Chapter 5
Multiresolution Dynamic Mode Decomposition

Modeling of multiscale systems, in both space and time, pervades modern developments in theory and computation across the engineering, biological, and physical sciences. A significant challenge is making effective and efficient connections between microscale and macroscale effects, especially as they are potentially separated by orders of magnitude in space and time. Wavelet-based methods and/or windowed Fourier transforms are ideally structured to perform such multiresolution analyses (MRAs), as they systematically remove temporal or spatial features by a process of recursive refinement of sampling from the data [166, 76, 78]. Typically, MRA is performed in either space or time, but not both simultaneously. In this chapter, we integrate the concept of MRA in time with DMD [167]. This multiresolution DMD (mrDMD) is shown to naturally separate multiscale spatiotemporal features, providing an effective means to uncover multiscale structures in the data.

5.1 ▪ Time-frequency analysis and the Gábor transform

The Fourier transform is one of the most important and foundational methods for the analysis of time-series signals. However, it was realized quite early on that Fourier transform–based methods have severe limitations. Specifically, when transforming a given time signal, the entire frequency content of the signal can be captured with the transform, but the transform fails to capture the *moment in time* when various frequencies are actually exhibited. Indeed, by definition, the Fourier transform eliminates all time-domain information by integrating over all time.

The limitations of the direct application of the Fourier transform, and its inability to localize a signal in both the time and the frequency domains, were articulated in the early development of radar and sonar detection. The Hungarian physicist/mathematician/electrical engineer Gábor Dénes (Physics Nobel Prize in 1971 for the discovery of holography in 1947) was the first to propose a formal method for localizing both time and frequency. His method involved a simple modification of the Fourier transform kernel. Gábor introduced the kernel

$$g_{t,\omega}(\tau) = e^{i\omega\tau} g(\tau - t), \tag{5.1}$$

where the new term to the Fourier kernel $g(\tau - t)$ was introduced with the aim of localizing both time and frequency. The *Gábor transform*, also known as the *short-*

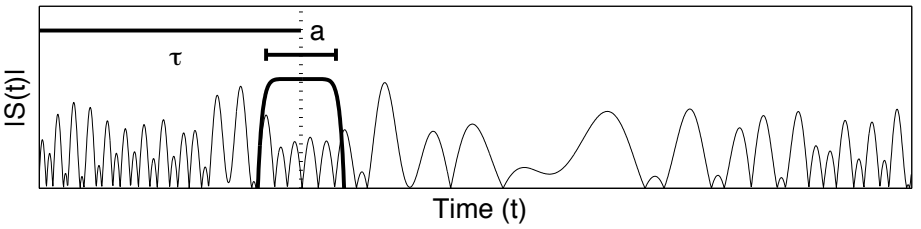

Figure 5.1. *Graphical depiction of the Gábor transform for extracting the time-frequency content of a signal $S(t)$. The time-filtering window $g(\tau - t)$ is centered at τ with width a.*

time Fourier transform (STFT) is then defined as

$$\mathcal{G}[f](t,\omega) = \tilde{f}_g(t,\omega) = \int_{-\infty}^{\infty} f(\tau)\bar{g}(\tau - t)e^{-i\omega\tau}d\tau = \left(f, \bar{g}_{t,\omega}\right), \tag{5.2}$$

where the bar denotes the complex conjugate of the function. Thus, the function $g(\tau - t)$ acts as a time filter for localizing the signal over a specific window of time. The integration over the parameter τ slides the time-filtering window down the entire signal, picking out the frequency information at each instant of time. Figure 5.1 illustrates the Gábor time-filtering scheme. In this figure, the time-filtering window is centered at τ with width a. Thus, the frequency content of a window of time is extracted and τ is modified to extract the frequencies of another window. The definition of the Gábor transform captures the entire time-frequency content of the signal; the Gábor transform is a function of the two variables t and ω.

Although the Gábor transform gives a method whereby time and frequency can be simultaneously characterized, there are obvious limitations to the method. Specifically, the method is limited by the time filtering itself. Consider the illustration of the method in Figure 5.1. The time window filters out the time behavior of the signal in a window centered at τ with width a. It follows that, when considering the spectral content of this window, any portion of the signal with a wavelength longer than the window is completely lost. Indeed, since the Heisenberg uncertainty relationship must hold, the shorter the time-filtering window, the less information there is concerning the frequency content. In contrast, longer windows retain more frequency components, but this comes at the expense of losing the time resolution of the signal. As a consequence of a fixed time-filtering window, there are trade-offs between time and frequency resolution; increased accuracy in one of these parameters comes at the expense of resolution in the other parameter.

5.2 ▪ Wavelets and MRA

The Gábor transform established two key principles for time-frequency analysis: *translation* of a short-time window and *scaling* of the short-time window to capture finer

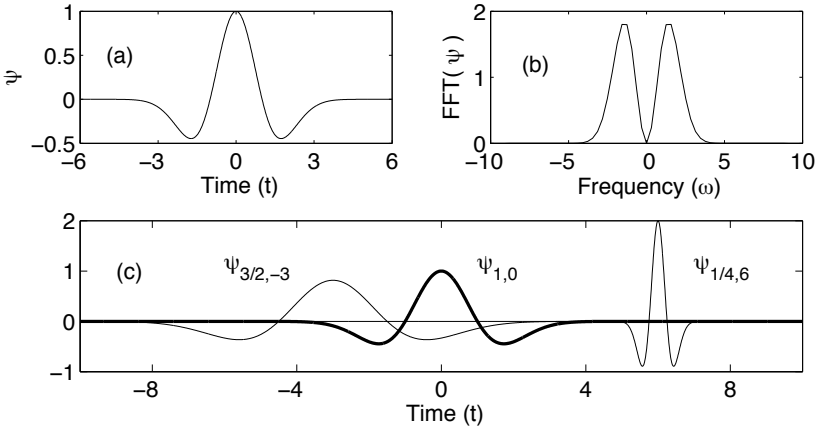

Figure 5.2. *Illustration of the Mexican hat wavelet $\psi_{1,0}$ (top left panel), its Fourier transform $\hat{\psi}_{1,0}$ (top right panel), and two additional dilations and translations of the basic $\psi_{1,0}$ wavelet, namely the $\psi_{3/2,-3}$ and $\psi_{1/4,6}$ (bottom panel).*

time resolution. A simple modification to the Gábor method is to allow the scaling window (a) to vary and successively extract improvements in the time resolution. In other words, first the low-frequency (poor time resolution) components are extracted using a broad scaling window. The scaling window is subsequently shortened to extract higher frequencies at better time resolution. By keeping a catalog of the extracting process, excellent resolution in both time and frequency of a given signal can be obtained. This principle is fundamental to *wavelet theory*. The term *wavelet* means little wave and originates from the processes whereby the scaling window extracts smaller and smaller pieces of waves from the larger signal.

Wavelet analysis begins with the consideration of a function known as the *mother wavelet*:

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right), \tag{5.3}$$

where $a \neq 0$ and b are real constants. The effect of these two parameters on the shape of the wavelet is illustrated in Figure 5.2; the a parameter controls scaling and b denotes translation (previously denoted by τ in Figure 5.1). Unlike Fourier analysis, and very much like Gábor transforms, a vast variety of mother wavelets can be constructed. In principle, the mother wavelet is designed to have certain properties that are somehow beneficial for a given problem. Depending on the application, different mother wavelets may be selected. Ultimately, the wavelet is simply another expansion basis for representing a given signal or function. Historically, the first wavelet was constructed by Haar in 1910 [124], so the concepts and ideas of wavelets are over a

century old. However, their use was not widespread until the mid-1980s.

The wavelet basis can be accessed via an integral transform of the form

$$(Tf)(\omega) = \int_t K(t,\omega) f(t) dt, \tag{5.4}$$

where $K(t,\omega)$ is the kernel of the transform. This is equivalent in principle to the Fourier transform, whose kernel consists of the oscillations given by $K(t,\omega) = \exp(-i\omega t)$. The key idea now is to define a transform that incorporates the mother wavelet as the kernel. We may define the *continuous wavelet transform* (CWT):

$$\mathcal{W}_\psi[f](a,b) = (f, \psi_{a,b}) = \int_{-\infty}^{\infty} f(t)\bar{\psi}_{a,b}(t) dt. \tag{5.5}$$

Much like the windowed Fourier transform, the CWT is a function of the dilation parameter a and translation parameter b. Parenthetically, a wavelet is admissible if the following property holds:

$$C_\psi = \int_{-\infty}^{\infty} \frac{|\hat{\psi}(\omega)|^2}{|\omega|} d\omega < \infty, \tag{5.6}$$

where the Fourier transform of the wavelet is defined by

$$\hat{\psi}_{a,b} = \frac{1}{\sqrt{|a|}} \int_{-\infty}^{\infty} e^{-i\omega t} \psi\left(\frac{t-b}{a}\right) dt = \frac{1}{\sqrt{|a|}} e^{-ib\omega} \hat{\psi}(a\omega). \tag{5.7}$$

Thus, provided the admissibility condition (5.6) is satisfied, the wavelet transform can be well defined.

In the wavelet transform, the signal is first split up into a collection of smaller signals by translating the wavelet with the parameter b over the entire time domain of the signal. Second, the same signal is processed at different frequency bands, or resolutions, by scaling the wavelet window with the parameter a. The combination of translation and scaling allows for processing of the signals at different times and frequencies. Figure 5.3 is a schematic that illustrates the Gábor and wavelet transform concepts in the time-frequency domain. In this figure, the standard time series, Fourier transform, and windowed Fourier transform are represented along with the multiresolution concept of the wavelet transform. In particular, the box illustrating the wavelet transform shows the multiresolution concept in action. Starting with a large Fourier domain window, the entire frequency content is extracted. The time window is then scaled in half, leading to finer time resolution at the expense of worse frequency resolution. This process is continued until a desired time-frequency resolution is obtained. This simple figure is critical for understanding wavelet application to time-frequency analysis.

As an example, consider one of the more common wavelets: the Mexican hat wavelet (Figure 5.2). This wavelet is essentially a second moment of a Gaussian in the frequency domain. The definitions of this wavelet and its transform are as follows:

$$\psi(t) = (1-t^2)e^{-t^2/2} = -\frac{d^2}{dt^2}\left(e^{-t^2/2}\right) = \psi_{1,0}, \tag{5.8a}$$

$$\hat{\psi}(\omega) = \hat{\psi}_{1,0}(\omega) = \sqrt{2\pi}\omega^2 e^{-\omega^2/2}. \tag{5.8b}$$

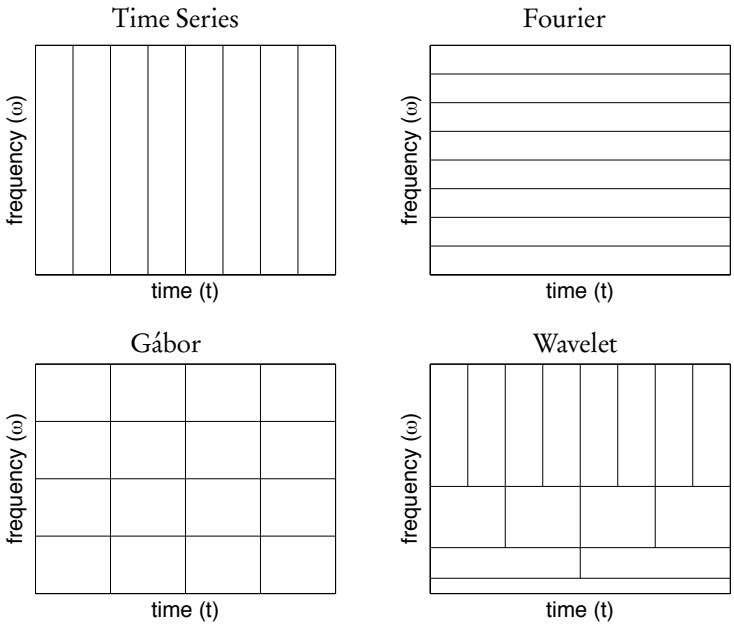

Figure 5.3. *Graphical depiction of the difference between the time-series analysis, Fourier analysis, Gábor analysis, and wavelet analysis of a signal. The wavelet transform starts with a large Fourier domain window so that the entire frequency content is extracted. The time window is then scaled in half, leading to finer time resolution at the expense of worse frequency resolution. This process is continued until a desired time-frequency resolution is obtained.*

The Mexican hat wavelet has excellent localization properties in both time and frequency due to the minimal time-bandwidth product of the Gaussian function. Figure 5.2 (top panels) shows the basic Mexican wavelet function $\psi_{1,0}$ and its Fourier transform, both of which decay in t (ω) like $\exp(-t^2)$ ($\exp(-\omega^2)$). The Mexican hat wavelet can be dilated and translated easily, as is depicted in Figure 5.2 (bottom panel). Here three wavelets are depicted: $\psi_{1,0}$, $\psi_{3/2,-3}$, and $\psi_{1/4,6}$, showing both scaling and translation of the wavelet.

5.3 ▪ Formulating mrDMD

The mrDMD model is inspired by the observation that the slow and fast modes can be separated for such applications as foreground/background subtraction in video streams, as discussed in Chapter 4. The mrDMD approach is a recursive computation of DMD to remove low-frequency, or slowly varying, features from a given collection of snapshots. This process is illustrated in Figure 5.4. In DMD, the number of snapshots m is chosen so that DMD modes provide an approximately full-rank approximation of the dynamics observed. Thus, m is chosen so that all high- and low-frequency content is present. In mrDMD, m is chosen initially to allow extraction of the lowest-frequency modes. At each resolution level, the slowest modes are removed, the time domain is divided into two segments with $m/2$ snapshots each, and DMD is once again performed

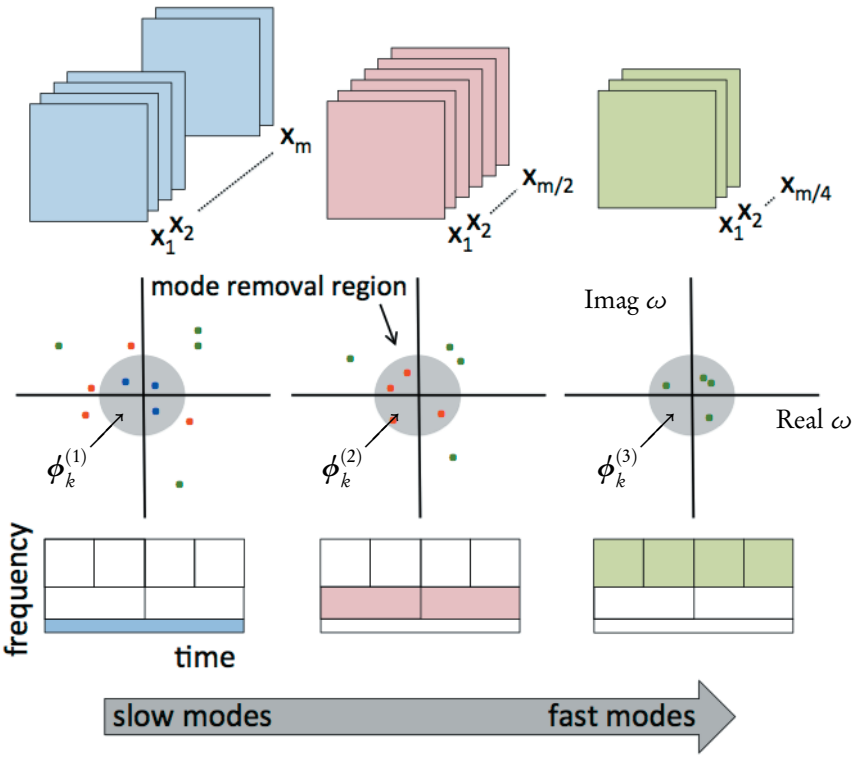

Figure 5.4. *The mrDMD approach takes successive samples of the data, initially with m snapshots and decreasing by a factor of two at each resolution level. The DMD spectrum is shown in the middle panel, where there are m_1 (blue dots) slow-dynamic modes at the slowest level, m_2 (red) modes at the next level, and m_3 (green) modes at the fastest time scale shown. The shaded region represents the modes that are removed at that level. The bottom panel shows the wavelet-like time-frequency decomposition of the data color-coded with the snapshots and DMD spectral representations* [167].

on each $m/2$ snapshot sequence. This recursive process continues until a desired termination. Because we are only interested in the lowest-frequency modes at each level, it is possible to subsample the snapshots, reducing m and increasing the computational efficiency.

Mathematically, mrDMD separates the DMD approximate solution (1.24) at the

first level as follows:

$$\begin{aligned}\mathbf{x}_{\text{mrDMD}}(t) &= \sum_{k=1}^{m} b_k(0)\boldsymbol{\phi}_k^{(1)}\exp(\omega_k t) \\ &= \underbrace{\sum_{k=1}^{m_1} b_k(0)\boldsymbol{\phi}_k^{(1)}\exp(\omega_k t)}_{\text{(slow modes)}} + \underbrace{\sum_{k=m_1+1}^{m} b_k(0)\boldsymbol{\phi}_k^{(1)}\exp(\omega_k t)}_{\text{(fast modes)}},\end{aligned} \tag{5.9}$$

where the $\boldsymbol{\phi}_k^{(1)}$ represent the DMD modes computed from the full m snapshots.

The first sum in expression (5.9) represents the slow-mode dynamics, whereas the second sum is everything else. Thus, the second sum can be computed to yield the matrix

$$\mathbf{X}_{m/2} = \sum_{k=m_1+1}^{m} b_k(0)\boldsymbol{\phi}_k^{(1)}\exp(\omega_k t). \tag{5.10}$$

The DMD analysis outlined in the previous section can now be performed once again on the data matrix $\mathbf{X}_{m/2}$. However, the matrix $\mathbf{X}_{m/2}$ is now separated into two matrices,

$$\mathbf{X}_{m/2} = \mathbf{X}_{m/2}^{(1)} + \mathbf{X}_{m/2}^{(2)}, \tag{5.11}$$

where the first matrix contains the first $m/2$ snapshots and the second matrix contains the remaining $m/2$ snapshots. The m_2 slow-DMD modes at this level are given by $\boldsymbol{\phi}_k^{(2)}$, where they are computed separately in the first or second interval of snapshots.

The mrDMD process works by recursively removing slow-frequency components and building the new matrices $\mathbf{X}_{m/2}, \mathbf{X}_{m/4}, \mathbf{X}_{m/8}, \ldots$ until a desired/prescribed multiresolution decomposition has been achieved. The approximate DMD solution can then be constructed as follows:

$$\begin{aligned}\mathbf{x}_{\text{mrDMD}}(t) = &\sum_{k=1}^{m_1} b_k^{(1)}\boldsymbol{\phi}_k^{(1)}\exp(\omega_k^{(1)}t) + \sum_{k=1}^{m_2} b_k^{(2)}\boldsymbol{\phi}_k^{(2)}\exp(\omega_k^{(2)}t) \\ &+ \sum_{k=1}^{m_3} b_k^{(3)}\boldsymbol{\phi}_k^{(3)}\exp(\omega_k^{(3)}t) + \cdots,\end{aligned} \tag{5.12}$$

where at the evaluation time t, the correct modes from the sampling window are selected at each level of the decomposition. Specifically, $\boldsymbol{\phi}_k^{(\ell)}$ and $\omega_k^{(\ell)}$ are the DMD modes and DMD eigenvalues at the ℓth level of decomposition, the $b_k^{(\ell)}$ are the initial projections of the data onto the time interval of interest, and the m_k are the number of slow modes retained at each level. The advantage of this method is apparent: different spatiotemporal DMD modes are used to represent key multiresolution features. There is not a single set of modes that dominates the SVD and potentially obscures features at other time scales.

Figure 5.4 illustrates the mrDMD process schematically. In the figure, a three-level decomposition is performed, with the slowest scale represented in blue (eigenvalues and snapshots), the midscale in red, and the fast scale in green. The connection to multiresolution wavelet analysis is also evident from the bottom panels, as one can see that the mrDMD method successively pulls out time-frequency information in a principled way.

The sampling strategy and algorithm can be further optimized since only the slow modes at each decomposition level need to be accurately computed. Thus, one can modify the algorithm so the sampling rate increases as the decomposition proceeds from one level to the next. This assures that the lowest levels of the mrDMD are not highly sampled, since the cost of the SVD would be greatly increased by such a fine sampling rate.

5.3.1 ▪ Formal mrDMD expansion

The solution (5.12) can be made more precise. Specifically, one must account for the number of levels (L) of the decomposition, the number of time bins (J) for each level, and the number of modes retained at each level (m_L). This can be easily seen in Figure 5.4. Thus, the solution is parameterized by the following three indices:

$$\ell = 1,2,\ldots,L \text{ : number of decomposition levels,} \tag{5.13a}$$

$$j = 1,2,\ldots,J \text{ : number time bins per level } (J = 2^{(\ell-1)}), \tag{5.13b}$$

$$k = 1,2,\ldots,m_\ell \text{ : number of modes extracted at level } L. \tag{5.13c}$$

To formally define the series solution for $\mathbf{x}_{\text{mrDMD}}(t)$, we define the indicator function

$$f^{\ell,j}(t) = \begin{cases} 1, & t \in [t_j, t_{j+1}] \\ 0, & \text{elsewhere} \end{cases}, \quad \text{with} \quad j = 1,2,\ldots,J \quad \text{and} \quad J = 2^{(\ell-1)}, \tag{5.14}$$

which is only nonzero in the interval, or time bin, associated with the value of j.

The three indices and indicator function (5.14) give the mrDMD solution expansion

$$\mathbf{x}_{\text{mrDMD}}(t) = \sum_{\ell=1}^{L}\sum_{j=1}^{J}\sum_{k=1}^{m_L} f^{\ell,j}(t) b_k^{(\ell,j)} \boldsymbol{\phi}_k^{(\ell,j)} \exp(\omega_k^{(\ell,j)} t). \tag{5.15}$$

This concise definition of the mrDMD solution includes the information on the level, time bin location, and number of modes extracted. Figure 5.5 demonstrates mrDMD in terms of the solution (5.15). In particular, each mode is represented in its respective time bin and level. An alternative interpretation of this solution is that it yields the least-square fit, at each level ℓ of the decomposition, to the discrete-time linear dynamical system

$$\mathbf{x}_{t+1}^{(\ell,j)} = \mathbf{A}^{(\ell,j)} \mathbf{x}_t^{(\ell,j)}, \tag{5.16}$$

where the matrix $\mathbf{A}^{(\ell,j)}$ captures the dynamics in a given time bin j at level ℓ.

The indicator function $f^{\ell,j}(t)$ acts as a sifting function for each time bin. Interestingly, this function acts as the Gábor window of a windowed Fourier transform [166].

Since our sampling bin has a hard cutoff of the time series, it may introduce some artificial high-frequency oscillations. Time-series analysis, and wavelets in particular, introduce various functional forms that can be used in an advantageous way. Thinking more broadly, one can imagine using wavelet functions for the sifting operation, thus allowing the time function $f^{\ell,j}(t)$ to take the form of one of the many potential wavelet bases, e.g., Haar, Daubechies, Mexican hat, etc. [76, 78]. For the present, we simply use the sifting function introduced in (5.14).

5.4 ▪ The mrDMD algorithm

The mrDMD is a recursive, hierarchical application of the basic DMD algorithm introduced in Chapter 1. However, the sampling windows (snapshot collection) are now

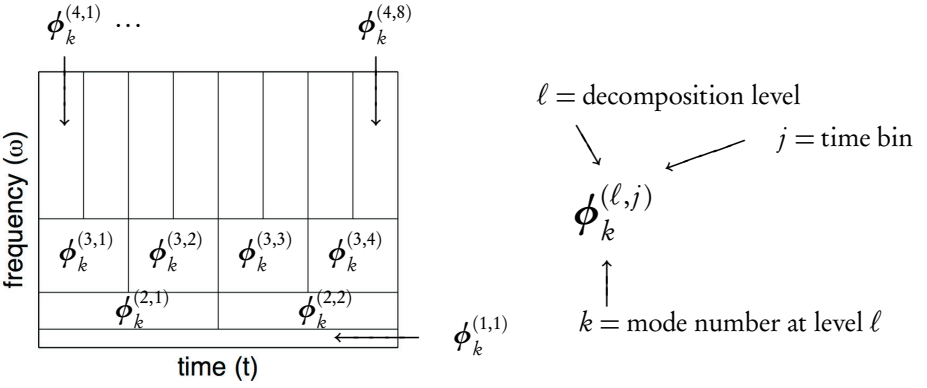

Figure 5.5. *Illustration of the mrDMD hierarchy. Represented are the modes $\phi_k^{\ell,j}$ and their position in the decomposition structure. The triplet of integer values ℓ, j, and k uniquely expresses the time level, bin, and mode of the decomposition.*

adjusted and only the slowest modes are extracted at each level of the decomposition. We highlight again the DMD algorithm but now in the context of the mrDMD innovation. As before, when the state dimension n is large, the matrix $\mathbf{A}$ may be intractable to analyze directly. Instead, DMD circumvents the eigendecomposition of $\mathbf{A}$ by considering a rank-reduced representation in terms of a POD-projected matrix $\tilde{\mathbf{A}}$. The mrDMD algorithm proceeds as follows:

1. Consider a sampling window in time at level ℓ and time bin j of the decomposition. In this sampling window, construct the data matrices $\mathbf{X}$ and $\mathbf{X}'$.
2. Take the SVD of $\mathbf{X}$:
$$\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^*, \tag{5.17}$$
where $*$ denotes the conjugate transpose, $\mathbf{U} \in \mathbb{C}^{n\times r}$, $\Sigma \in \mathbb{C}^{r\times r}$, and $\mathbf{V} \in \mathbb{C}^{m\times r}$. Here r is the rank of the reduced SVD approximation to $\mathbf{X}$. The left singular vectors $\mathbf{U}$ are POD modes.
3. Next, compute $\tilde{\mathbf{A}}^{(\ell,j)}$, the $r \times r$ projection of the full matrix $\mathbf{A}^{(\ell,j)}$ onto POD modes:
$$\begin{aligned} &\mathbf{A}^{(\ell,j)} = \mathbf{X}'\mathbf{V}\Sigma^{-1}\mathbf{U}^* \\ \Longrightarrow \quad &\tilde{\mathbf{A}}^{(\ell,j)} = \mathbf{U}^*\mathbf{A}^{(\ell,j)}\mathbf{U} = \mathbf{U}^*\mathbf{X}'\mathbf{V}\Sigma^{-1}. \end{aligned} \tag{5.18}$$
4. Compute the eigendecomposition of $\tilde{\mathbf{A}}^{(\ell,j)}$:
$$\tilde{\mathbf{A}}^{(\ell,j)}\mathbf{W} = \mathbf{W}\Lambda, \tag{5.19}$$

where the columns of $\mathbf{W}$ are eigenvectors and $\mathbf{\Lambda}$ is a diagonal matrix containing the corresponding eigenvalues λ_k.

5. Identify and collect the slow eigenvalues, if any, where $||\lambda_j|| < \rho$, and use them to construct the slow modes at this level of decomposition.

6. Finally, reconstruct the eigendecomposition of $\mathbf{A}^{(\ell,j)}$ from $\mathbf{W}$ and $\mathbf{\Lambda}$. In particular, the eigenvalues of $\mathbf{A}^{(\ell,j)}$ are given by $\mathbf{\Lambda}$ and the eigenvectors of $\mathbf{A}^{(\ell,j)}$ (DMD modes) are given by the columns of $\mathbf{\Phi}^{(\ell,j)}$:

$$\mathbf{\Phi}^{(\ell,j)} = \mathbf{X}'\mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{W}. \tag{5.20}$$

The power at level ℓ and time bin j may now be computed, following the approach described in Chapter 8.

7. Divide the original sampling window in half, and repeat all of the above steps for each of the first and second halves of the sampling window at level $\ell + 1$.

5.4.1 ▪ Parameters of the mrDMD algorithm

This algorithm has a few parameters. In step 1, at each level ℓ, the number of snapshots in each sampling window may be a subsample of the full-resolution data based on the desired slow eigenvalue threshold in step 5. This subsampling enables a substantial increase in efficiency of execution, especially at lower levels of recursion with larger windows, without sacrificing the ability to extract the lowest-amplitude (i.e., slowest) modes. In step 2, the rank r of the reduced SVD approximation may be chosen to exploit low-rank truncation of the data. As noted in Chapters 1 and 8, this r may be chosen in a principled way.

In step 5, the parameter ρ is selected to extract only slow modes. There is some freedom in the choice of ρ, and this value may be selected based on the duration of the sampling window. To be specific, the implementation in the example code in Algorithm **5.3** below chooses ρ to select modes with fewer than two cycles of oscillations within the sampling window.

Note that the above algorithm also allows construction of the best-fit dynamical system at each level (5.16). After decomposing at a specified number of levels, the solution may be reconstructed using (5.15).

5.5 ▪ Example code and decomposition

The first example we consider for application of mrDMD is illustrated in Figure 5.6. This example consists of a synthetic movie of $m = 1000$ frames where each frame has $n = 80 \times 80 = 6400$ pixels, sampled at $dt = 0.01$ sec. The three underlying modes of the data have different, overlapping spatial distributions. Mode 1 oscillates at 5.55 Hz for the first half of the movie. Mode 2 oscillates at 0.9 Hz from seconds 3 to 7 of the movie. Mode 3 has a slow, stationary oscillation of 0.15 Hz for the entire duration of the movie. For this demonstration, Gaussian white sensor noise was added to every pixel so that the signal-to-noise ratio is approximately 3. The following section of code constructs an instance of this example data.

ALGORITHM 5.1. Construct example movie (**mrDMD_demo.m**).

```
% parameters
nx = 80; % horizontal pixels
ny = 80; % vertical pixels
n  = nx * ny;
T = 10; % sec
dt = 0.01;
t = dt:dt:T;
sig = 0.1; % magnitude of gaussian noise added

% 3 underlying modes
[Xgrid, Ygrid] = meshgrid(1:nx, 1:ny);

mode1.u = exp(-((Xgrid-40).^2/250+(Ygrid-40).^2/250));
mode1.f = 5.55; % cycles/sec
mode1.A = 1;
mode1.lambda = exp(1i * mode1.f*2*pi*dt);
mode1.range = [0, 5];

mode2.u = zeros(nx, ny);
mode2.u(nx-40:nx-10, ny-40:ny-10) = 1;
mode2.f = 0.9; % cycles/sec
mode2.A = 1;
mode2.lambda = exp(1i * mode2.f*2*pi*dt);
mode2.range = [3, 7];

mode3.u = zeros(nx, ny);
mode3.u(1:nx-20, 1:ny-20) = 1;
mode3.f = 0.15; % cycles/sec
mode3.A = 0.5;
mode3.lambda = exp(1i * mode3.f*2*pi*dt);
mode3.range = [0, T];

modes(1) = mode1;
modes(2) = mode2;
modes(3) = mode3;

% make the movie
Xclean = zeros(n, numel(T));
for ti = 1:numel(t),
    Snap = zeros(nx, ny);
    for mi = 1:numel(modes),
        mymode = modes(mi);
        if ti > round(mymode.range(1)/dt) && ...
            ti < round(mymode.range(2)/dt),
            Snap = Snap + mymode.A * mymode.u * ...
                real(mymode.lambda^ti);
        end;
    end;
    Xclean(:, ti) = Snap(:);
end;

% add noise
Noise = sig * randn(size(Xclean));
```

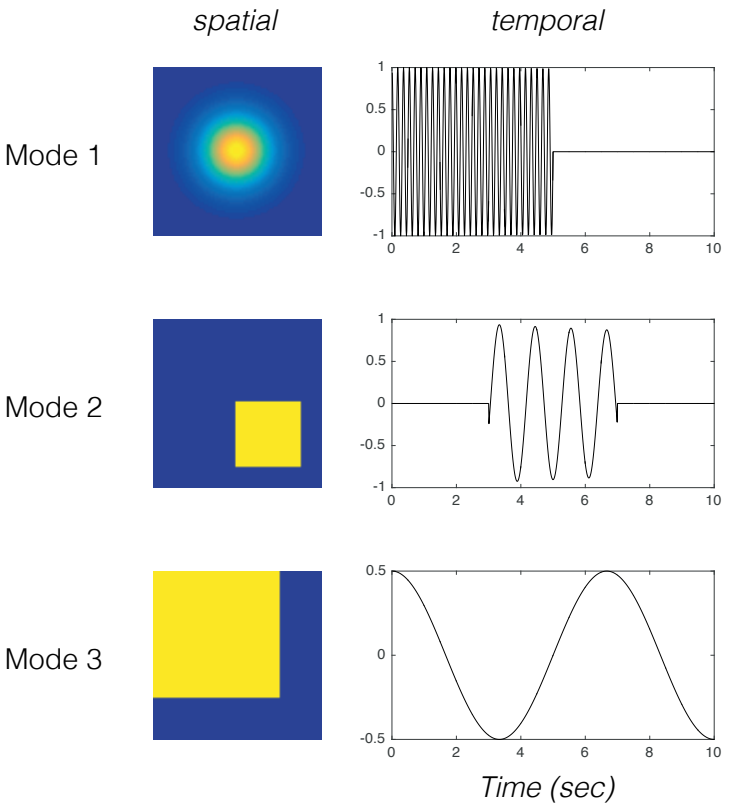

Figure 5.6. *Modes of example data to demonstrate mrDMD. The data is $n = 6400$ pixels by $m = 1000$ snapshots, sampled at $dt = 0.01$ sec. The three modes have overlapping spatial distributions and oscillate at 5.55, 0.9, and 0.15 Hz, respectively. The first two modes are only active for a part of the movie.*

```
X = Xclean + Noise;
```

To visualize the data constructed above as a movie, we use the following code.

***ALGORITHM* 5.2. Visualize example as a movie (`mrDMD_demo.m`).**

```
figure;
for ti = 1:numel(t),
    Pic = reshape(X(:, ti), nx, ny);
    imagesc(Pic, range(X(:))/2*[-1 1]);
    axis square; axis off;
    pause(dt);
end;
```

We seek a method to decompose this data and characterize its dynamics in space and time within the sampling window. The code in Algorithm **5.3** calls the function `mrDMD.m`, which implements the algorithm described in this chapter. Figure 5.7

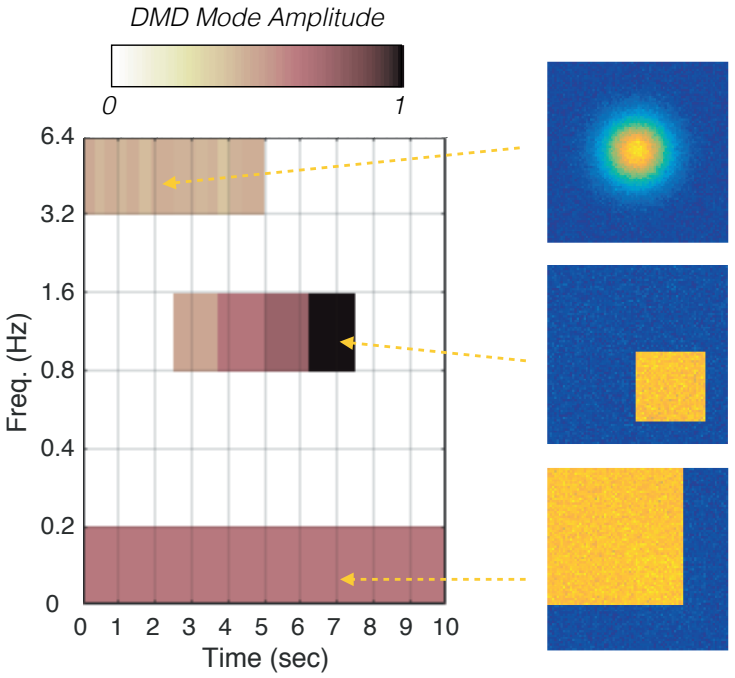

Figure 5.7. *Results of mrDMD for the spatiotemporal dynamics example in Figure* 5.6. *The multiresolution view of power as it varies in time and in frequency is shown as an image. The absolute values of spatial DMD modes at a few levels* $\phi^{(\ell,j)}$ *are shown on the right, and these closely resemble the underlying spatial modes.*

shows the resulting output as a wavelet-inspired view of the multiresolution structures present in the data, following the axes illustrated in Figure 5.4, as well as the associated spatial modes at each level of description.

Comparing these modes extracted at levels 1, 4, and 6 to the generative modes in Figure 5.6, it is clear that mrDMD is a natural description of the data. Note that where the time course of the mode does not match well with the halving windows of the algorithm, such as for the middle mode in Figure 5.7 in its first nonzero time window, our ability to estimate the mode amplitude is poor.

***ALGORITHM* 5.3. Compute mrDMD of example movie (`mrDMD_demo.m`).**

```
L = 6; % number of levels
r = 10; % rank of truncation

mrdmd = mrDMD(X, dt, r, 2, L);

% compile visualization of multires mode amplitudes
[map, low_f] = mrDMD_map(mrdmd);
[L, J] = size(mrdmd);

%%
figure;
imagesc(-map);
```

```
set(gca, 'YTick', 0.5:(L+0.5), 'YTickLabel', floor(low_f*10)/10)
    ;
set(gca, 'XTick', J/T*(0:T) + 0.5);
set(gca, 'XTickLabel', (get(gca, 'XTick')-0.5)/J*T);
axis xy;
xlabel('Time (sec)');
ylabel('Freq. (Hz)');
colormap pink;
grid on;
```

The function `mrDMD.m` used in Algorithm **5.3** is shown below. Note that this function recurses, terminating only when the parameter `L` is 1.

***ALGORITHM* 5.4. Recursive function to compute mrDMD (`mrDMD.m`).**

```
function tree = mrDMD(Xraw, dt, r, max_cyc, L)
% function tree = mrDMD(Xraw, dt, r, max_cyc, L)

% Inputs:
% Xraw      n by m matrix of raw data,
%           n measurements, m snapshots
% dt        time step of sampling
% r         rank of truncation
% max_cyc   to determine rho, the freq cutoff, compute
%           oscillations of max_cyc in the time window
% L         number of levels remaining in the recursion

T = size(Xraw, 2) * dt;
rho = max_cyc/T; % high freq cutoff at this level
sub = ceil(1/rho/8/pi/dt); % 4x Nyquist for rho

%% DMD at this level
Xaug = Xraw(:, 1:sub:end); % subsample
Xaug = [Xaug(:, 1:end-1); Xaug(:, 2:end)];
X = Xaug(:, 1:end-1);
Xp = Xaug(:, 2:end);

[U, S, V] = svd(X, 'econ');
r = min(size(U,2), r);
U_r = U(:, 1:r); % rank truncation
S_r = S(1:r, 1:r);
V_r = V(:, 1:r);

Atilde = U_r' * Xp * V_r / S_r;
[W, D] = eig(Atilde);
lambda = diag(D);
Phi = Xp * V(:,1:r) / S(1:r,1:r) * W;

%% compute power of modes
Vand = zeros(r, size(X, 2)); % Vandermonde matrix
for k = 1:size(X, 2),
    Vand(:, k) = lambda.^(k-1);
end;
```

```
% the next 5 lines follow Jovanovic et al, 2014 code:
G = S_r * V_r';
P = (W'*W).*conj(Vand*Vand');
q = conj(diag(Vand*G'*W));
Pl = chol(P,'lower');
b = (Pl')\(Pl\q); % Optimal vector of amplitudes b

%% consolidate slow modes, where abs(omega) < rho
omega = log(lambda)/sub/dt/2/pi;
mymodes = find(abs(omega) <= rho);

thislevel.T = T;
thislevel.rho = rho;
thislevel.hit = numel(mymodes) > 0;
thislevel.omega = omega(mymodes);
thislevel.P = abs(b(mymodes));
thislevel.Phi = Phi(:, mymodes);

%% recurse on halves
if L > 1,
    sep = floor(size(Xraw,2)/2);
    nextlevel1 = mrDMD(Xraw(:, 1:sep),dt, r, max_cyc, L-1);
    nextlevel2 = mrDMD(Xraw(:, sep+1:end),dt, r, max_cyc, L-1);
else
    nextlleve1 = cell(0);
    nextlevel2 = cell(0);
end;

%% reconcile indexing on output
% (because MATLAB does not support recursive data structures)
tree = cell(L, 2^(L-1));
tree{1,1} = thislevel;

for l = 2:L,
    col = 1;
    for j = 1:2^(l-2),
       tree{l, col} = nextlevel1{l-1, j};
       col = col + 1;
    end;
    for j = 1:2^(l-2)
       tree{l, col} = nextlevel2{l-1, j};
       col = col + 1;
    end;
end;
```

5.5.1 ▪ Global temperature data and El Niño

In this example, we use mrDMD on data measuring global surface temperature over the ocean. The data is open source from the NOAA/OAR/ESRL PSD, Boulder, Colorado, USA. The NOAA_OI_SST_V2 data set considered can be downloaded at http://www.esrl.noaa.gov/psd/. The data spans a 20-year period from 1990 to 2010.

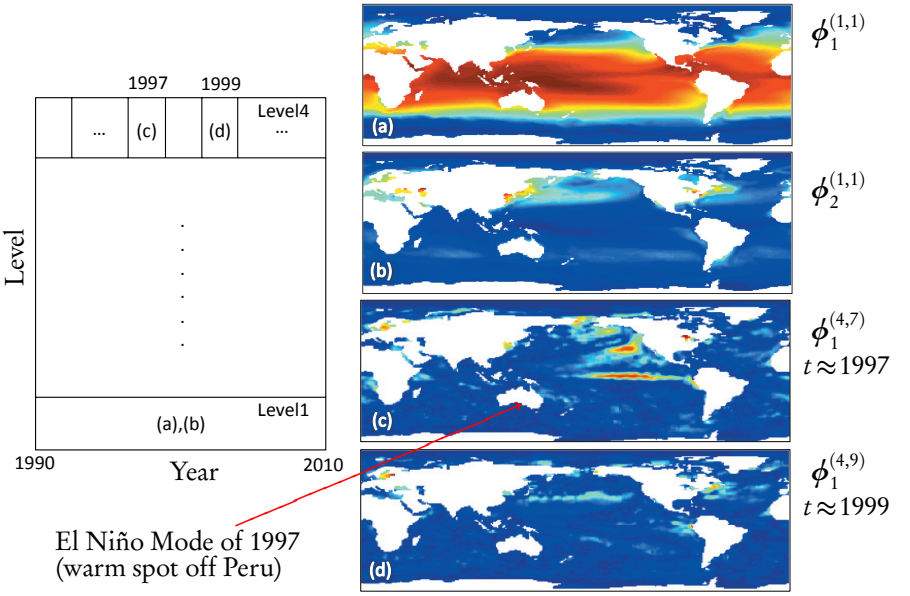

Figure 5.8. *Application of mrDMD on SST data from* 1990 *to* 2010. *The left panel illustrates the process for a four-level decomposition. At each level, modes slower than a specific cutoff are extracted. At a given level, the zero-mode component has period* $T = \infty$. *Illustrated are the* (a) *level-*1 *mrDMD mode with period* $T = \infty$ *and* (b) *Level-*1 *mrDMD mode with period* $T = 52$ *weeks. Further on in the decomposition we can extract the* (c) *level-*4 *mrDMD mode of* 1997 *and* (d) *level-*4 *mrDMD mode of* 1999. *Mode* (c) *clearly shows the El Niño mode of interest that develops in the central and east-central equatorial Pacific. The El Niño mode was not present in* 1999, *as is clear from mode* (d) [167].

Figure 5.8 shows the results of the mrDMD algorithm. Specifically, a four-level decomposition was performed with the slow spatiotemporal modes pulled at each level, as suggested in Figure 5.4. At the first level of the decomposition, two modes are extracted: the zero mode ($T = \infty$) depicted in Figure 5.8(a) and a yearly cycle ($T = 52$ weeks) shown in Figure 5.8(b). The yearly cycle is the *slowest* mode extracted at level 1. Note that the $\omega = 0$ mode component of level 1, $\boldsymbol{\phi}_1^{(1,1)}$, corresponds to the average ocean temperature over the entire 20-year data set.

We continue the mrDMD analysis through to $L = 4$. At the fourth level, the data-driven method of mrDMD discovers the 1997 El Niño mode generated from the well-known El Niño, Southern Oscillation (ENSO). El Niño is the warm phase of the ENSO cycle and is associated with a band of warm ocean water that develops in the central and east-central equatorial Pacific (between approximately the International Date Line and 120°W), including off the Pacific coast of South America. ENSO refers to the cycle of warm and cold temperatures, as measured by sea surface temperature, SST, of the tropical central and eastern Pacific Ocean. El Niño is accompanied by high air pressure in the western Pacific and low air pressure in the eastern Pacific. The cool phase of ENSO is called La Niña, with SST in the eastern Pacific below average and air

pressures high in the eastern and low in the western Pacific. The ENSO cycle, both El Niño and La Niña, causes global changes of both temperature and rainfall.

Indeed, 1997 is known to have been a strong El Niño year, as verified by its strong modal signature in the $\ell = 4$ level decomposition of mrDMD. In contrast, the same sampling window shifted down to 1999 produces no El Niño mode, which is in keeping with known ocean patterns that year. The mrDMD mode clearly shows this band of warm ocean water as a spatiotemporal mode in 1997 in Figure 5.8(c).

These results could not have been produced with DMD unless the correct sampling windows were chosen ahead of time, requiring a supervised learning step not required by mrDMD. Thus mrDMD provides a principled, algorithmic approach that is capable of data-driven discovery in data from complex systems, such as ocean/atmospheric data.

5.6 ▪ Overcoming translational and rotational invariances

The final application of mrDMD is to an example that is notoriously difficult for SVD-based methods to characterize, namely, when translational and/or rotational structures are present. Indeed, such invariances completely undermine our ability to compute low-rank embeddings of the data as driven by correlated structures, or POD/PCA modes. This has been the Achilles heel of many SVD-based methods in applications such as PCA-based face recognition, where well-cropped and centered faces are required for reasonable performance, so that translation and rotation are removed in an expensive preprocessing procedure.

In a dynamical systems setting, a simple traveling wave will appear to be a high-dimensional object in POD space despite the fact that it is only constructed from two modes, one associated with translational invariance. For dynamical cases exhibiting translation and/or rotation, Rowley and Marsden [234] formulated one of the only mathematical strategies to date to extract the low-dimensional embeddings. In particular, they developed a template-matching technique to first remove the invariance before applying SVD. Although effective, it is not suited for cases where multiple objects and time scales are present in the data.

The mrDMD approach is able to handle invariances such as translation and rotation. Consider once again the case of a simple traveling wave. Standard DMD applied to the traveling wave would result in a solution approximation requiring many DMD modes due to the slow fall-off of the singular values (1.18) in step 1 of the DMD algorithm. Further, the eigenvalues ω_k in (1.24) would also be bounded away from the origin as there is no *background* mode for such translating data.

In the mrDMD architecture, the fact that the eigenvalues are bounded away from the origin (and typically $O(1)$) in the initial snapshot window $\mathbf{X}$ would simply allow the mrDMD method to ignore the traveling wave at the first level of mrDMD. Once the domain is divided into two for the next level of analysis, the traveling wave is now effectively moving at half the speed in this new domain, i.e., the eigenvalues have migrated toward the origin and the traveling wave is now reevaluated. The recursive procedure eventually advances to a sampling window where the traveling wave looks sufficiently stationary and low rank to be extracted in the multiresolution analysis. The level at which it is extracted also characterizes the speed of the traveling wave. Specifically, the higher the level in the decomposition where the traveling wave is extracted, the faster its speed.

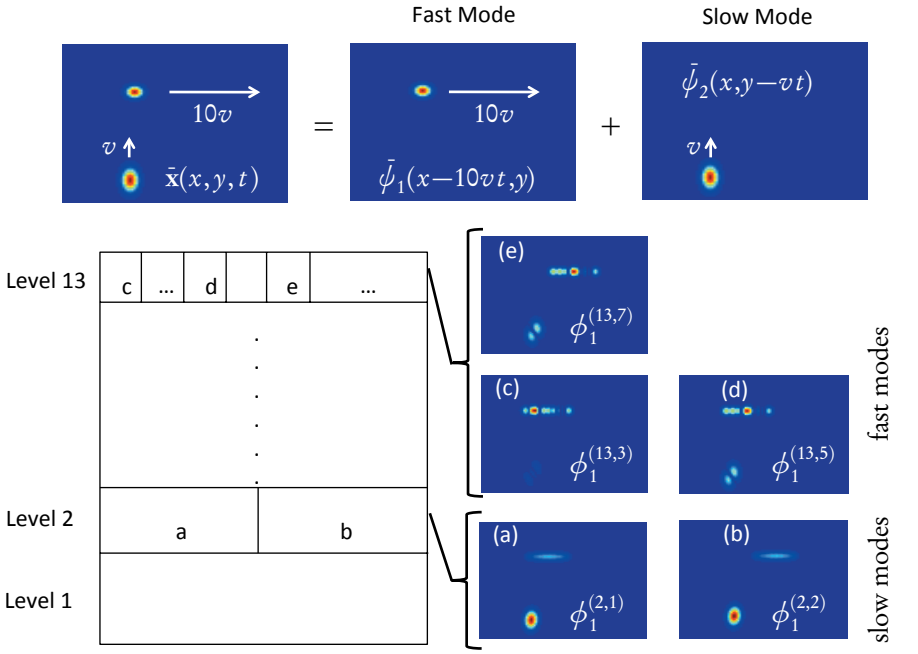

Figure 5.9. *The mrDMD approach separates moving objects. Two modes (top panel labeled "Fast Mode" and "Slow Mode") are combined in the data snapshots (top right panel). The "Fast Mode" moves rightward at speed* $10v$ *while the "Slow Mode" moves upward at speed* v*. We take* $v = 1/40$ *without loss of generality. Thus the fast and slow mode speeds differ by approximately an order of magnitude. In mrDMD (bottom left panel), the "Slow Mode" is extracted at level* 2*, as represented by panels* (a) *and* (b)*. The "Fast Mode" is extracted at level* 13*, as represented in the three representative panels* (c)–(e)*. Although there is some shadow (residual) of the slow mode on the fast mode and vice versa, mrDMD is remarkably effective at extracting the correct modes. Moreover, the level at which they are extracted can allow for a reconstruction of the velocity and direction. To our knowledge, this is the best performance achieved to date with an SVD-based method with multiple time-scale objects.*

Figure 5.9 demonstrates the application of the mrDMD method to a simple example in which there are two moving objects, one moving slowly and the other moving faster. Specifically, the example constructed results from the dynamical sequence

$$\bar{\mathbf{x}} = \bar{\psi}_1(x - 10vt, y) + \bar{\psi}_2(x, y - vt), \tag{5.21}$$

where the two modes used to construct the true solution are $\bar{\psi}_j(x,y)$ for $j = 1,2$. The two modes are Gaussians of the form $\bar{\psi}_j = \exp(-\sigma(x-x_0)^2 - \sigma(y-y_0)^2)$ with $\sigma = 0.1$, and $(x_0, y_0) = (-18, 20)$ and $(x_0, y_0) = (-20, -9)$ for the fast and slow modes, respectively. Note that the first mode is translating rightward at speed $10v$ and the second mode is translating upward at speed v. Without loss of generality, v can be set to any value. It is chosen to be $v = 1/40$ for the domain and Gaussians considered.

In this case, the straightforward template-matching procedure would fail due to the two distinct time scales of the objects. As shown in the figure, mrDMD is capable of pulling out the two modes at levels 2 (slow) and 13 (fast) of the analysis. Although there is a residual in both extracted modes, slow and fast, mrDMD does a reasonable job of

extracting the fast and slow modes. To our knowledge, this is the only SVD-based method capable of such unsupervised performance.

The motivation for this example is quite clear when considering video processing and surveillance. In particular, for many applications in this area, background subtraction is only one step in the overall video assessment. Identification of the foreground objects is the next, and ultimately more important, task. Consider the example of a video of a street scene in which there is a pedestrian walking (slow translation) along with a vehicle driving down the road (fast translation). Not only would we want to separate the foreground from the background, but also we would want to separate the pedestrian from the vehicle. More precisely, we would want to make a separate video and identify different objects in the video. The results of Figure 5.9 show that mrDMD may be effective at this kind of task.