: Mean-squared error

Bad behaviour: local minima, optimum value is not zero, large pleateau where we would prefer a downward slope toward the correct solution, slope depends on the initialization.