

Sensor Arrays

CS545 – Machine Learning for Signal Processing



Today's lecture

- Sensor arrays
- Beamforming and localization
- Array-based source separation

A question

- Why do you have two ears/eyes/nostrils?
 - Or two antennae, forked tongues, etc.

Sensor arrays

- Multiple sensors allow us to do more
 - They reveal the “spatial dimension”
- We can thus do things involving space
 - Focus on one direction only
 - Locate the emanating source of a signal

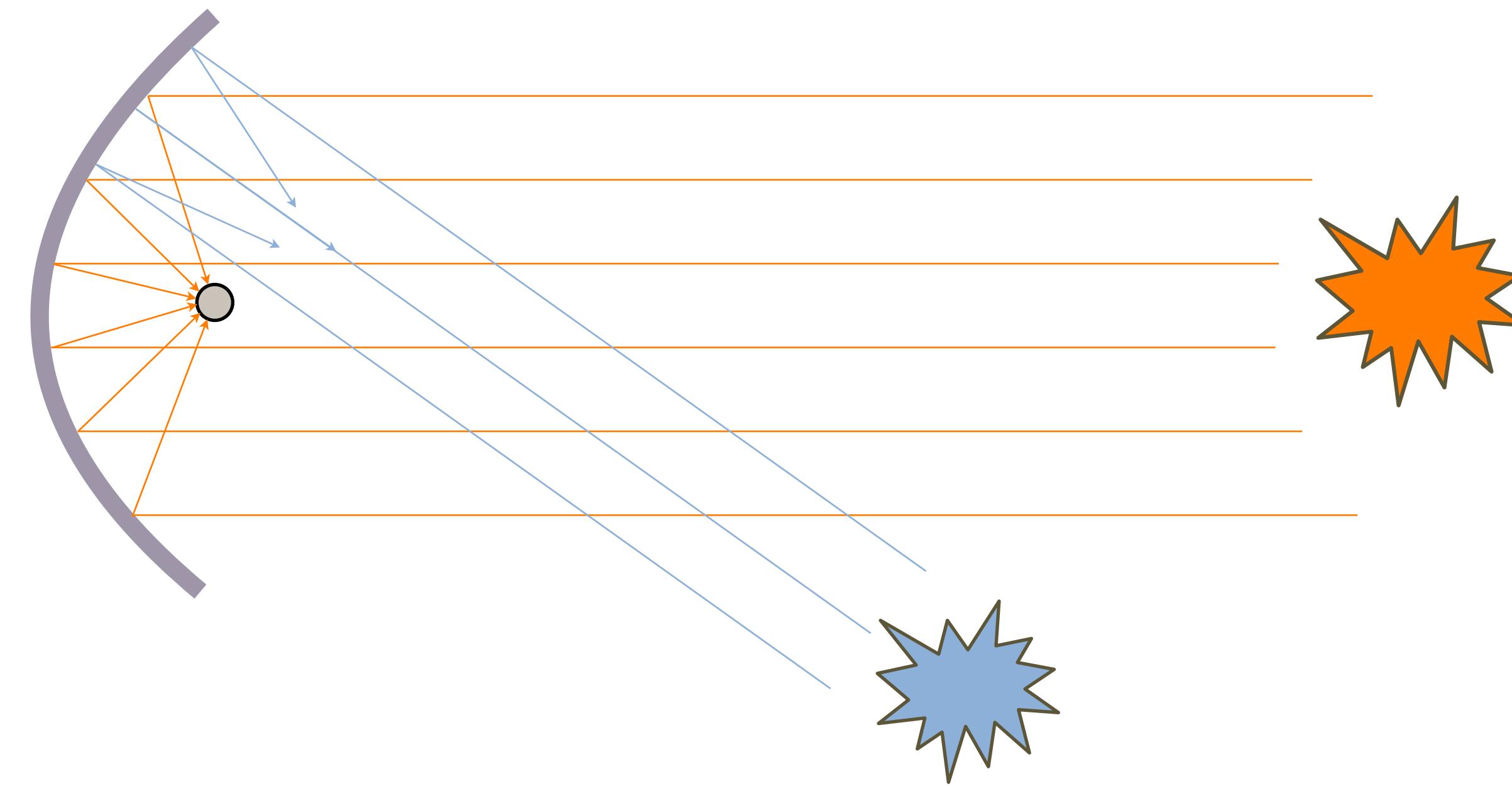
One more question

- Why does your ear have that strange shape on the outside?



Collecting from multiple sources

- A parabolic sensor
 - Focuses towards right ahead and ignores other directions

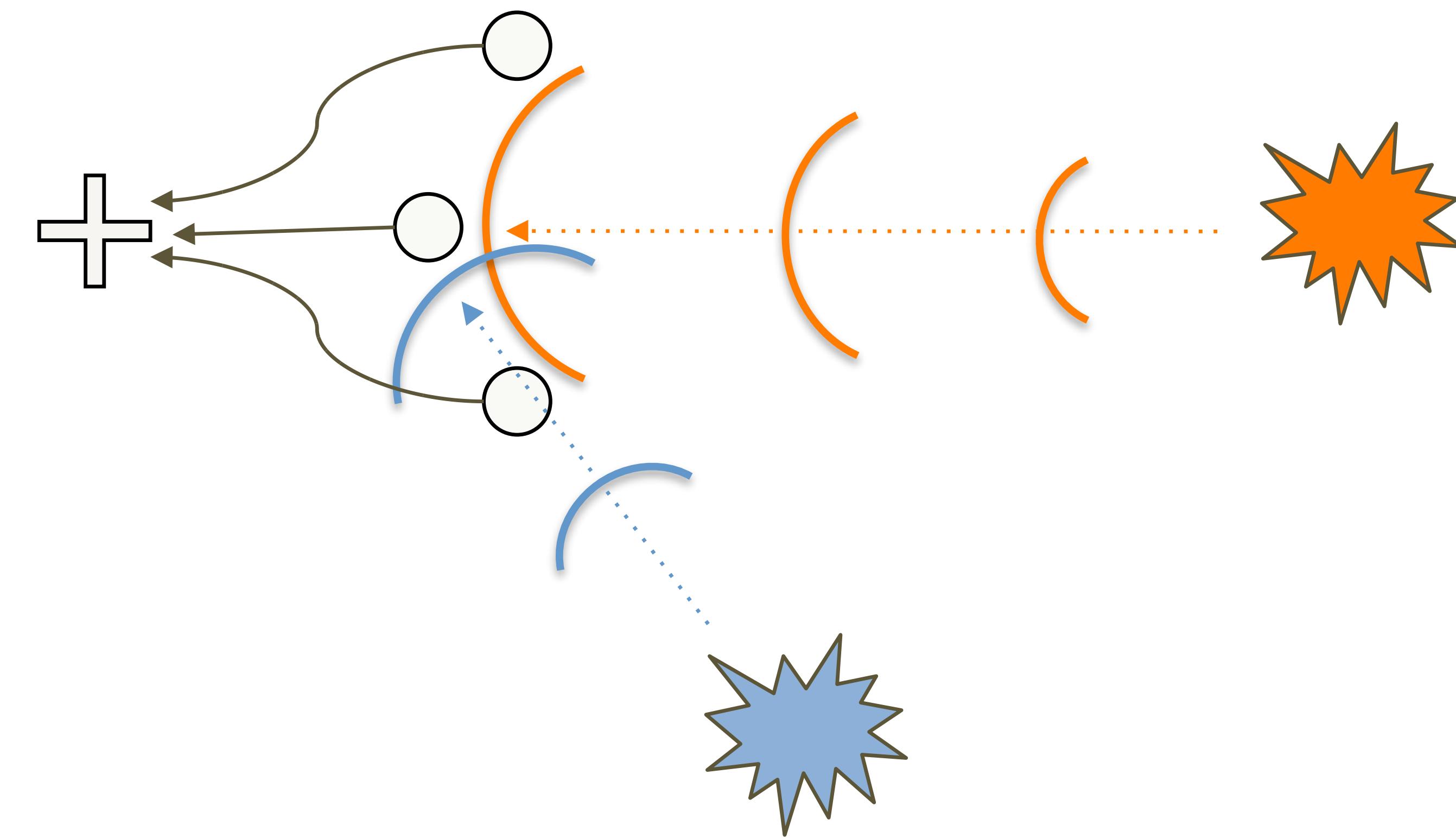


Known for a while

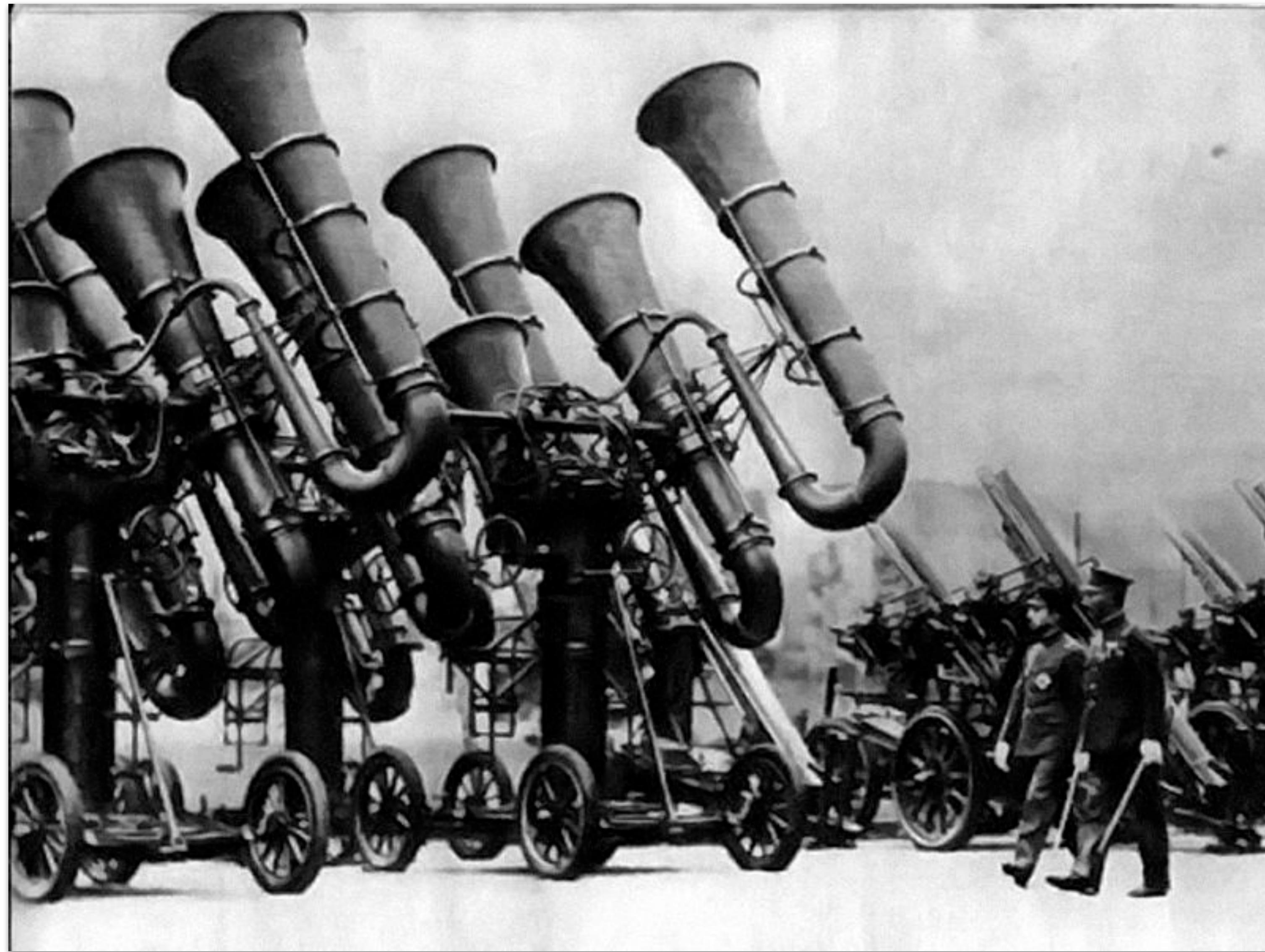


Discretizing the parabolic dish

- Emulating that process with a small number of discrete sensors

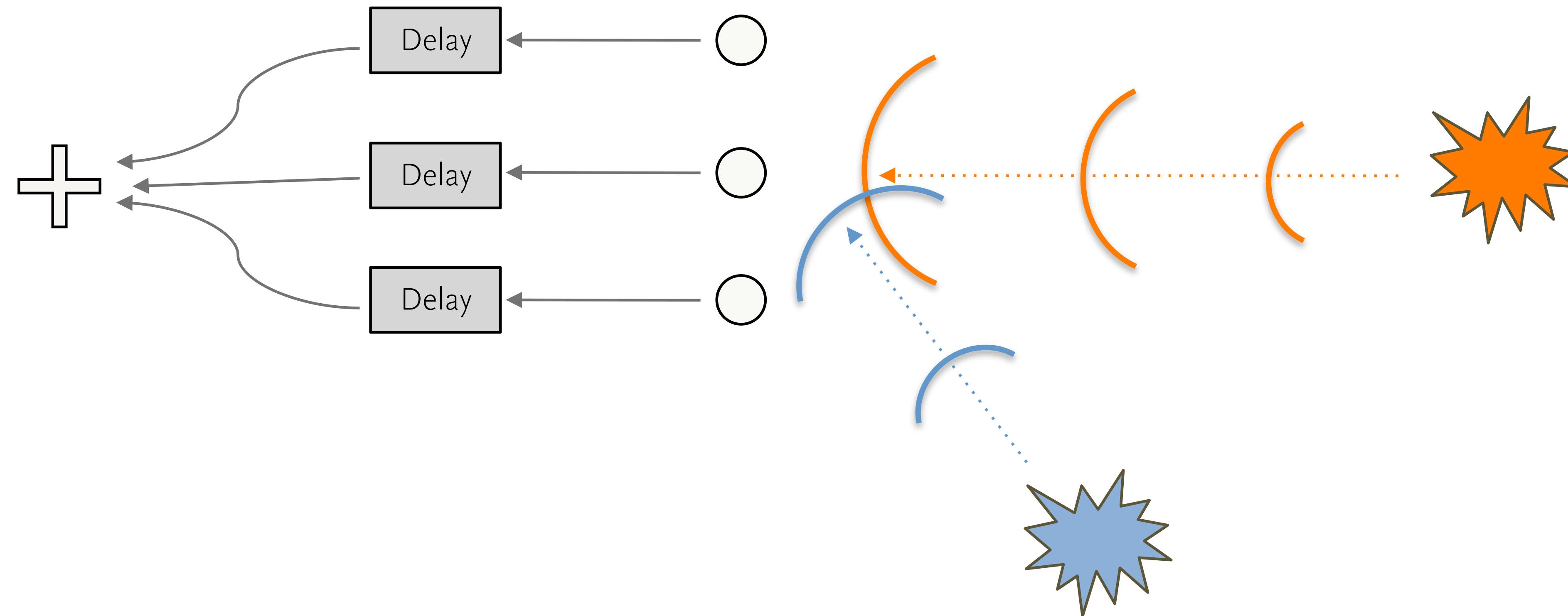


That's been known for a while too



A more flexible setup

- Use adjustable delays for each sensor
 - We can now “steer” the array



The delay and sum beamformer

- Adjust sensor delays to focus on a source coming from a specific direction

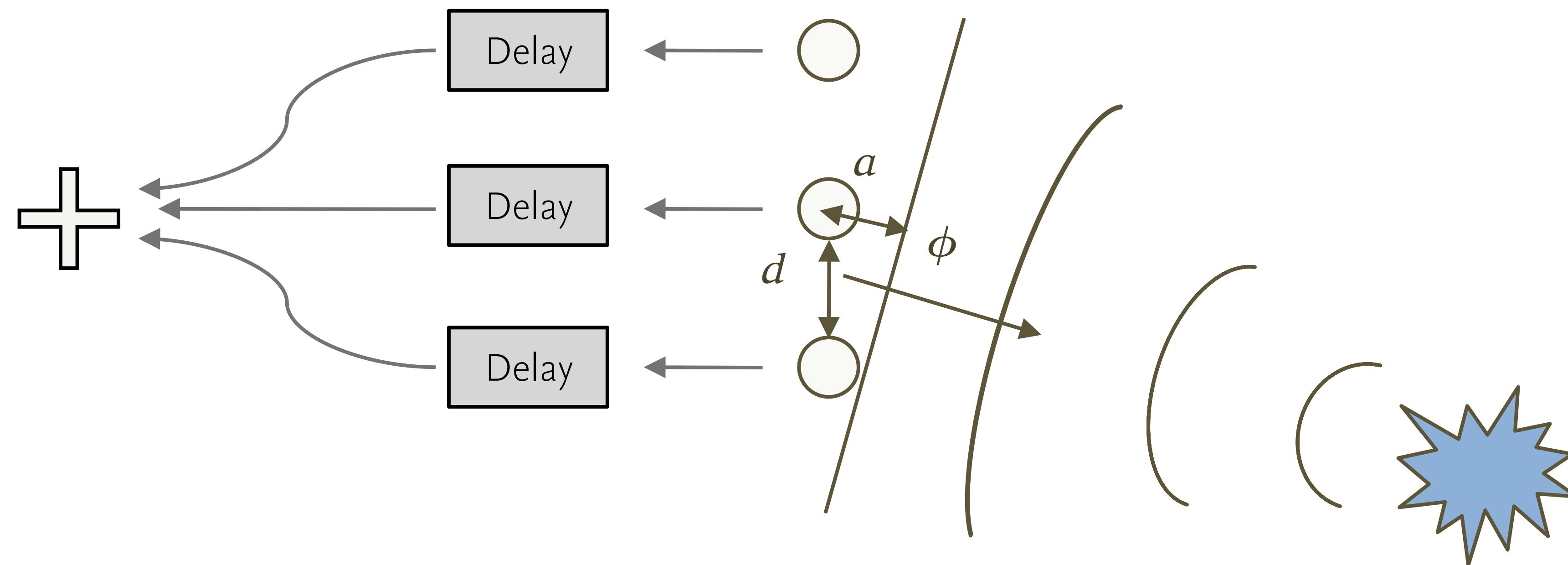
$$y(t) = \sum \tau_i * x_i(t)$$

- The delays depend on the array sensor layout
 - We need to know the layout a priori

Uniform Linear Array (ULA)

- Far field assumption
 - “Wavefront is flat”

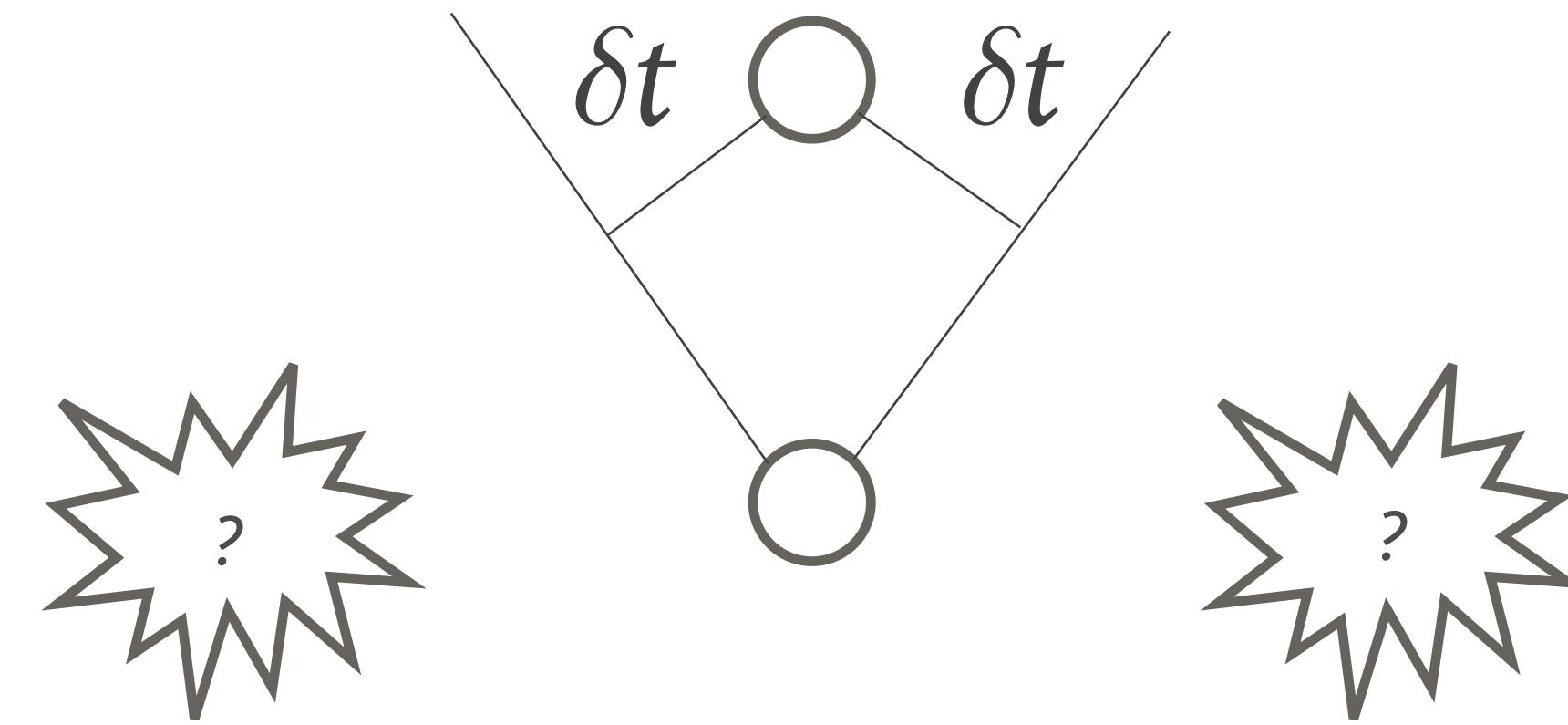
$$\tau_i(t) = \begin{cases} 1 & t = \frac{d \cos \phi}{c} \\ 0 & \text{otherwise} \end{cases}$$



Some problems

- Direction confusion

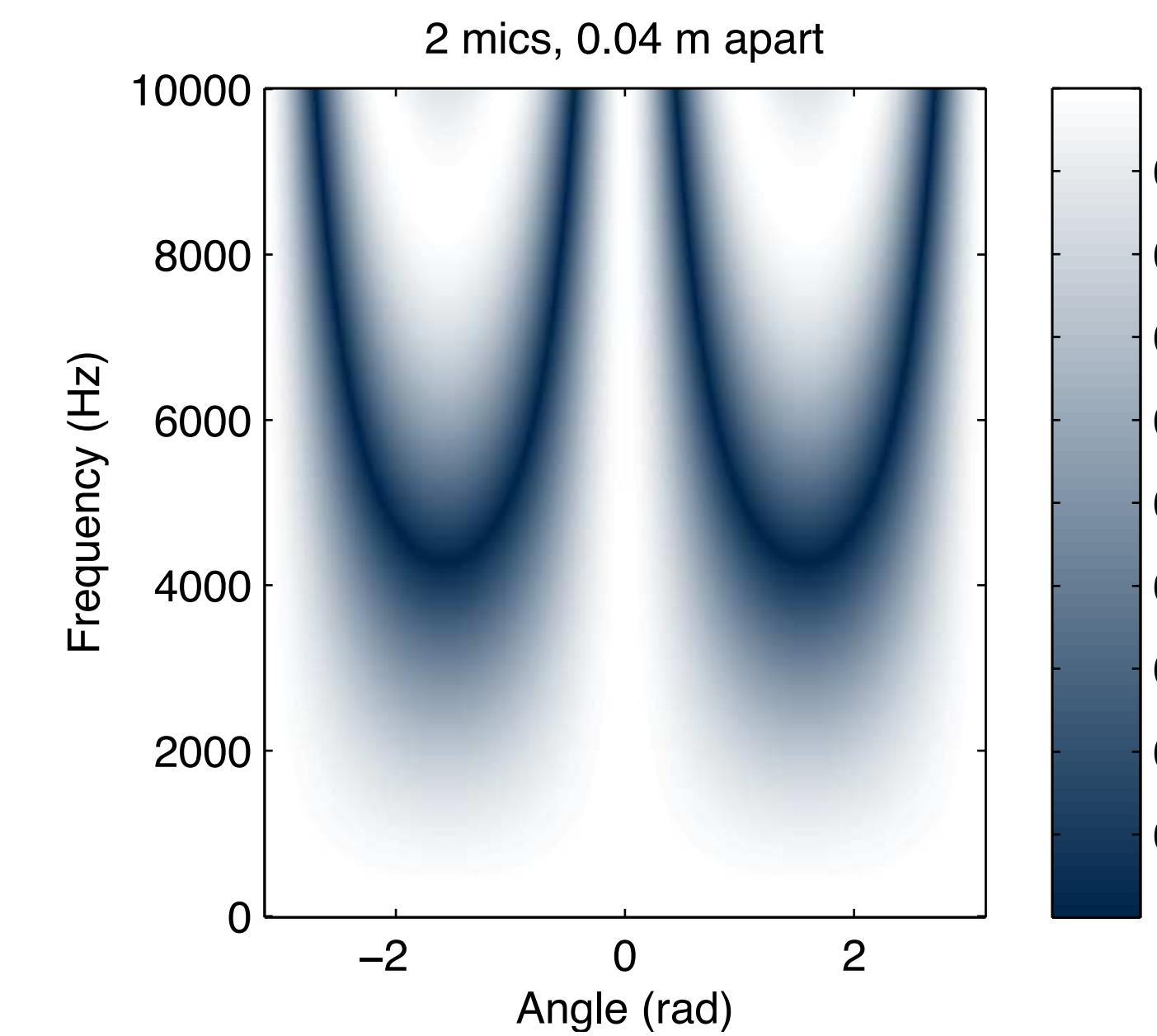
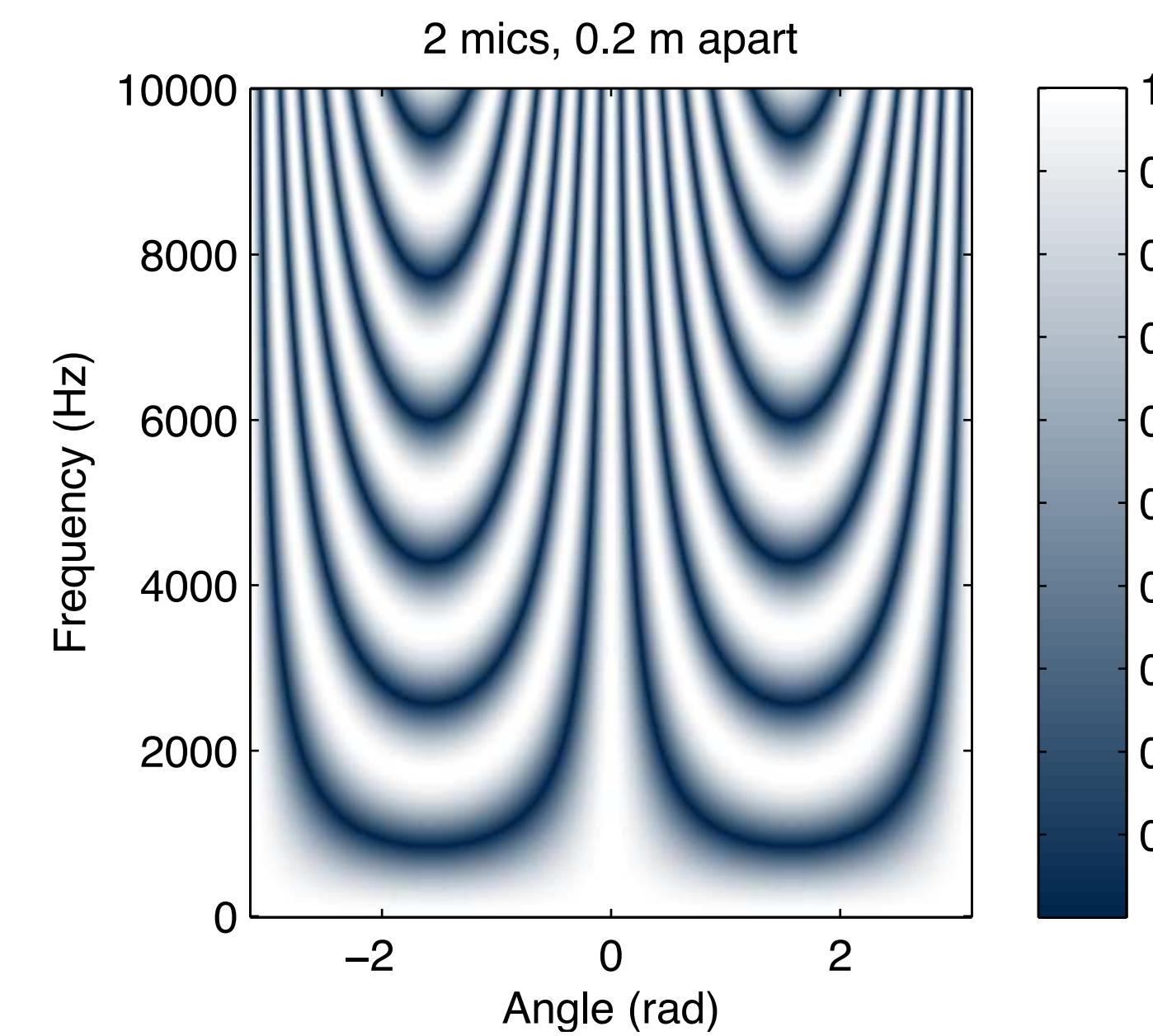
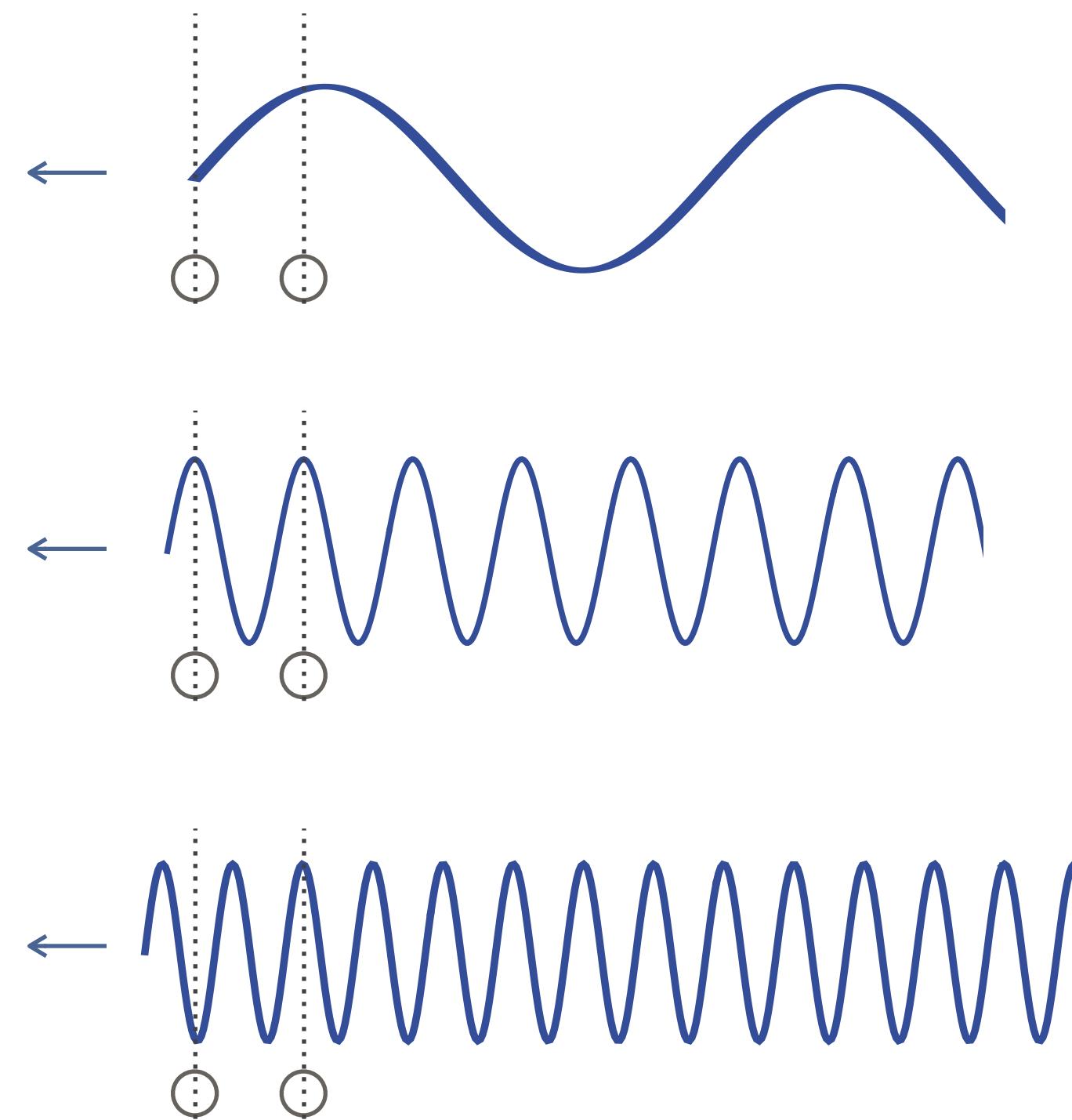
$$\delta t = \frac{d \cos \phi}{C} = \frac{d \cos(-\phi)}{C}$$



- Need at least $D+1$ sensors for D dimensions
- But regardless, the more the sensors the better!

The array response

- Array response changes across frequencies
- Spatial aliasing = ambiguities in high frequencies
 - Sensor distance must be less than half wavelength of highest freq



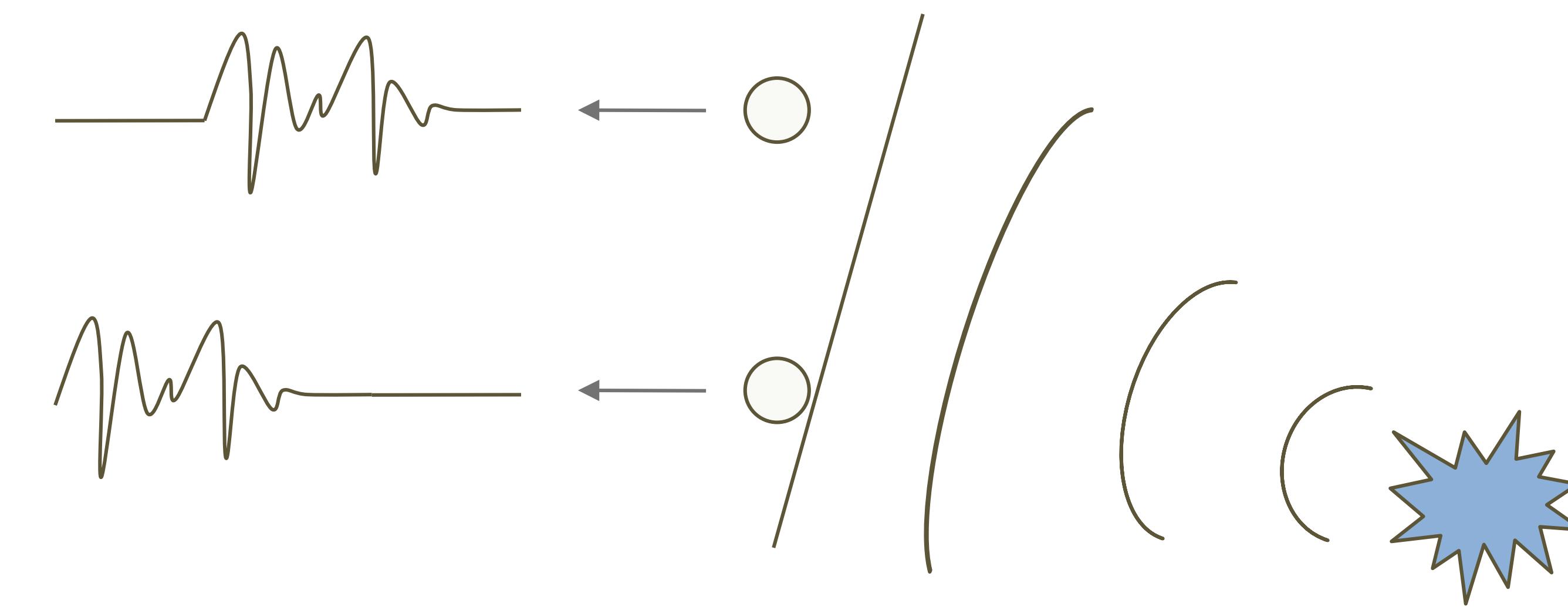
Many more things ...

- Sensors might have non-uniform response
 - Across varying directions or frequencies
- We can use dense adaptive filters instead of delays
 - Suppress certain areas, boost others
- Rich literature in radio, sonar and audio

Extreme beamforming demo

Localization

- Determine delays that align with input
 - If you know the delays you can determine the angle of most likely incidence



Localization of a single source

- Cross-correlate sensor inputs

$$c_{i,j}(t) = x_i(t) * x_j(-t)$$

- Peak denotes relative delay between inputs

- Convert delays to angle

$$\phi_{i,j} = \cos^{-1} \frac{\tau_{i,j} C}{d_{i,h}}$$

- But there's a problem

Localization of multiple sources

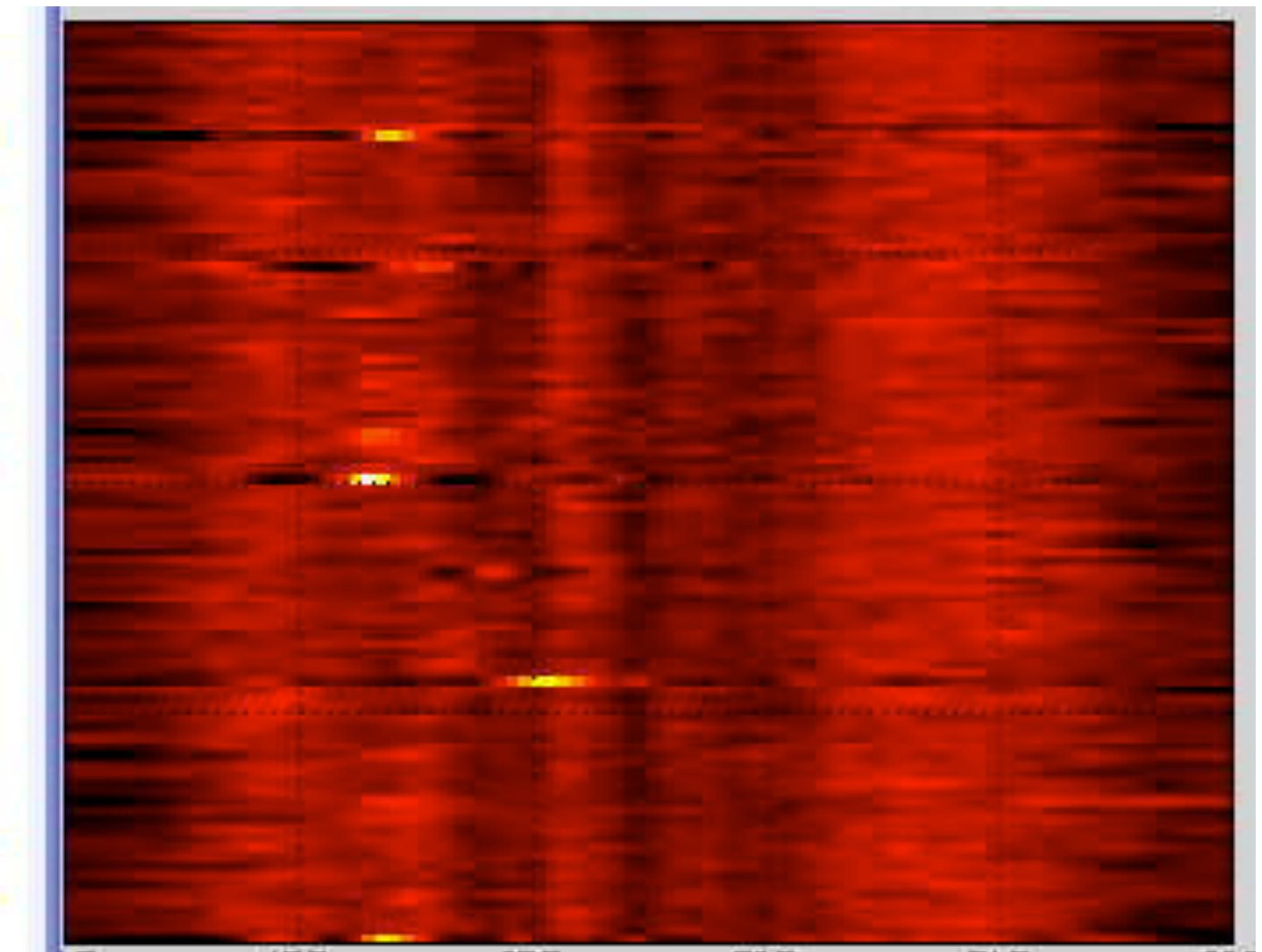
- If we have multiple sources estimation of one delay is not feasible
- Instead we can measure the energy from each location in the space we operate
 - Steer the array beam and measure signal
- This provides a spatial energy map

Multi-source demo

Situation video



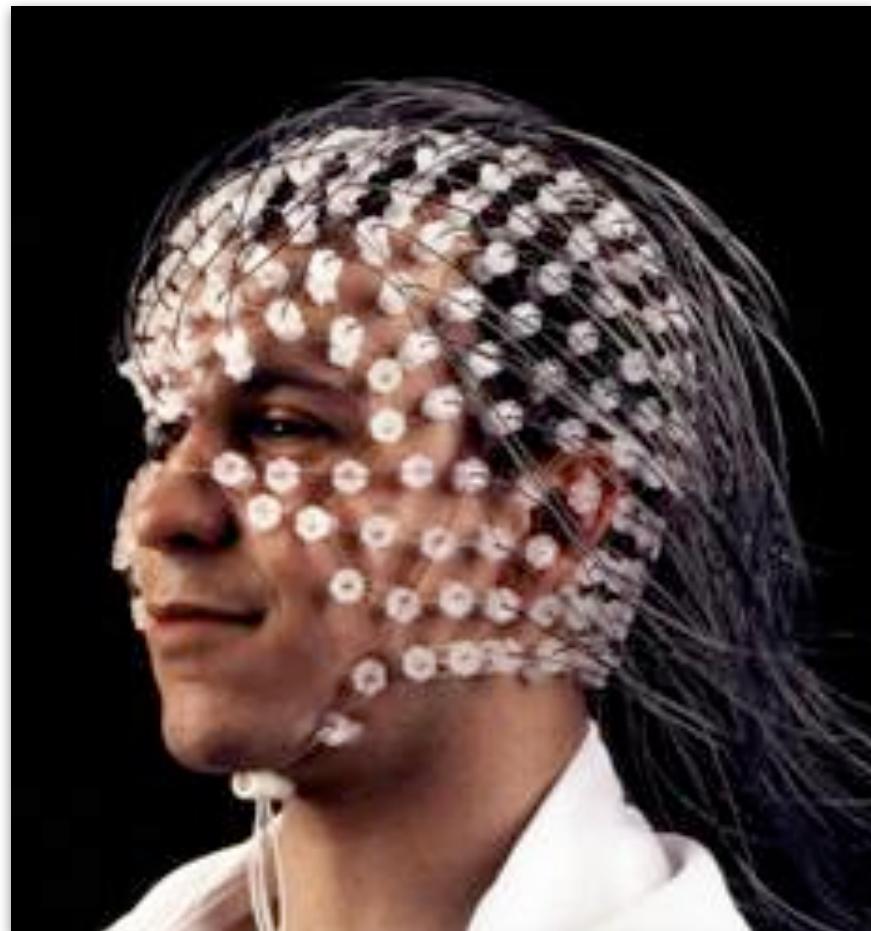
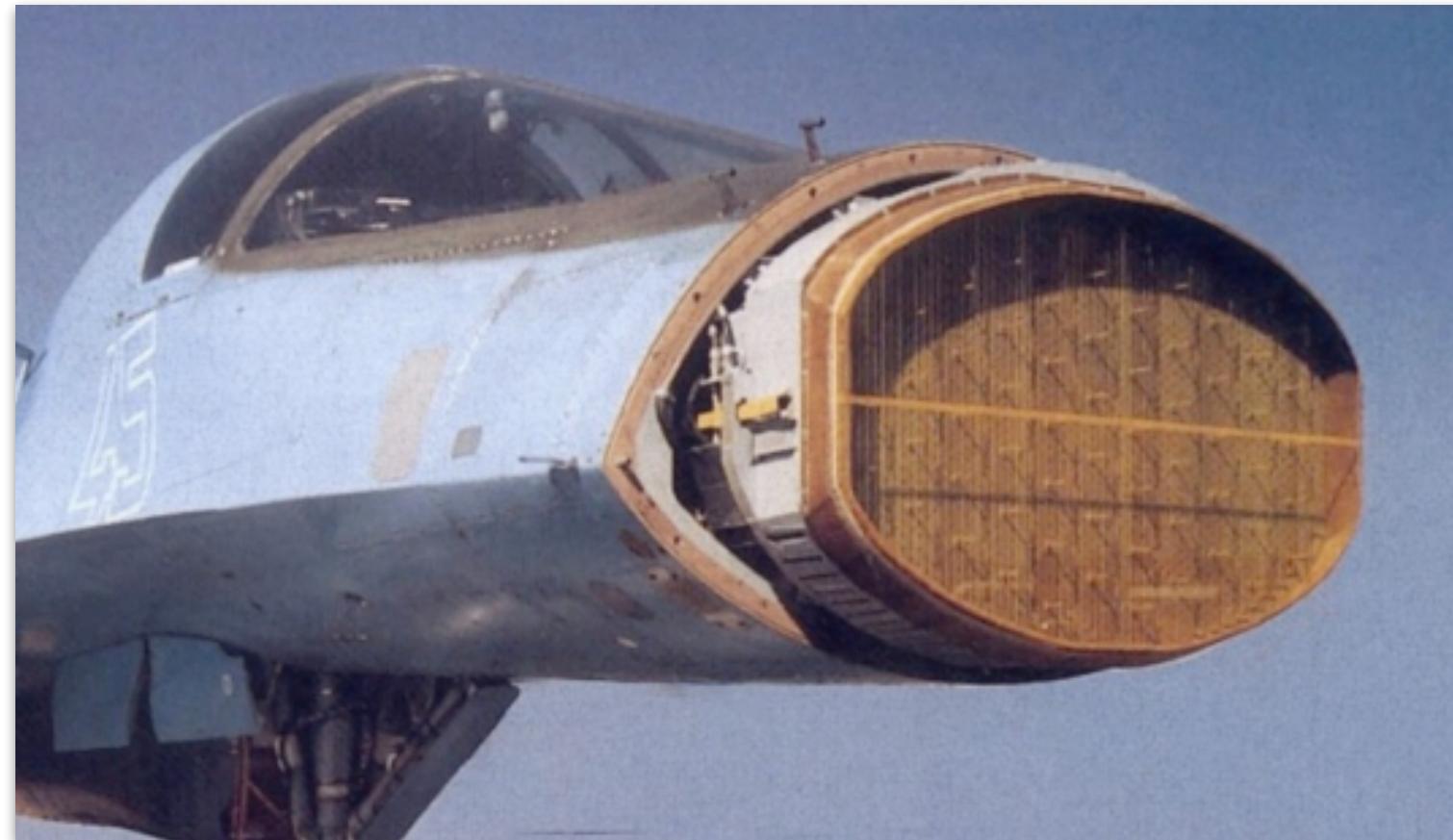
Energy over angle and time



Angle

Time

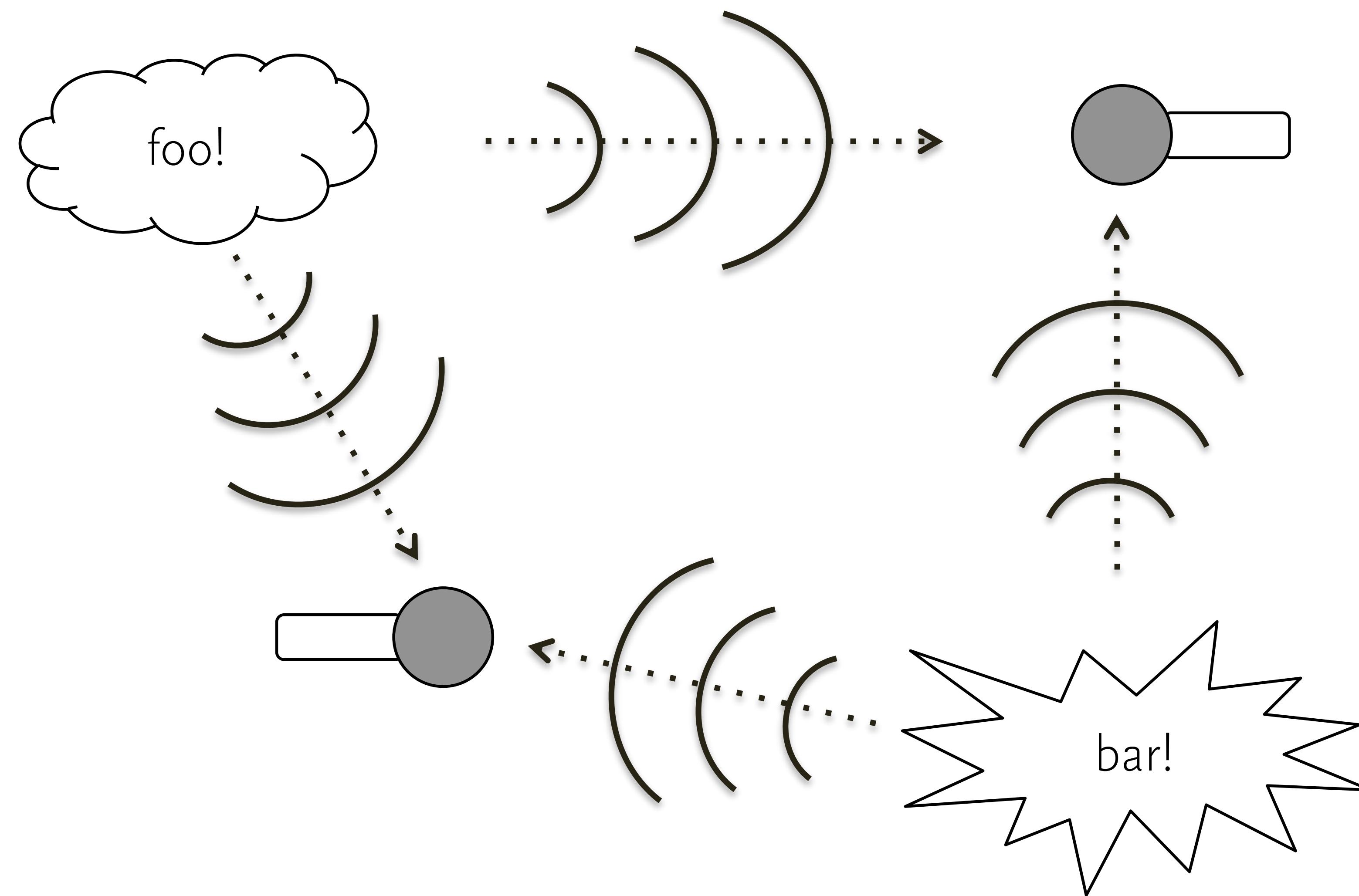
Arrays are everywhere



Using more machine learning

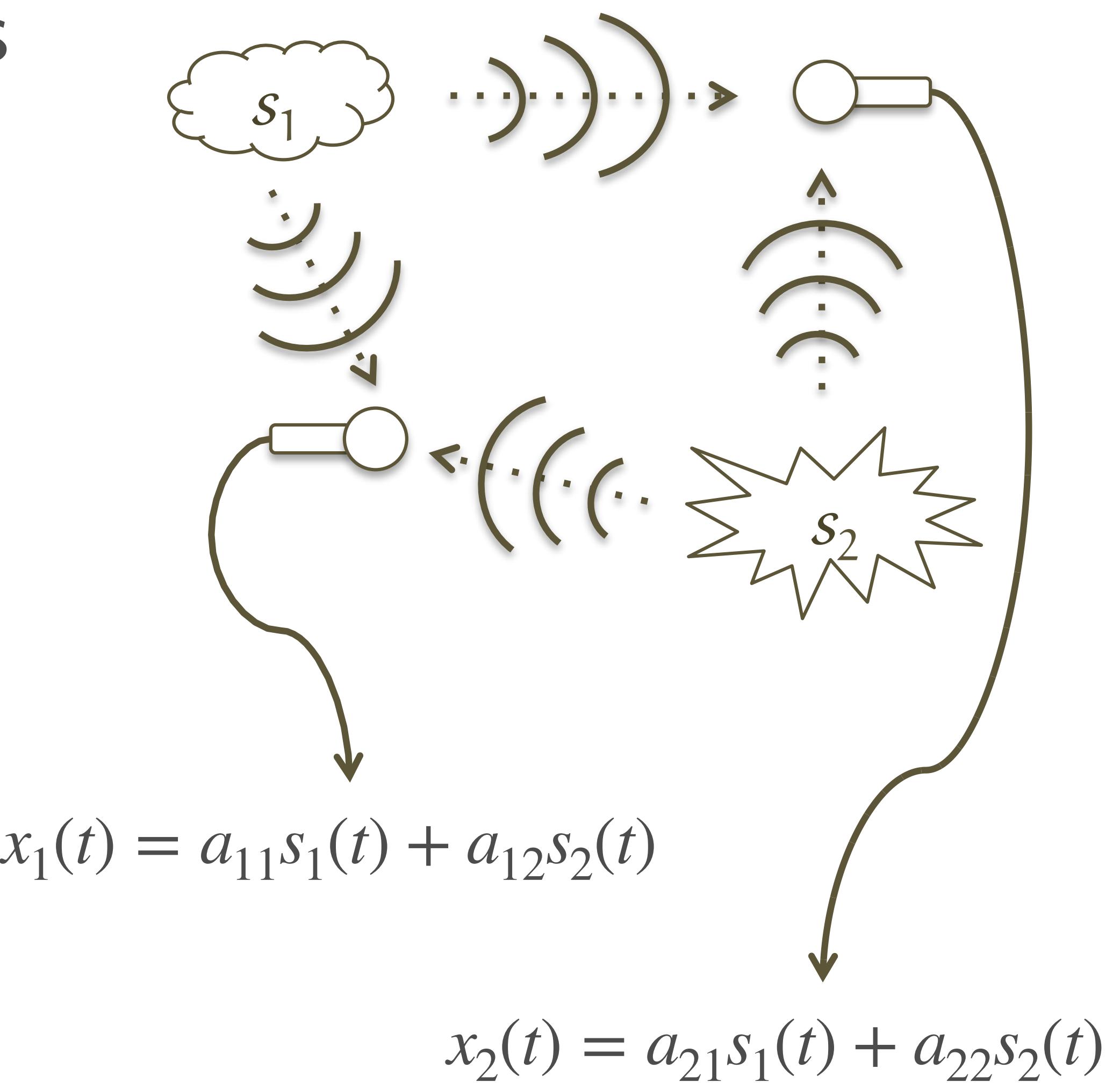
- Most array literature is DSP-based
 - Beamforming has its limitations
- Source separation and blind methods
 - Machine learning applications on arrays

A “simple” audio problem



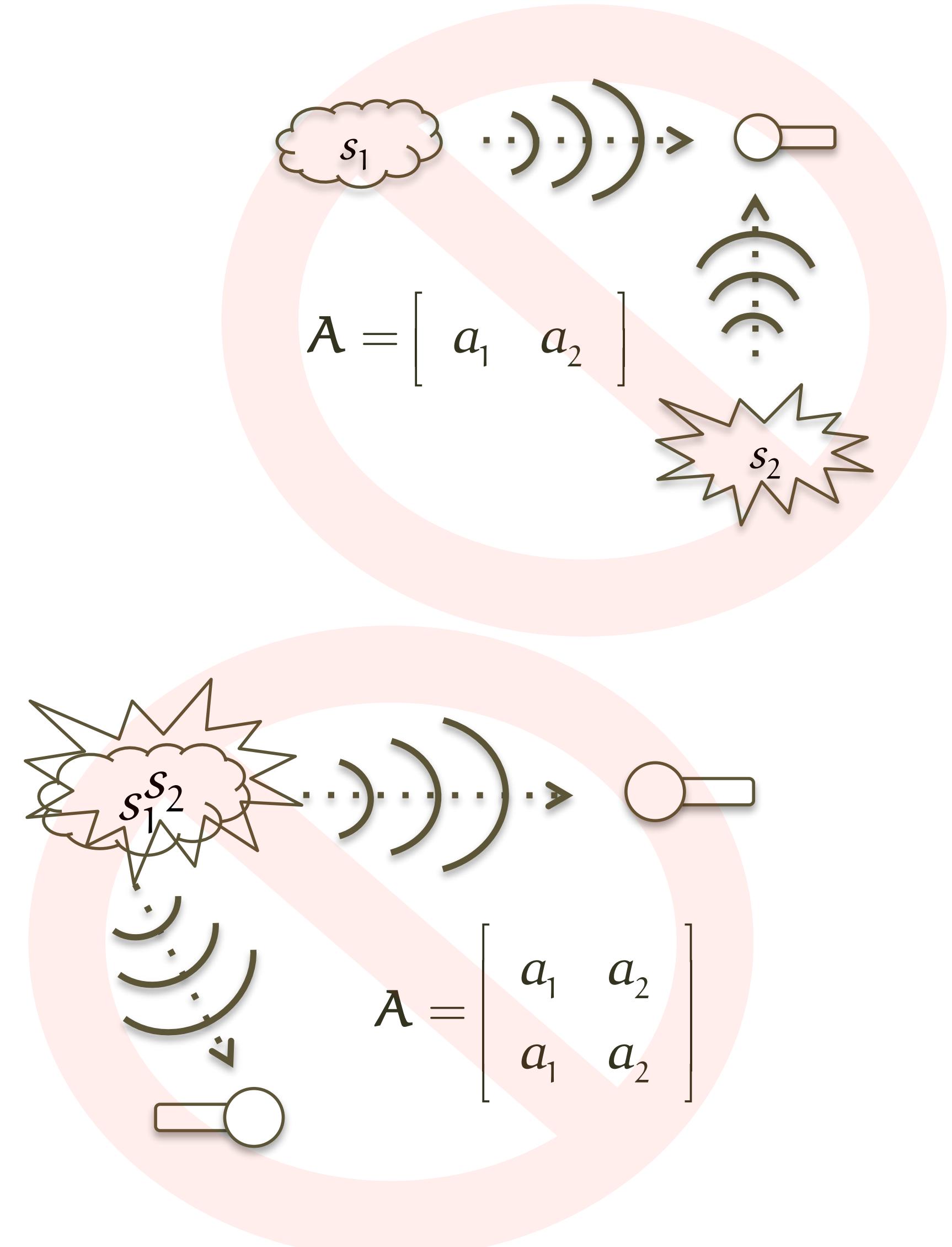
Formalizing the problem

- Each mic receives a mix of both sounds
 - Sound waves superimpose linearly
 - Ignoring propagation delays for now
- The simplified mixing model is:
$$\mathbf{x}(t) = \mathbf{A} \cdot \mathbf{s}(t)$$
- How do we solve this system and find the original signals $\mathbf{s}(t)$?



When can we solve this?

- The mixing equation is:
$$\mathbf{x}(t) = \mathbf{A} \cdot \mathbf{s}(t)$$
- Our estimates of $\mathbf{s}(t)$ will be:
$$\hat{\mathbf{s}}(t) = \mathbf{A}^{-1} \cdot \mathbf{x}(t)$$
- To recover $\mathbf{s}(t)$, \mathbf{A} must be invertible:
 - We need as many mics as sources
 - The mics/sources must not coincide
 - All sources must be audible
- Otherwise this is a different story ...



A simple example

- A simple invertible problem

$$\boxed{\text{x}(t)} = \mathbf{A} \cdot \boxed{\text{s}(t)}$$

The diagram illustrates a linear system for signal processing. On the left, a bracket labeled $\text{x}(t)$ encloses two blue waveforms representing mixed signals. An equals sign follows, with the letter \mathbf{A} positioned above a matrix multiplication symbol. To the right of the multiplication symbol is another bracket labeled $\text{s}(t)$, which encloses two blue waveforms representing the original sources. The matrix \mathbf{A} is defined as:

$$\mathbf{A} = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}$$

- $\text{s}(t)$ contains two structured waveforms
- \mathbf{A} is invertible (but we don't know its values)
- $\text{x}(t)$ looks messy, doesn't reveal $\text{s}(t)$ clearly
- Known as the Blind Source Separation problem (BSS)
 - *Blind* because we don't know anything about the original sources

What to look for

- We can only use $\mathbf{x}(t)$

$$\mathbf{x}(t) = \mathbf{A} \cdot \mathbf{s}(t)$$

- Is there a property we can take advantage of?
 - Yes! We know that different sounds are “different”
- The plan: Find a solution that enforces “differentness”

A first try

- Find $s(t)$ by minimizing correlations
- Our estimate of $s(t)$ is computed by: $\hat{s}(t) = \mathbf{W} \cdot \mathbf{x}(t)$
 - If $\mathbf{W} \approx \mathbf{A}^{-1}$ then we have a good solution

- The goal is that the outputs become uncorrelated:

$$\langle \hat{s}_i(t) \cdot \hat{s}_j(t) \rangle = 0, \forall i \neq j$$

- We assume here that our signals are zero mean
- So the problem to solve is: $\arg \min_{\mathbf{W}} \left\langle \sum_k w_{ik} x_k(t) \cdot \sum_k w_{jk} x_k(t) \right\rangle, \forall i \neq j$

Seen that before?

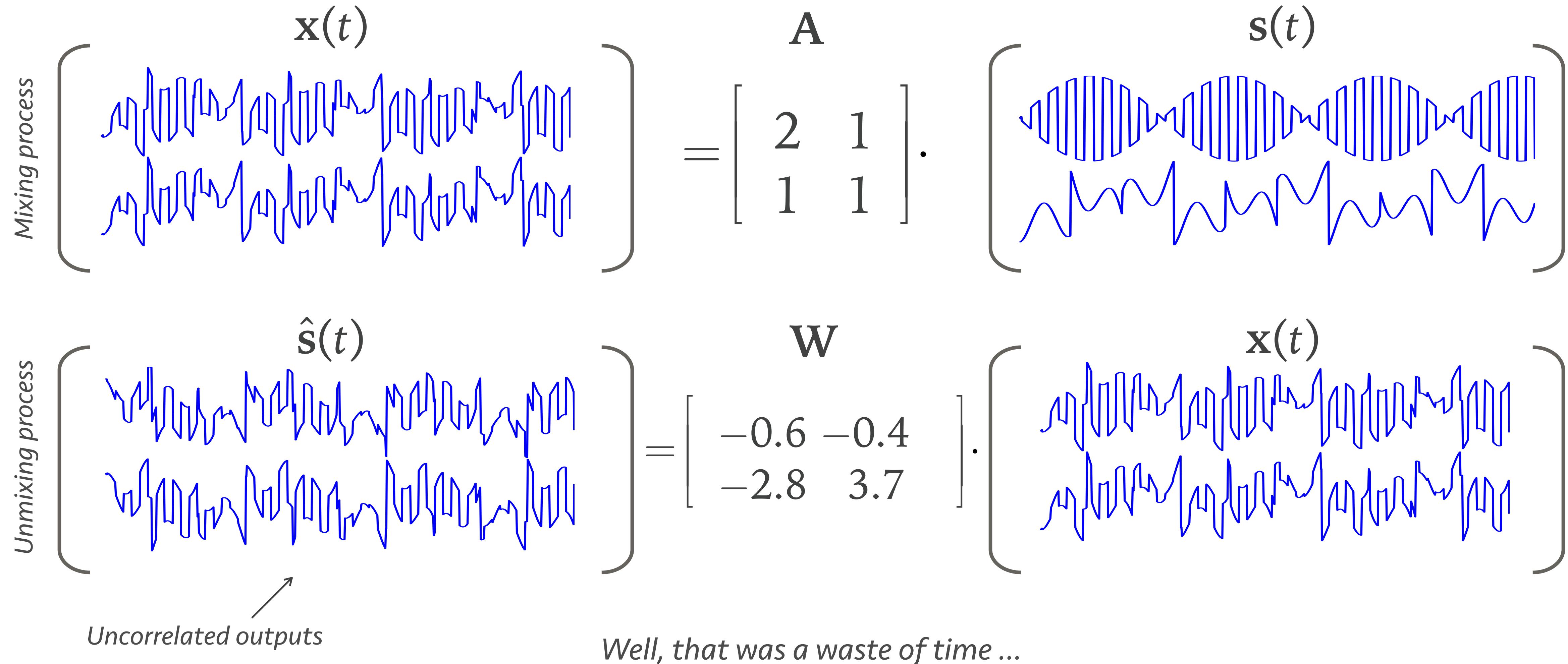
- This is PCA!

$$\begin{bmatrix} \hat{s}_1(t) & \dots & \hat{s}_1(T) \\ \hat{s}_2(t) & \dots & \hat{s}_2(T) \\ \hat{s}_3(t) & \dots & \hat{s}_3(T) \\ \hat{\mathbf{s}}(1), & \dots, & \hat{\mathbf{s}}(T) \end{bmatrix} = \mathbf{W} \cdot \begin{bmatrix} x_1(t) & \dots & x_1(T) \\ x_2(t) & \dots & x_2(T) \\ x_3(t) & \dots & x_3(T) \\ \mathbf{x}(1), & \dots, & \mathbf{x}(T) \end{bmatrix} \Rightarrow$$
$$\Rightarrow \hat{\mathbf{S}} = \mathbf{W} \cdot \mathbf{X}$$

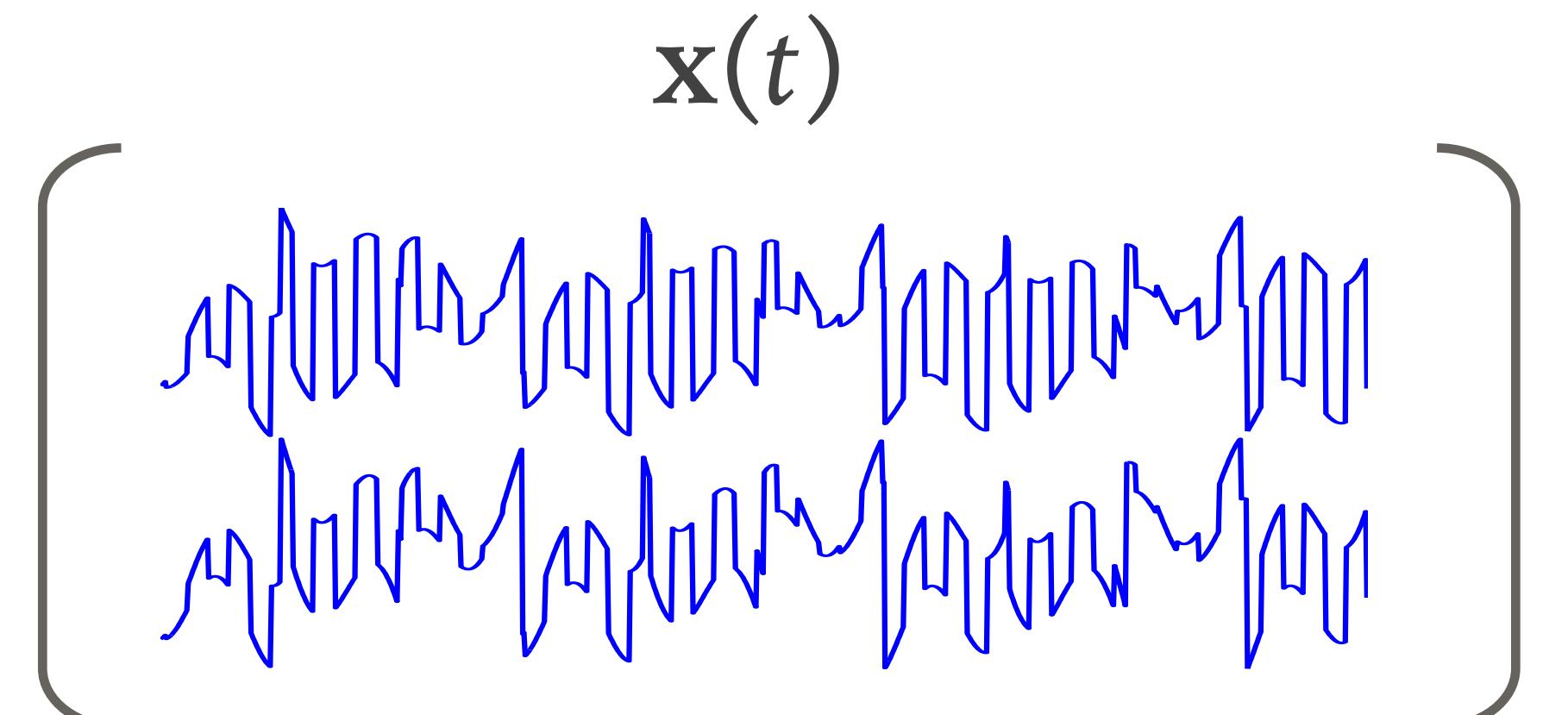
- Easy to compute with a simple decomposition:

$$\mathbf{W} = \text{eig}(\mathbf{X} \cdot \mathbf{X}^\top)$$

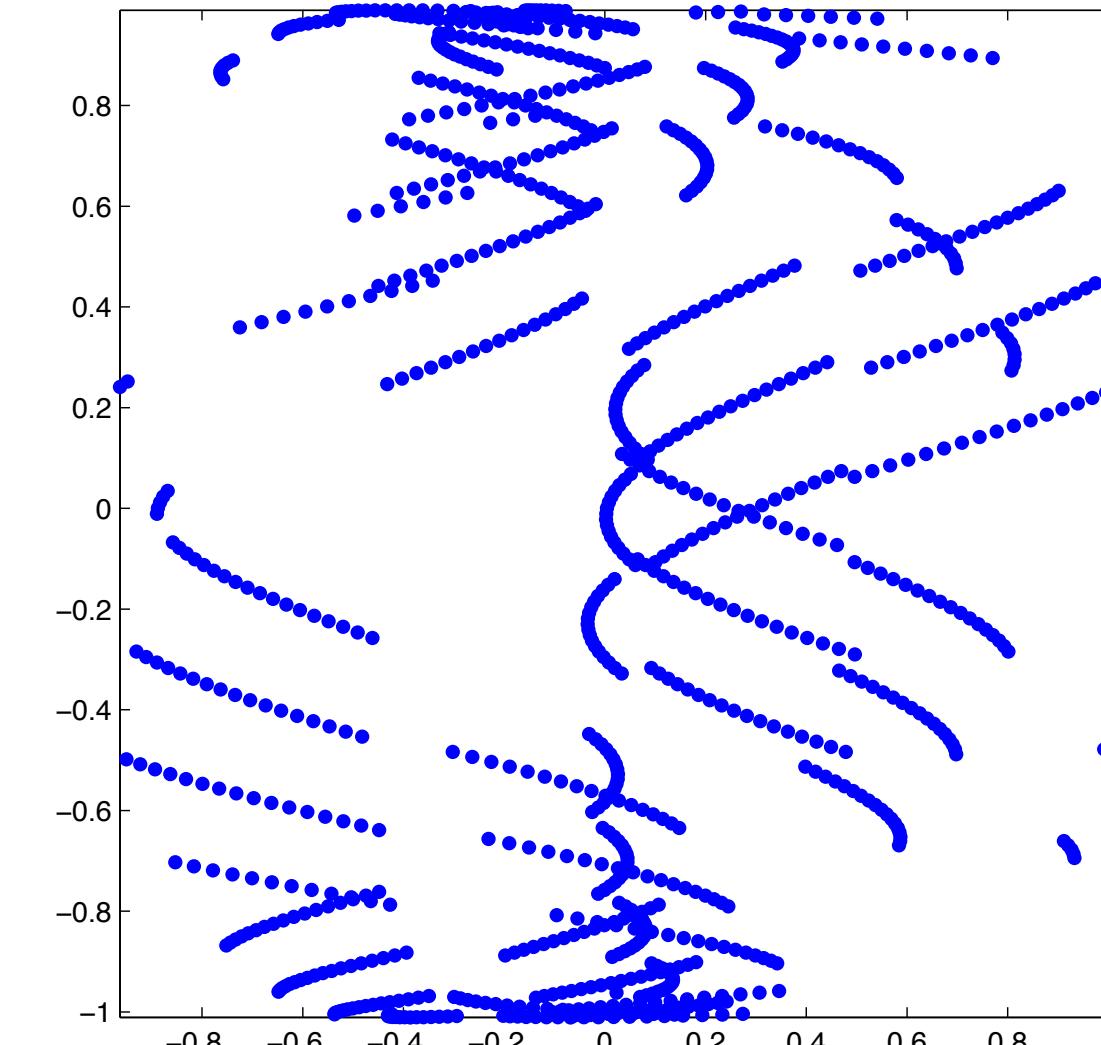
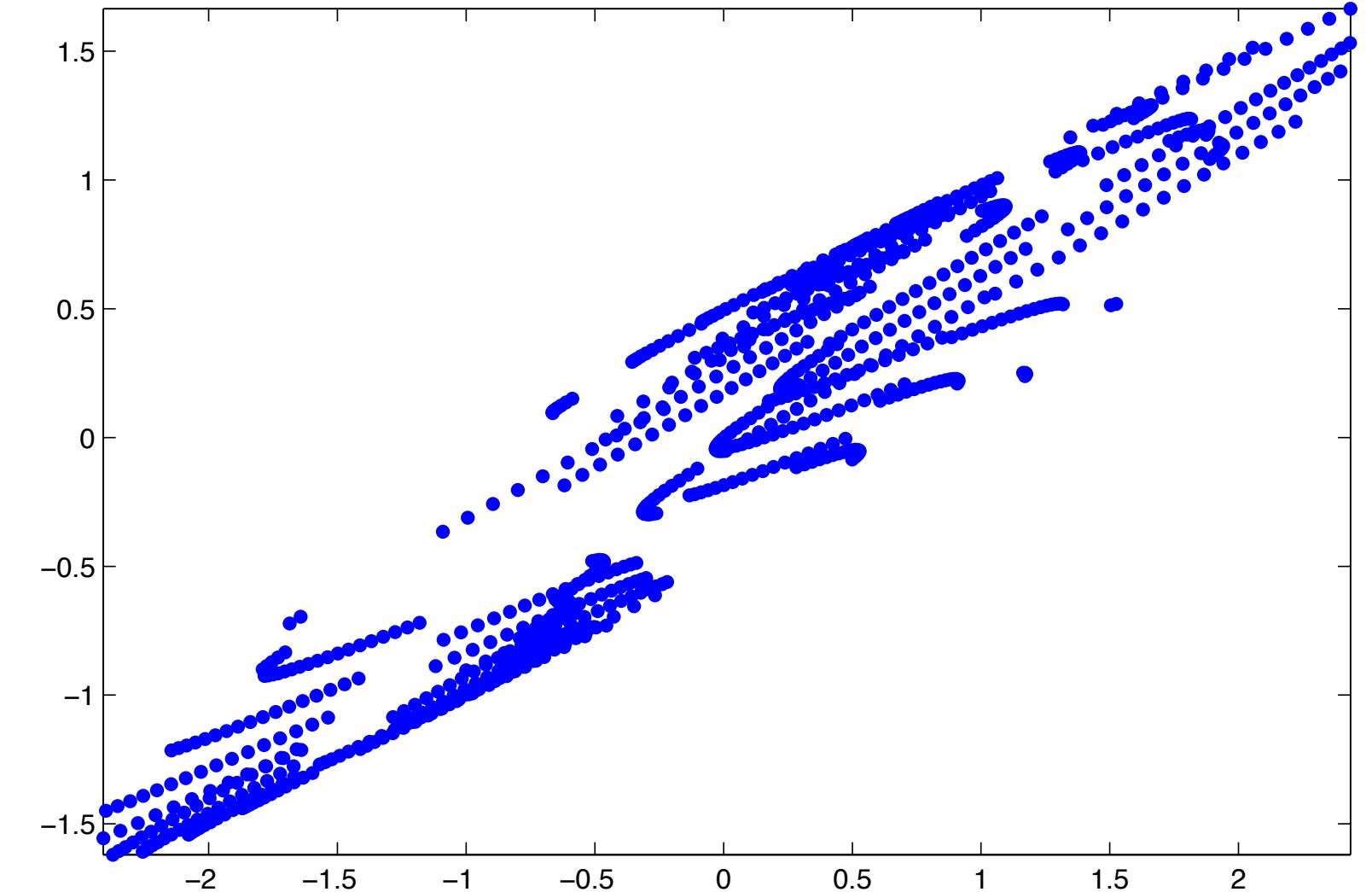
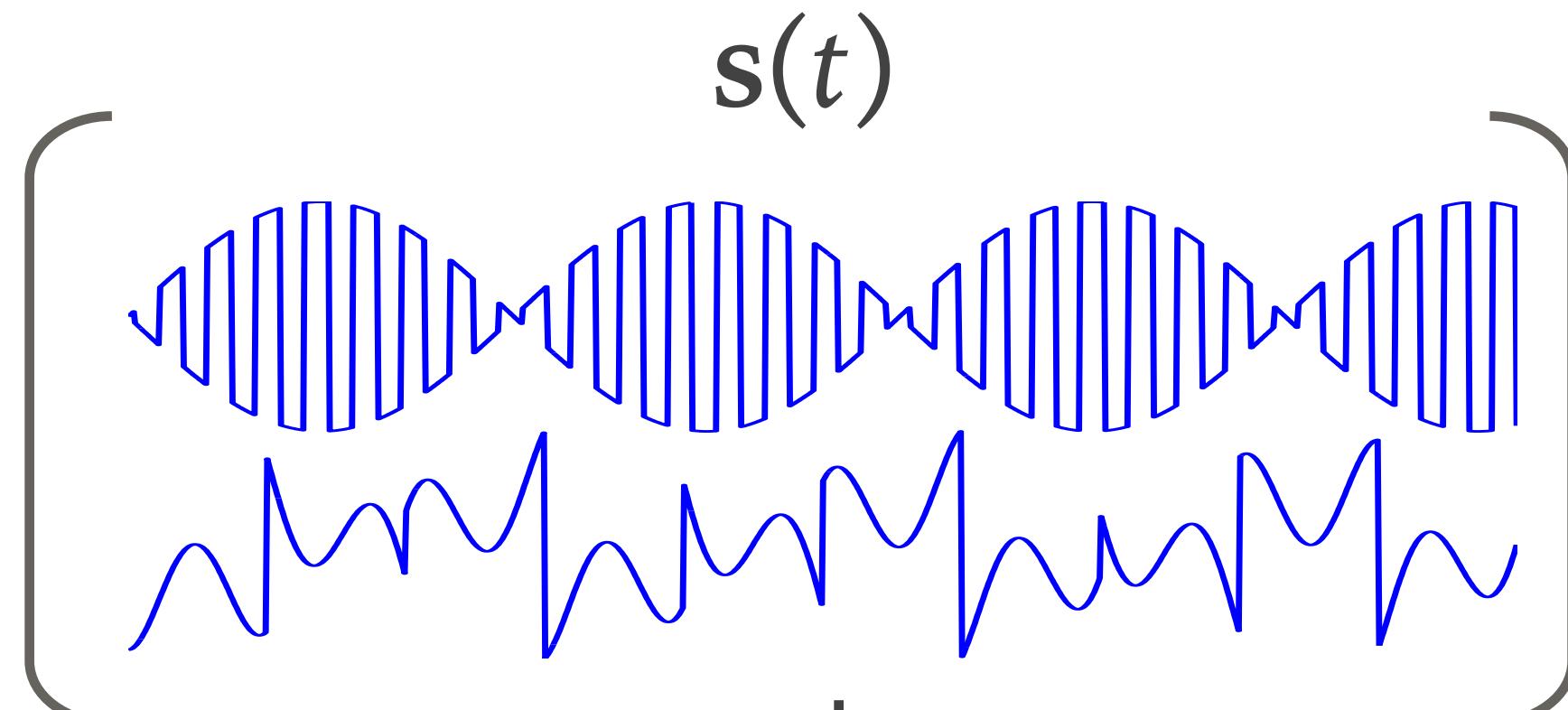
So how well does this work?



What went wrong?

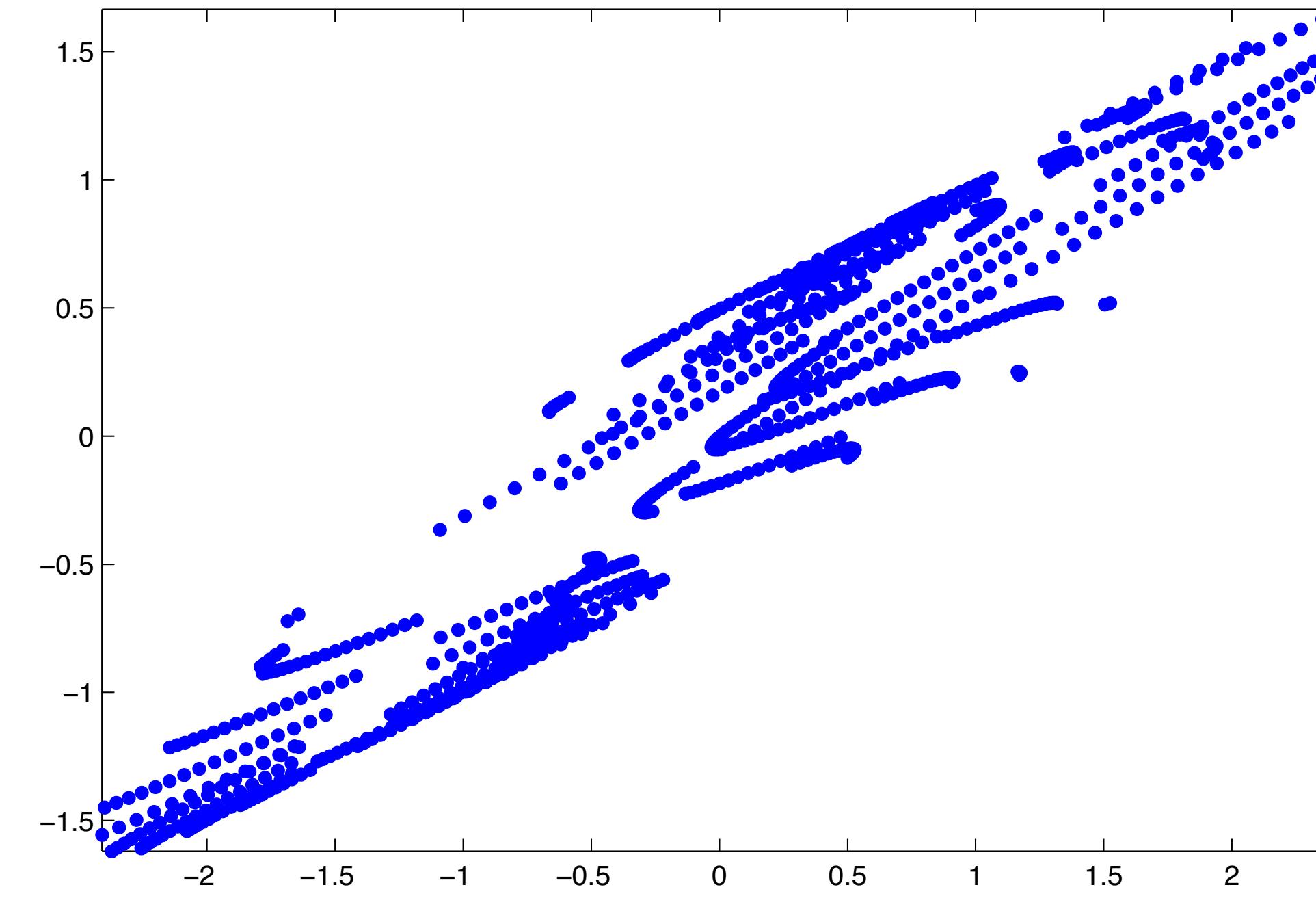


$$= \begin{bmatrix} A & & \\ & 2 & 1 \\ & 1 & 1 \end{bmatrix} .$$

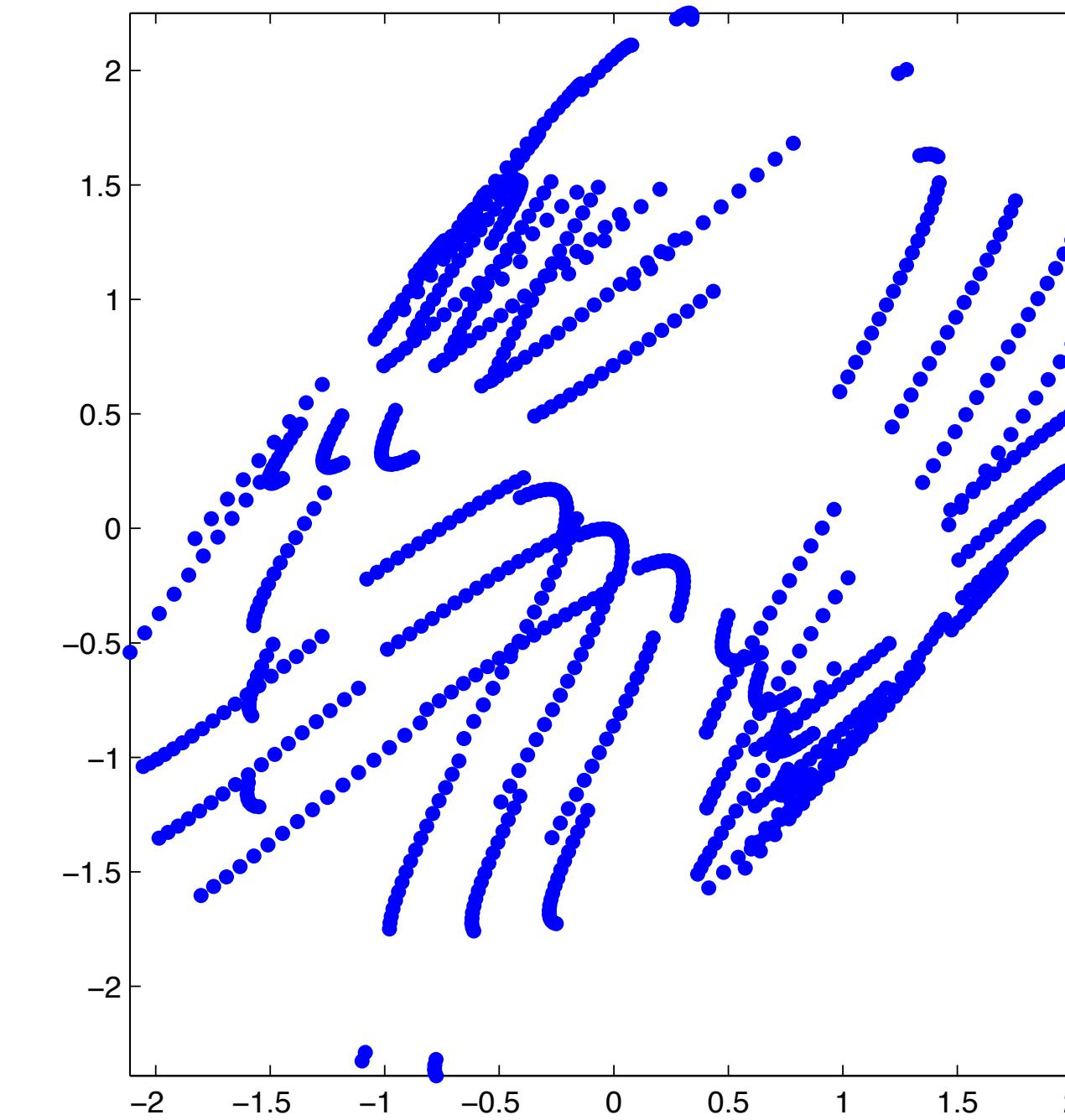


PCA does the wrong thing

What are the mixture eigenvectors?



Scaling/rotating to get decorrelation



What's wrong?

- Our data is not Gaussian
 - Been there before, right?
- We shouldn't use decorrelation, we need “statistical independence”
 - We should use independent component analysis!

ICA for array mixtures

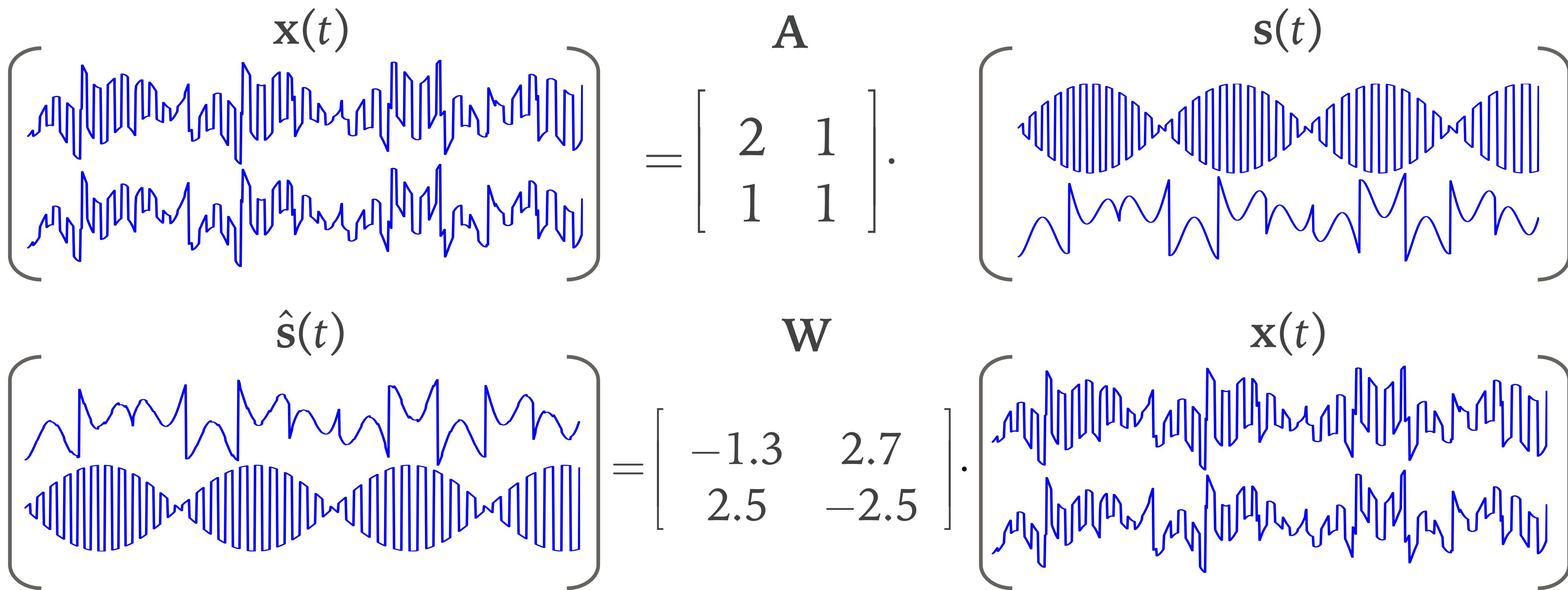
- Given the input:

$$\hat{s}(t) = \mathbf{W} \cdot \mathbf{x}(t)$$

- Find \mathbf{W} so that the transformed data has *maximally statistically independent outputs*
- So far in ICA we looked at the features (\mathbf{W}), now we care about the transformed data itself (\mathbf{s})

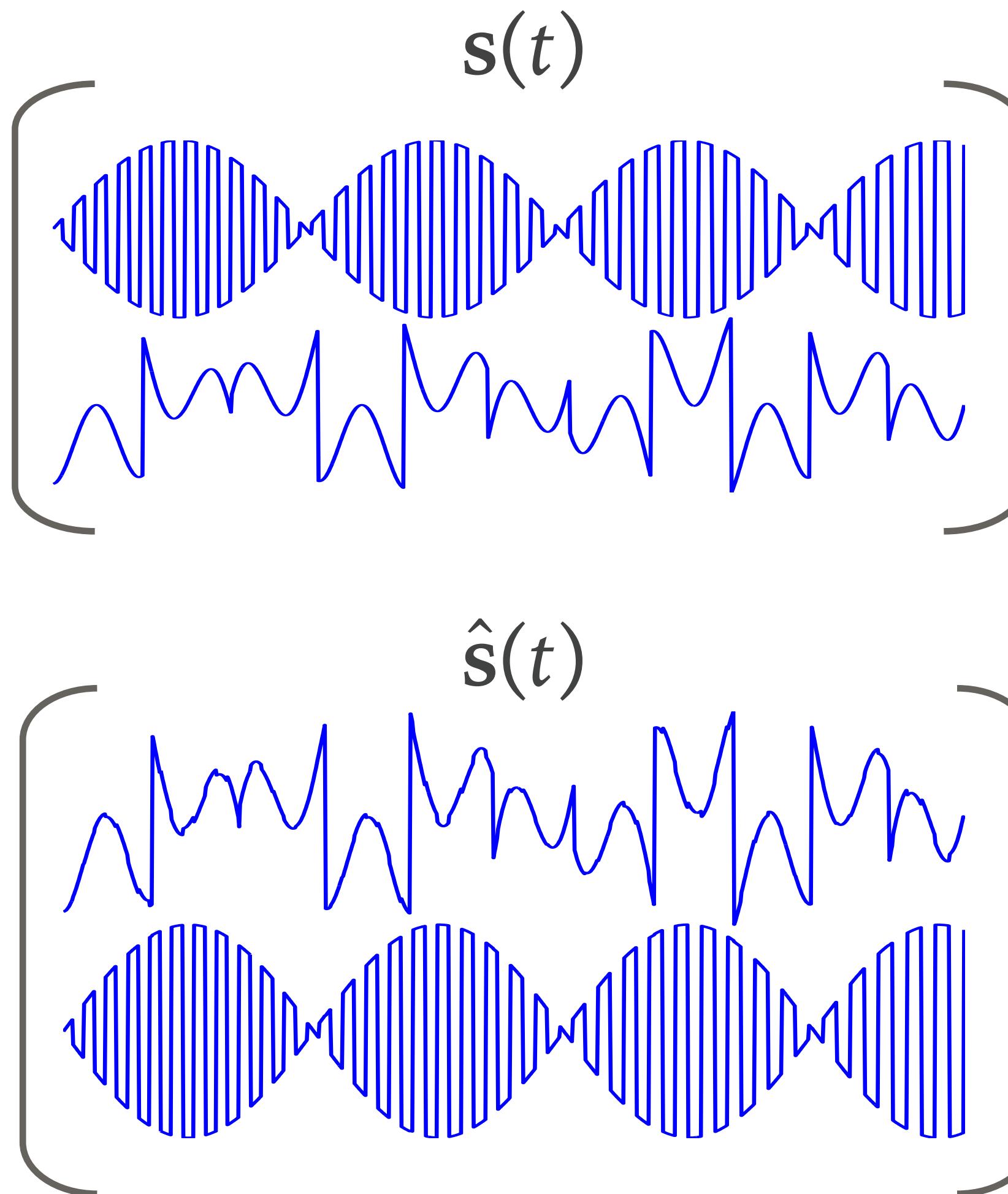
Trying this on our dataset

- We can now separate the mixture!



A couple of issues (not bad ones—yet)

- Permutation invariance
 - Order of outputs is random
- Scaling invariance
 - Scale is also random
- ICA will actually recover:
$$\hat{s}(t) = D \cdot P \cdot s(t)$$
 - Where **D** is diagonal and **P** is a permutation matrix

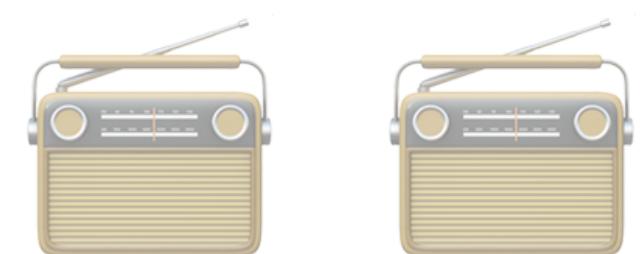
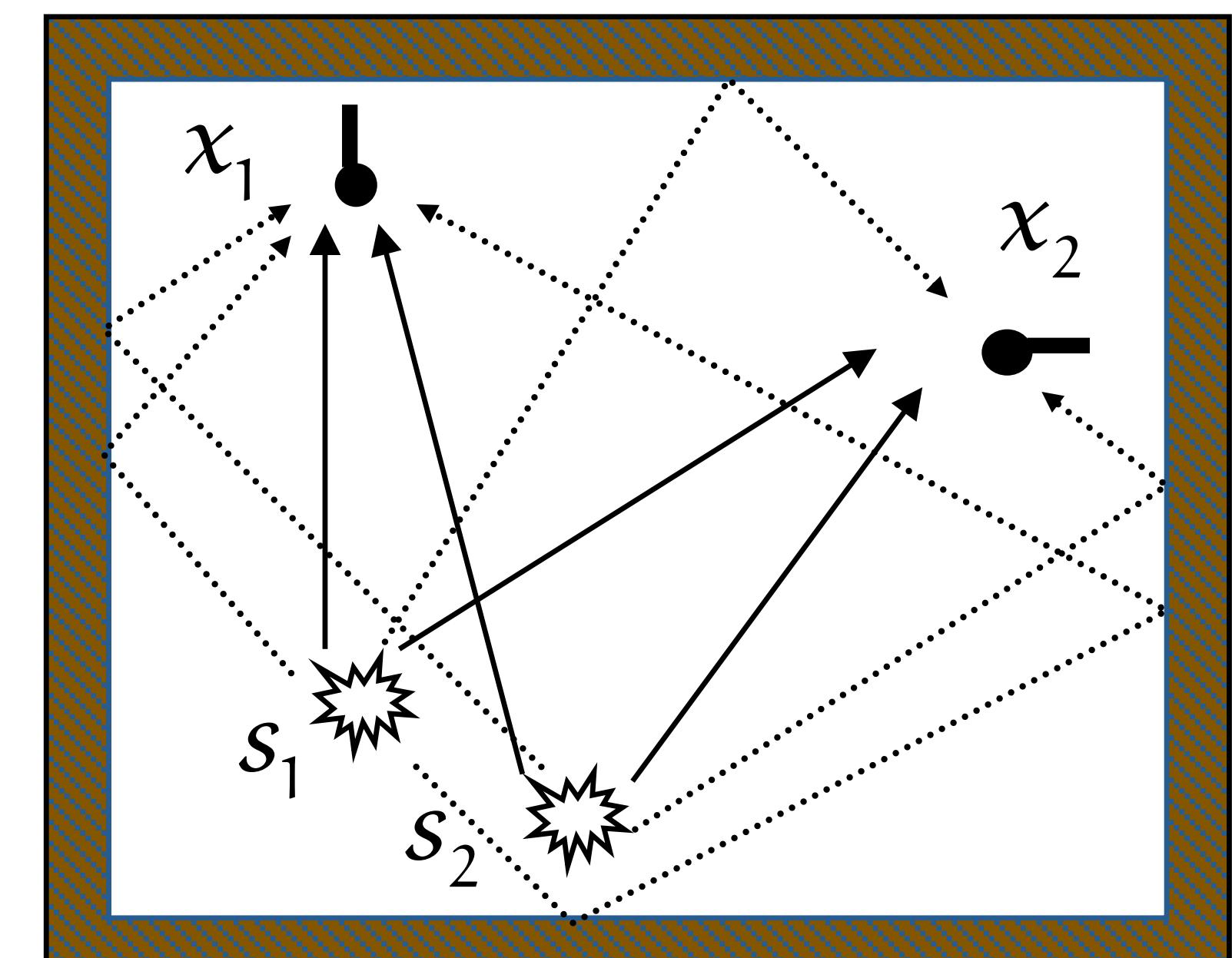


Widely applicable algorithm

- Also used heavily in bio signal analysis
 - Separating mother/fetal heartbeat
 - Finding “sources” in EEG recordings
- Popular in communications
 - Decouple mixed transmissions when using array antennas
 - Separate hyperspectral images from satellites
- Discover chemical makeups
 - E.g. find petroleum elements in crude mixtures

Instantaneous mixing limitations

- Sounds don't mix instantaneously
- There are multiple effects
 - Room reflections
 - Sensor response
 - Propagation delays
 - Propagation and reflection filtering
- Most can be seen as filters
- We must model a *convolutive* mixture

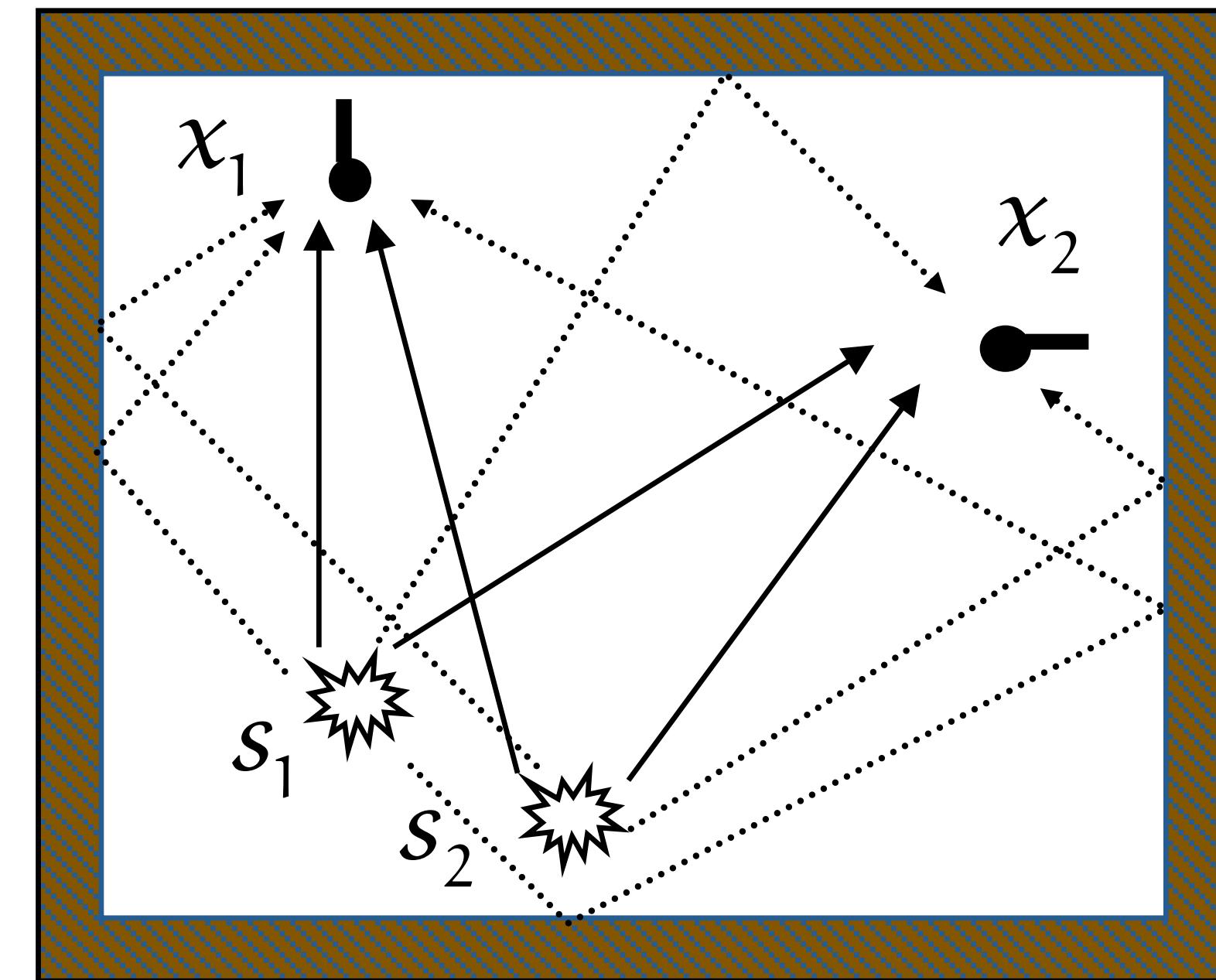


Convulsive mixing

- Instead of *instantaneous mixing*:

$$x_i(t) = \sum_{j=1} a_{ij} s_j(t)$$

- We now have *convulsive mixing*:
$$x_i(t) = \sum_j \sum_k a_{ij}(k) s_j(t - k)$$
- The mixing filters $a_{ij}(k)$ capture all the mixing effects in this model
- How do we do ICA now?



FIR matrix algebra

- Matrices with FIR filters as elements

$$\underline{\mathbf{A}} = \begin{bmatrix} \underline{a}_{11} & \underline{a}_{12} \\ \underline{a}_{21} & \underline{a}_{22} \end{bmatrix}$$

$$\underline{a}_{ij} = \begin{bmatrix} a_{ij}(0) & \dots & a_{ij}(k-1) \end{bmatrix}$$

- FIR matrix multiplication performs convolution and accumulation

$$\underline{\mathbf{A}} \cdot \underline{\mathbf{b}} = \begin{bmatrix} \underline{a}_{11} & \underline{a}_{12} \\ \underline{a}_{21} & \underline{a}_{22} \end{bmatrix} \cdot \begin{bmatrix} \underline{b}_1 \\ \underline{b}_2 \end{bmatrix} = \begin{bmatrix} \underline{a}_{11} * \underline{b}_1 + \underline{a}_{12} * \underline{b}_2 \\ \underline{a}_{21} * \underline{b}_1 + \underline{a}_{22} * \underline{b}_2 \end{bmatrix}$$

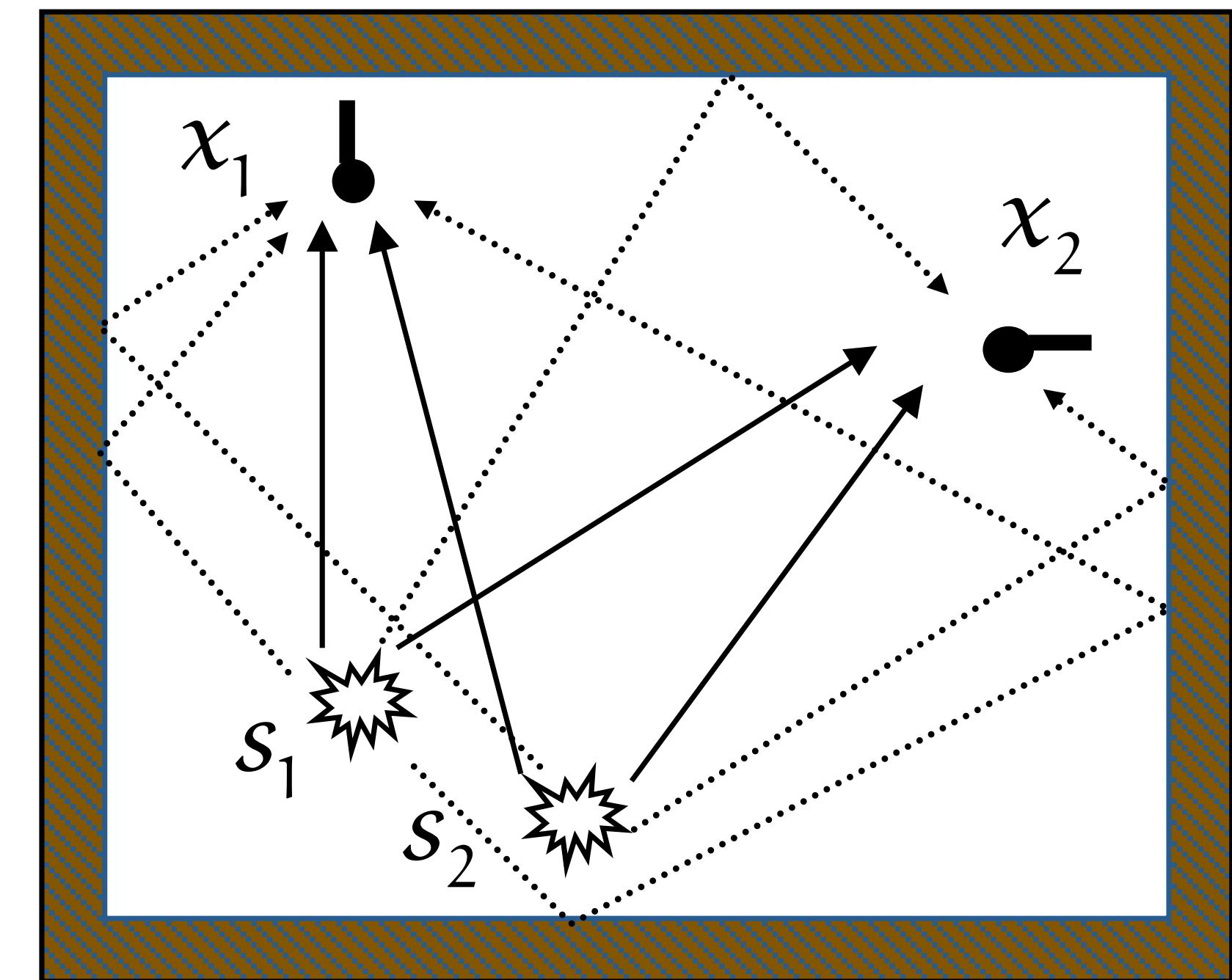
Back to convolutive mixing

- Rewrite convolutive mixing as:

$$x_i(t) = \sum_j \sum_k a_{ij}(k)s_j(t-k) \Rightarrow$$

$$\Rightarrow \underline{x}(t) = \underline{A} \cdot \underline{s}(t) = \begin{bmatrix} a_{11} * s_1(t) + a_{12} * s_2(t) \\ a_{21} * s_1(t) + a_{22} * s_2(t) \end{bmatrix}$$

- Tidier formulation!
- We can use the FIR matrix abstraction to solve this problem



The easy solution

- Straightforward translation of instantaneous learning rules using FIR matrices, e.g.:

$$\Delta \underline{\mathbf{W}} \propto \left(\mathbf{I} - f(\underline{\mathbf{W}} \cdot \underline{\mathbf{x}}) \cdot (\underline{\mathbf{W}} \cdot \underline{\mathbf{x}})^\top \right) \cdot \underline{\mathbf{W}}$$

- Not so easy with algebraic approaches
 - FIR tensors, FIR eigendecompositions, etc ...
- Multiple other (and more rigorous/better behaved) approaches have been developed

Complications with this approach

- Required convolutions are expensive
 - Real-room filters are long
 - FIR products can become very time consuming
- Convergence is hard to achieve
 - Huge parameter space
 - Tightly interwoven parameter relationships
- A slow optimization nightmare!

FIR matrix algebra, part II

- FIR matrices have frequency domain versions:

$$\underline{\mathbf{A}} = \begin{bmatrix} \underline{a}_{11} & \underline{a}_{12} \\ \underline{a}_{21} & \underline{a}_{22} \end{bmatrix} \xrightarrow{\text{frequency domain}} \hat{\underline{\mathbf{A}}} = \begin{bmatrix} \hat{\underline{a}}_{11} & \hat{\underline{a}}_{12} \\ \hat{\underline{a}}_{21} & \hat{\underline{a}}_{22} \end{bmatrix}$$

$$\hat{\underline{a}}_{ij} = \text{DFT} [\underline{a}_{ij}]$$

- And their products are simpler:

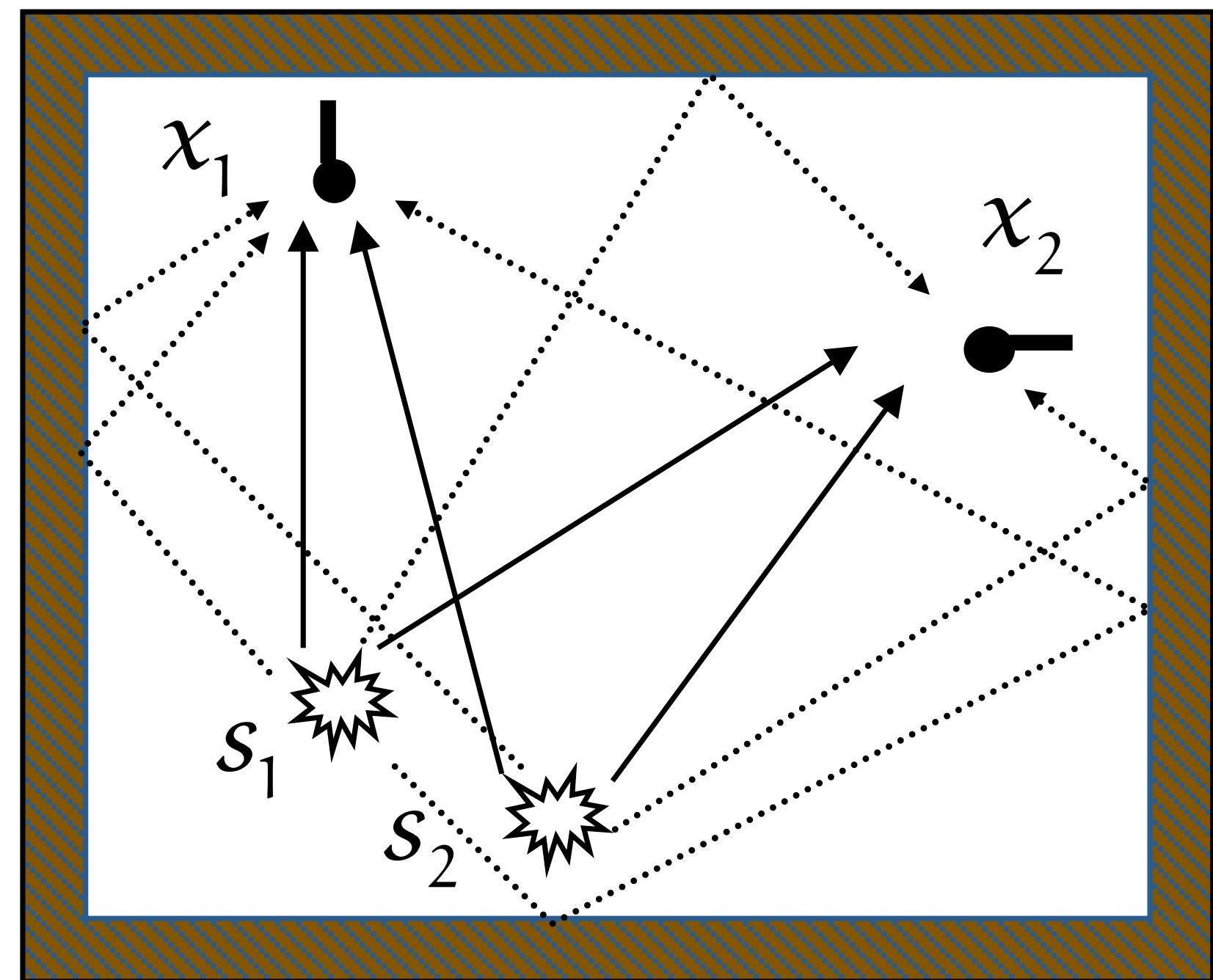
$$\begin{aligned}\hat{\underline{\mathbf{A}}} \cdot \hat{\underline{\mathbf{b}}} &= \begin{bmatrix} \hat{\underline{a}}_{11} \circ \hat{\underline{b}}_1 + \hat{\underline{a}}_{12} \circ \hat{\underline{b}}_2 \\ \hat{\underline{a}}_{21} \circ \hat{\underline{b}}_1 + \hat{\underline{a}}_{22} \circ \hat{\underline{b}}_2 \end{bmatrix} \\ \hat{\underline{a}} \circ \hat{\underline{b}} &= \begin{bmatrix} a(0) \cdot b(0) & \dots & a(k-1) \cdot b(k-1) \end{bmatrix}\end{aligned}$$

Yet another formulation

- We now model the mixing in the frequency domain instead:

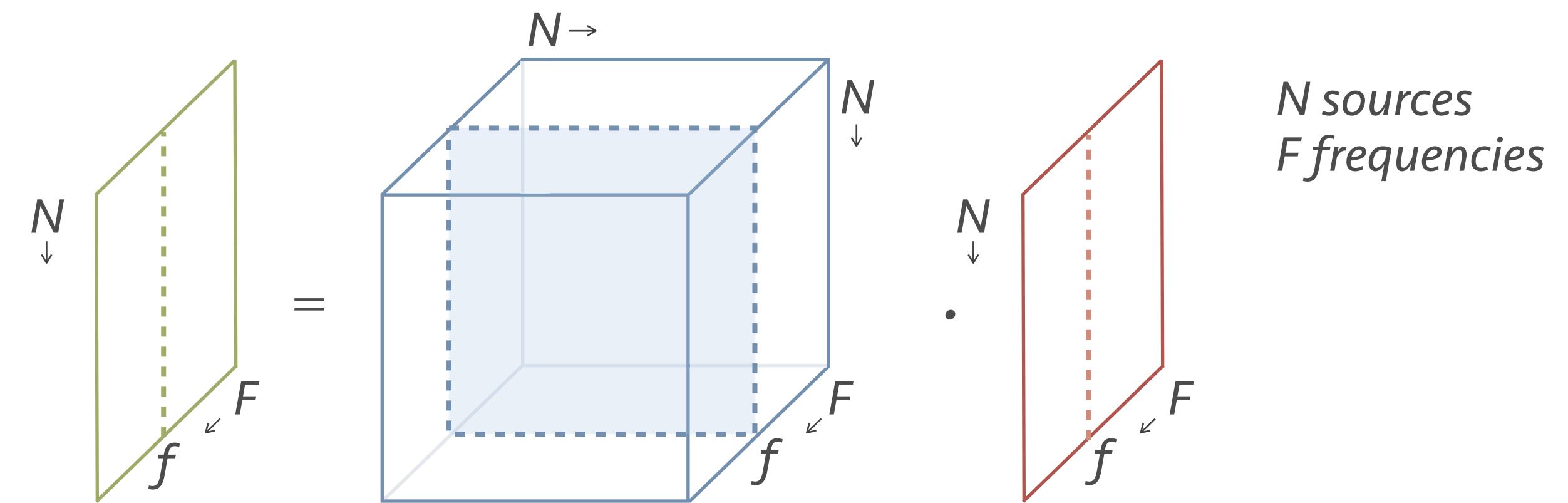
$$\hat{\underline{X}} = \hat{\underline{A}} \cdot \hat{\underline{S}}$$

- Compact form as before
 - Makes the notation manageable
 - But also has a hidden advantage!



Peeking a little deeper

- Looking at slices of the FIR matrices in $\hat{\underline{X}} = \hat{\underline{A}} \cdot \hat{\underline{S}}$

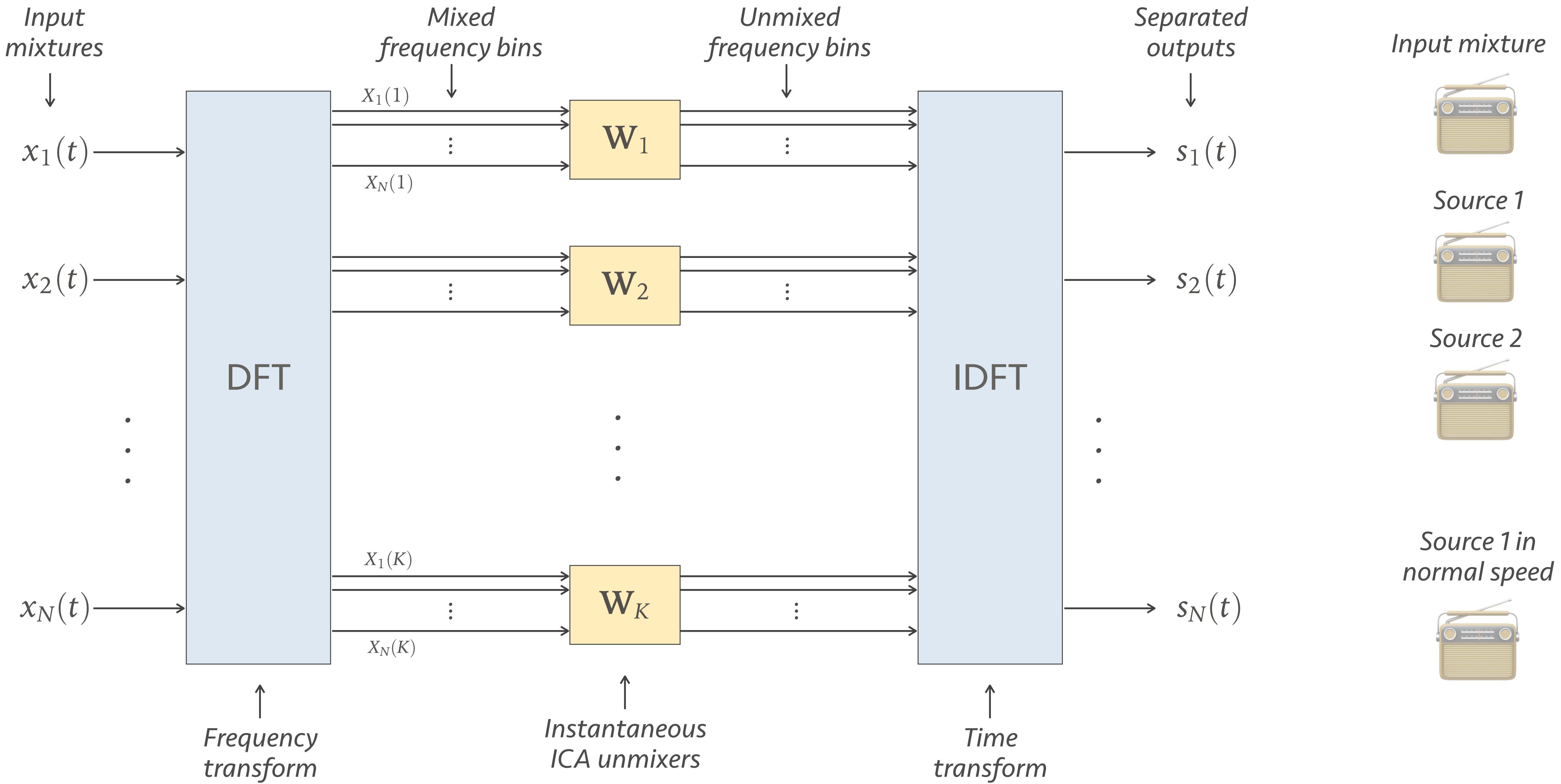


- Each slice is:

$$\hat{\underline{X}}^{(f)} = \begin{bmatrix} \underline{x}_1^{(f)} \\ \underline{x}_2^{(f)} \end{bmatrix} = \begin{bmatrix} a_{1,1}^{(f)} & a_{1,2}^{(f)} \\ a_{2,1}^{(f)} & a_{2,2}^{(f)} \end{bmatrix} \cdot \begin{bmatrix} \underline{s}_1^{(f)} \\ \underline{s}_2^{(f)} \end{bmatrix}$$

Hey, that's an instantaneous mixture!

Frequency domain ICA



Some complications ...

- **Permutation issues**

- We don't know which source will end up in each narrowband output ...
- Resulting output can have separated narrowband elements from both sounds!

Original source



Colored source



- **Scaling issues**

- Narrowband outputs can be scaled arbitrarily
- This results in spectrally colored outputs

Extracted source with permutations

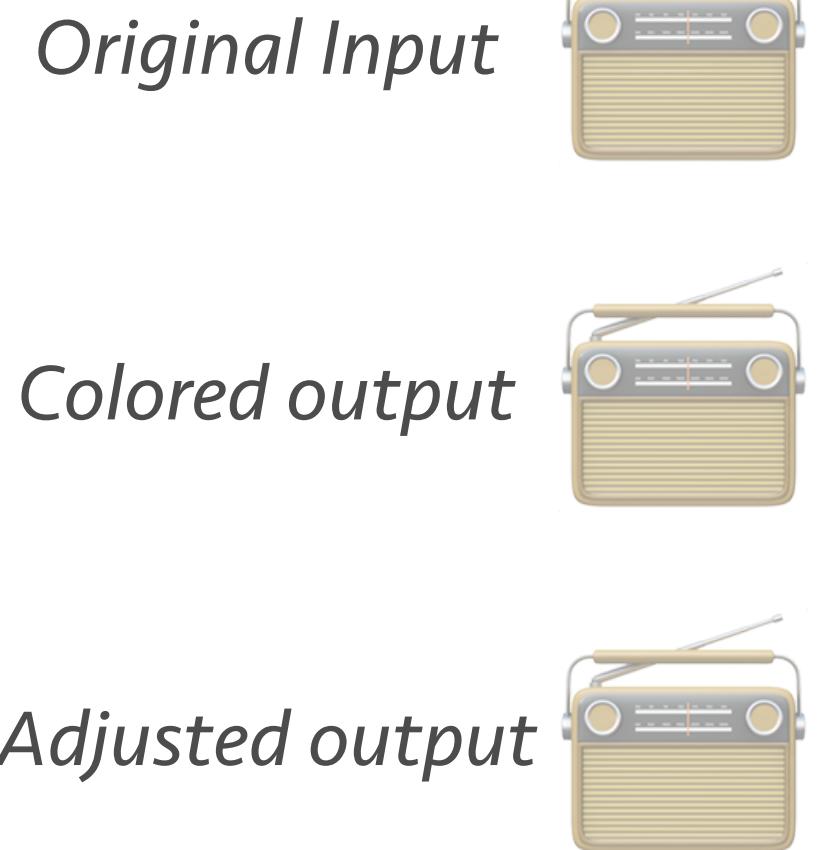


Scaling issue

- One simple fix is to normalize the separating matrices:

$$\mathbf{W}_f^{norm} = \mathbf{W}_f^{orig} \cdot \left| \mathbf{W}_f^{orig} \right|^{1/N}$$

- Results into more reasonable scaling
- More sophisticated approaches exist but this is not a major problem
- Some spectral coloration is unavoidable

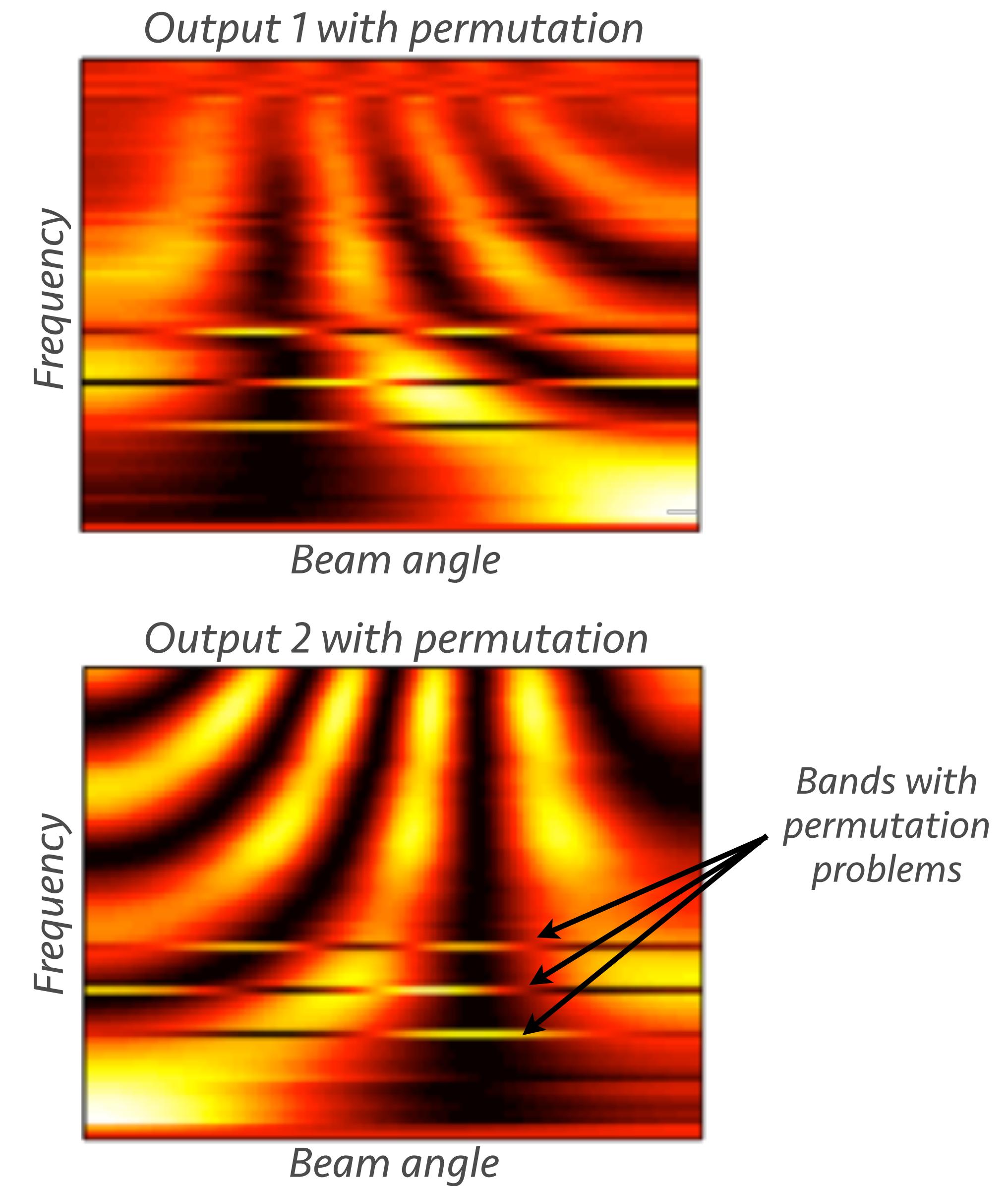


The permutation problem

- Continuity of unmixing matrices
 - Adjacent unmixing matrices tend to be similar
 - We can permute/bias them accordingly (doesn't work that great)
- Smoothness of spectral output
 - Narrowband components from each source tend to modulate the same way
 - Permute unmixing matrices to ensure adjacent narrowband output are similarly modulated (works ok)
- The above can fail miserably for more than two sources!
 - Combinatorial explosion!

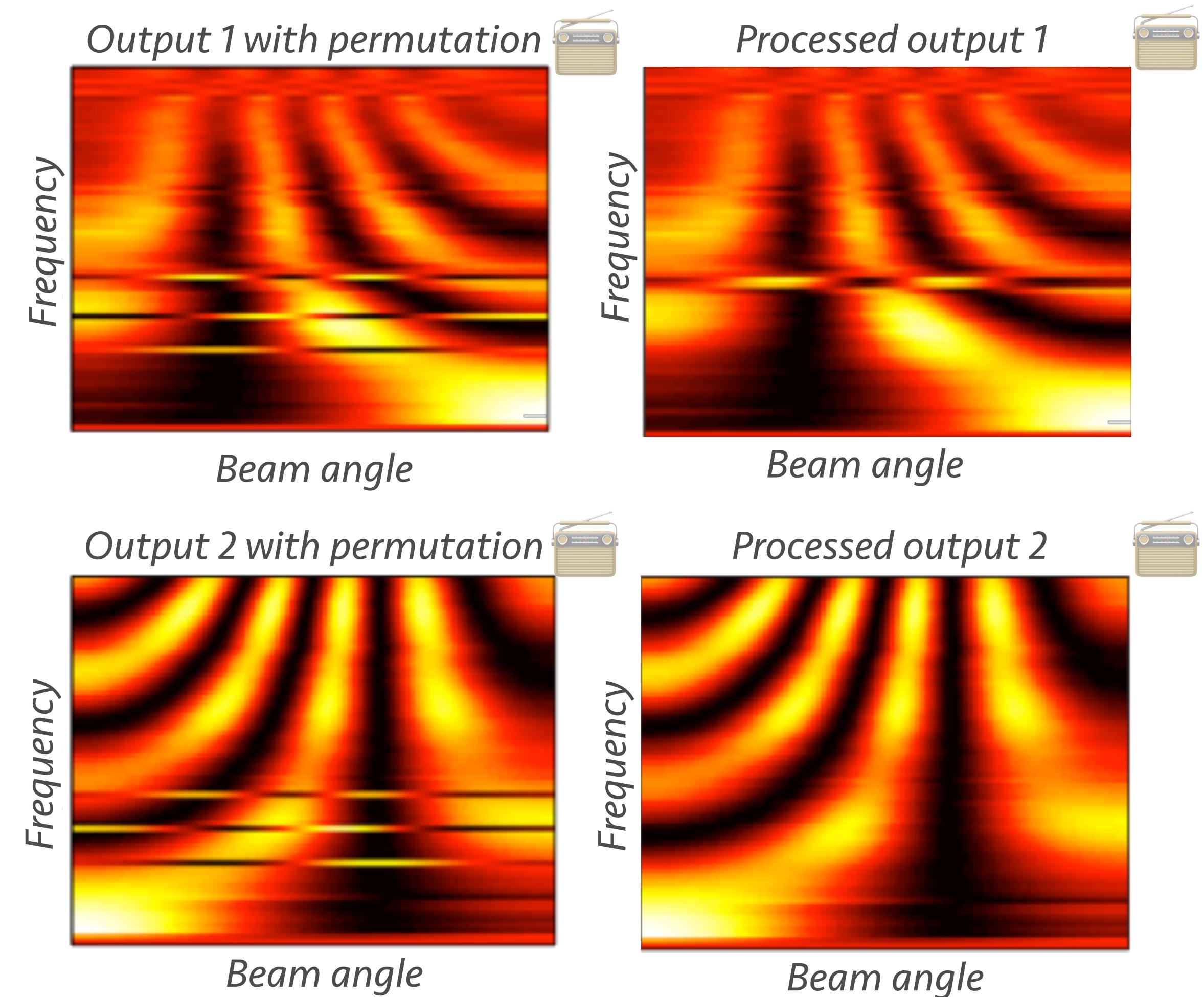
Beamforming and ICA

- If we know the placement of the sensors we can obtain the spatial response of the ICA solution
- ICA places nulls to cancel out interfering sources
 - Just as in the instantaneous case we cancel out sources
- We can visualize the permutation problem now
 - Seen as out-of-place bands



Beamforming to resolve permutations

- Spatial information can be used to resolve permutations
 - Find permutations that preserve zeros or smooth out the responses
- Works fine, although can be flaky if the array response is not clean



The N -input N -output problem

- ICA, in any formulation, inverts a square mixing matrix
 - Implies that we have the same number of input as outputs
 - E.g. in a street with 30 sources we need at least 30 mics!
- Solutions exist for M ins, N outs where $M > N$
- If $N > M$ we can only beamform
 - In some cases extra sources can be treated as noise
 - This can be very restrictive
 - But we'll see ways out of it in the next lecture

Recap

- Sensor arrays
- Beamforming and localization
- Array source separation
 - FIR Matrix algebra

Next lecture

- Underconstrained source separation
- Single-channel source separation

Reading

- Beamforming and localization
 - <http://onlinelibrary.wiley.com/book/10.1002/9780470994443>
(uiuc access only)
- Blind source separation and ICA
 - <http://paris.cs.illinois.edu/pubs/smaragdis-neurocomp.pdf>
 - <https://ieeexplore.ieee.org/abstract/document/1323089>