Chapter 4
Video Processing

This chapter introduces DMD for robust separation of video frames into background (low-rank) and foreground (sparse) components in real time. The method, as originally conceived by Grosek and Kutz [121], provides a novel application of the DMD technique and its dynamical decomposition for state-of-the-art video processing. DMD modes with Fourier frequencies near the origin (zero modes) are interpreted as background (low-rank) portions of the given video frames, and modes with Fourier frequencies bounded away from the origin constitute their sparse counterparts. An approximate low-rank/sparse separation is achieved at the computational cost of just one SVD and one linear equation solve, producing results orders of magnitude faster than a competing separation method, robust principal component analysis (RPCA). The DMD method developed here is demonstrated to work robustly in real time with personal laptop-class computing power and without any parameter tuning, which is a transformative improvement in performance that is ideal for video surveillance and recognition applications [168, 91].

4.1 ▪ Background/foreground video separation

There is a growing demand for accurate and real-time video surveillance techniques. Specifically, many computer vision applications focus on algorithms that can remove *background* variations in a video stream, which are highly correlated between frames, to highlight *foreground* objects of potential interest. Background/foreground separation is typically an integral step in detecting, identifying, tracking, and recognizing objects in video sequences. Most modern computer vision applications demand algorithms that can be implemented in real time and that are robust enough to handle diverse, complicated, and cluttered backgrounds. Competitive methods often need to be flexible enough to accommodate variations in a scene, including changes in illumination throughout the day. Given the importance of this task, a variety of iterative techniques and methods have already been developed to perform background/foreground separation [175, 279, 187, 128, 56]. (See also, for instance, the recent review [25], which compares error and timing of various methods.)

One potential viewpoint for this computational task is to separate a matrix (or video) into *low-rank* (background) and *sparse* (foreground) components. Recently, this viewpoint has been advocated by Candès et al. in the framework of RPCA [56]. By

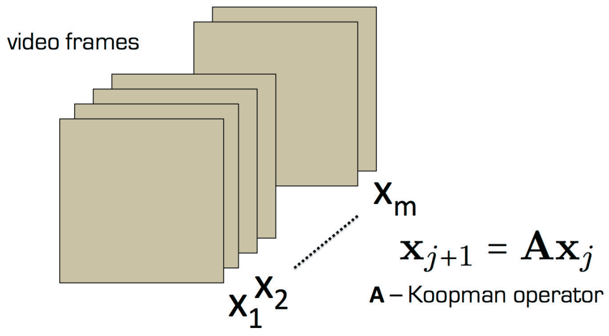

Figure 4.1. *Illustration of the DMD method where snapshots (pixels of video frames)* $\mathbf{x}_k$ *are vectorized and a linear transformation* $\mathbf{A}$ *is constructed. The DMD method constructs the best matrix* $\mathbf{A}$ *that minimizes the least-square error for all transformations* $\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k$ *with* $k = 1, 2, \ldots, m-1$.

weighting a combination of the nuclear and the ℓ_1-norms, a convenient convex optimization problem (*principal component pursuit*) was demonstrated, under suitable assumptions, to exactly recover the low-rank and sparse components of a given data matrix. The RPCA technique, which has its computational costs dominated by the convex optimization procedure, was shown to be competitive in comparison to the state-of-the-art computer vision procedure developed by De la Torre and Black [169].

Here, we use DMD instead of RPCA for video processing. In the application of video surveillance, the video frames can be thought of as snapshots of some underlying complex/nonlinear dynamics, as in Figure 4.1. DMD yields oscillatory time components of the video frames that have contextual implications. Namely, those modes that are near the origin represent dynamics that are unchanging, or changing slowly, and can be interpreted as stationary background pixels, or low-rank components of the data matrix. In contrast, those modes bounded away from the origin are changing on $\mathcal{O}(1)$ time scales or faster and represent the foreground motion in the video, or the sparse components of the data matrix. Thus, by applying the dynamical systems DMD interpretation to video frames, an approximate RPCA technique can be enacted at a fixed cost of an SVD and a linear equation solve.

Unlike the convex optimization procedure of Candès et al. [56], which can be guaranteed to exactly produce a low-rank and sparse separation under certain assumptions, no such guarantees are currently given for the DMD procedure. However, in comparison with the RPCA and other computer vision [169] methods, the DMD procedure is orders of magnitude faster in computational performance, resulting in real-time separation on laptop-class computing power.

4.2 ▪ RPCA and DMD

Given a collection of data from a potentially complex, nonlinear system, the RPCA method seeks out the sparse structures within the data, while simultaneously fitting the remaining entries to a low-rank basis. As long as the given data is truly of this nature, in that it lies on a low-dimensional subspace and has sparse components, then the RPCA algorithm has been proven by Candès et al. [56] to perfectly separate the given data $\mathbf{X}$ according to

$$\mathbf{X} = \mathbf{L} + \mathbf{S}, \tag{4.1}$$

where

$$\begin{aligned} \mathbf{L} &\rightarrow \text{low-rank}, \\ \mathbf{S} &\rightarrow \text{sparse}. \end{aligned}$$

The key to the RPCA algorithm is formulating this problem into a tractable, nonsmooth convex optimization problem known as *principal component pursuit* (PCP):

$$\begin{aligned} \arg\min \ & ||\mathbf{L}||_* + \lambda ||\mathbf{S}||_1 \\ \text{subject to } \ & \mathbf{X} = \mathbf{L} + \mathbf{S}. \end{aligned} \tag{4.2}$$

Here PCP minimizes the weighed combination of the nuclear norm, $||\mathbf{M}||_* := \text{trace}\left(\sqrt{\mathbf{M}^*\mathbf{M}}\right)$, and the ℓ_1-norm, $||\mathbf{M}||_1 := \sum_{ij} |m_{ij}|$. The scalar regularization parameter is nonnegative: $\lambda \geq 0$. From the optimization problem (4.2), it can be seen that as $\lambda \rightarrow 0$, the low-rank structure will incorporate all of the given data, $\mathbf{L} \rightarrow \mathbf{X}$, leaving the sparse structure as a zero matrix. It is also true that as λ increases, the sparse structure will embody more and more of the original data matrix, $\mathbf{S} \rightarrow \mathbf{X}$, as L commensurately approaches the zero matrix [56, 166].

In effect, λ controls the dimensionality of the low-rank subspace; however, one does not need to know the rank of $\mathbf{L}$ a priori. Candès et al. [56] have shown that the choice

$$\lambda = \frac{1}{\sqrt{\max(n, m)}}, \tag{4.3}$$

where $\mathbf{X}$ is $n \times m$, has a high probability of success at producing the correct low-rank and sparse separation provided that the matrices $\mathbf{L}$ and $\mathbf{S}$ are incoherent, which is the case for many practical applications.

Although there are multiple algorithms that can solve the convex PCP problem, the *augmented Lagrange multiplier* (ALM) method stands out as a simple and stable algorithm with robust, efficient performance characteristics. The ALM method is effective because it achieves high accuracies in fewer iterations when compared against other competing methods [56]. Moreover, there is an *inexact* ALM variant [177] to the *exact* ALM method [176], which is able to converge in even fewer iterations at the cost of weaker guaranteed convergence criteria. MATLAB code that implements these methods, along with a few other algorithms, can be downloaded from the University of Illinois Perception and Decision Lab website [2]. This is the code implemented throughout this chapter.

4.2.1 ▪ Video interpretation of RPCA

In a video sequence, stationary background objects appear as highly correlated pixel regions from one frame to the next, which suggests a low-rank structure within the video data. The snapshot in each frame is two-dimensional by nature; pixels need to be reshaped into one-dimensional column vectors and united into a single data matrix $\mathbf{X}$ (see Figure 4.1). The RPCA algorithm can then implement the background/foreground separation found in $\mathbf{X} = \mathbf{L} + \mathbf{S}$, where the low-rank matrix $\mathbf{L}$ is the background and the sparse matrix $\mathbf{S}$ is a complementary video of the moving foreground objects. Because the foreground objects exhibit a spatial coherency throughout the video, the

RPCA method is no longer guaranteed a high probability of success; however, in practice, RPCA achieves an acceptable separation almost every time [56].

As λ is decreased, the sparse reconstruction of the video $\mathbf{S}$ starts to bring in more of the original video, including erroneous stationary pixels that should be part of the low-rank background. When λ is increased, the sparse reconstruction of the video begins to see a decrease in the pixel intensities that correspond to the moving objects, and some foreground pixels disappear altogether.

4.2.2 ▪ RPCA interpretation of DMD

The DMD algorithm can be used to produce a similar low-rank and sparse separation as in (4.1). For the DMD case, the separation relies on the interpretation of the ω_k frequencies in the solution reconstructions represented in general by (1.5) and more specifically in DMD by (1.24). Low-rank features in video are such that $|\omega_j| \approx 0$; that is to say, they are slowly changing in time. Thus, if one sets a threshold to gather all the low-rank modes where $|\omega_j| \leq \epsilon \ll 1$, then the separation can be accomplished. This reproduces a representation of the $\mathbf{L}$ and $\mathbf{S}$ matrices of the form

$$\begin{aligned} \mathbf{L} &\approx \sum_{|\omega_k|\leq\epsilon} b_k \boldsymbol{\phi}_k \exp(\omega_k t), \\ \mathbf{S} &\approx \sum_{|\omega_k|>\epsilon} b_k \boldsymbol{\phi}_k \exp(\omega_k t). \end{aligned} \tag{4.4}$$

Note that the low-rank matrix $\mathbf{L}$ picks out only a small number of the total number of DMD modes to represent the *slow* oscillations or direct current (DC) content in the data ($\omega_j \approx 0$). This DC content is exactly the background mode when interpreted in the video stream context.

The advantage of the DMD method and its sparse/low-rank separation is the computational efficiency of achieving (4.4), especially when compared to the optimization methods thus far developed. However, the DMD method does not do well for delta function–like behaviors in time, i.e., a very rapid on/off behavior. Such a behavior would require many Fourier modes in time to resolve, undermining the method of associating correlated spatial activity with single oscillatory behaviors in time.

4.3 ▪ DMD for background subtraction

A video sequence offers a natural application for DMD because the frames of the video are equally spaced in time, and the pixel data collected at every snapshot can readily be vectorized. Given a video with m frames, the $n_x \times n_y = n$ pixels in each frame can be extracted as $n \times 1$ vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$. DMD can attempt to reconstruct any given frame by calculating $\mathbf{x}_{\text{DMD}}(t)$ at time t. The validity of the reconstruction depends on how well the specific video sequence meets the assumptions and criteria of the DMD method.

To reconstruct the entire video, consider the $1 \times m$ time vector $\mathbf{t} = [t_1 \; t_2 \; \cdots \; t_m]$, which contains the times at which the frames were collected. The video sequence $\mathbf{X}$ is reconstructed with DMD as follows:

$$\mathbf{X}_{\text{DMD}} = \sum_{k=1}^{r} b_k \boldsymbol{\phi}_k e^{\omega_k \mathbf{t}} = \boldsymbol{\Phi} \text{diag}(\exp(\omega t))\mathbf{b}. \tag{4.5}$$

Notice that ϕ_k is an $n \times 1$ vector, which is multiplied by the $1 \times m$ vector $\mathbf{t}$ to produce the proper $n \times m$ video size. By the construction of the DMD methodology, $\mathbf{x}_1 = \mathbf{\Phi b}$, which means that $\mathbf{\Psi b}$ renders the first frame of the video with a dimensionality reduction chosen through the parameter r.

Thus, the diagonal matrix of frequencies ω dictates how that first frame gets altered over time to reconstruct the subsequent frames. It becomes apparent that any portion of the first video frame that does not change in time, or changes very slowly in time, must have an associated Fourier mode (ω_k) located near the origin in complex space: $||\omega_k|| \approx 0$. This observation makes it possible to separate background (approximate low-rank) information from foreground (approximate sparse) information with DMD.

Assume that ω_p, where $p \in \{1,2,\ldots,r\}$, satisfies $||\omega_p|| \approx 0$ (typically this is only a single mode) and that $||\omega_k|| \;\forall\; k \neq p$ is bounded away from zero. We may now rewrite (4.5) as

$$\mathbf{X}_{\text{DMD}} = \underbrace{b_p \phi_p e^{\omega_p \mathbf{t}}}_{\text{Background Video}} + \underbrace{\sum_{k\neq p} b_j \phi_j e^{\omega_k \mathbf{t}}}_{\text{Foreground Video}}. \tag{4.6}$$

Assuming that $\mathbf{X} \in \mathbb{R}^{n\times m}$, then a proper DMD reconstruction should also produce $\mathbf{X}_{\text{DMD}} \in \mathbb{R}^{n\times m}$. However, each term of the DMD reconstruction is potentially complex, $b_k \phi_k \exp(\omega_k \mathbf{t}) \in \mathbb{C}^{n\times m} \;\forall\, k$, although they sum to a real-valued matrix. This poses a problem when separating the DMD terms into approximate low-rank and sparse reconstructions, because real-valued outputs are desired for a video interpretation, and proper handling of the complex elements can make a significant difference in the accuracy of the results.

Consider calculating DMD's approximate low-rank reconstruction according to

$$\mathbf{X}_{\text{DMD}}^{\text{Low-Rank}} = b_p \phi_p e^{\omega_p \mathbf{t}}.$$

Since it should be true that

$$\mathbf{X} = \mathbf{X}_{\text{DMD}}^{\text{Low-Rank}} + \mathbf{X}_{\text{DMD}}^{\text{Sparse}}, \tag{4.7}$$

then DMD's approximate sparse reconstruction,

$$\mathbf{X}_{\text{DMD}}^{\text{Sparse}} = \sum_{k\neq p} b_k \phi_k e^{\omega_k \mathbf{t}},$$

can be calculated with real-valued elements only as follows:

$$\mathbf{X}_{\text{DMD}}^{\text{Sparse}} = \mathbf{X} - \left|\mathbf{X}_{\text{DMD}}^{\text{Low-Rank}}\right|,$$

where $|\cdot|$ yields the modulus of each element within the matrix. However, this may result in $\mathbf{X}_{\text{DMD}}^{\text{Sparse}}$ having negative values in some of its elements, which would not make sense in terms of having negative pixel intensities. These residual negative values can be put into an $n \times m$ matrix $\mathbf{R}$ and then added back into $\mathbf{X}_{\text{DMD}}^{\text{Low-Rank}}$ as follows:

$$\mathbf{X}_{\text{DMD}}^{\text{Low-Rank}} \leftarrow \mathbf{R} + \left|\mathbf{X}_{\text{DMD}}^{\text{Low-Rank}}\right|,$$
$$\mathbf{X}_{\text{DMD}}^{\text{Sparse}} \leftarrow \mathbf{X}_{\text{DMD}}^{\text{Sparse}} - \mathbf{R}.$$

In this way, the magnitudes of the complex values from the DMD reconstruction are accounted for, while maintaining the important constraints in (4.7) so that none of the pixel intensities are below zero and ensuring that the approximate low-rank and sparse DMD reconstructions are real valued. As demonstrated in the next section, this method works well empirically.

4.4 ▪ Simple example and algorithm

Let us first consider a simple example of one-dimensional dynamics in time. Specifically, we can consider a function comprising a time-independent background mode mixed with a dynamical time-varying mode. Our objective is to use DMD to extract the two modes due to their time-independent and time-dependent nature. The specific modes considered are

$$\begin{aligned} f(x,t) &= f_1(x) + f_2(x,t) \\ &= 0.5\cos(x) + 2\operatorname{sech}(x)\tanh(x)\exp(i2.8t), \end{aligned} \tag{4.8}$$

where $f_1(x)$ is time independent and $f_2(x,t)$ is time dependent. The code in Algorithm **4.1** constructs the example data.

ALGORITHM **4.1. Mixing of two signals.**

```
%% Define time and space discretizations
n = 200;
m = 80;
x = linspace(-15,15,n);
t = linspace(0,8*pi,m);
dt = t(2) - t(1);
[Xgrid,T] = meshgrid(x,t);

%% Create two spatiotemporal patterns
f1 = 0.5*cos(Xgrid) .* (1+0*T);  % time-independent!
f2 = (sech(Xgrid).*tanh(Xgrid)) .* (2*exp(1j*2.8*T));

%% Combine signals and make data matrix
X = (f1 + f2)'; % Data Matrix

figure;
surfl(real(X'));
shading interp; colormap(gray); view(-20,60);
```

The DMD algorithm follows the standard steps outlined in the introduction. In this particular case (Algorithm **4.2**), we also perform a rank $r = 50$ decomposition and sift out the zero eigenvalue generated by the background mode.

ALGORITHM **4.2. DMD of signals.**

```
%% Create data matrices for DMD
X1 = X(:,1:end-1);
X2 = X(:,2:end);

%% SVD and rank-50 truncation
r = 50; % rank truncation
[U, S, V] = svd(X1, 'econ');
Ur = U(:, 1:r);
```

```
Sr = S(1:r, 1:r);
Vr = V(:, 1:r);

%% Build Atilde and DMD Modes
Atilde = Ur'*X2*Vr/Sr;
[W, D] = eig(Atilde);
Phi = X2*Vr/Sr*W;  % DMD Modes

%% DMD Spectra
lambda = diag(D);
omega = log(lambda)/dt;

figure;
plot(omega, '.');
```

The omega eigenvalues that are nearly zero are considered background modes since they have no time evolution, i.e., they are time independent. The background mode associated with the zero value omega is then used to construct the DMD background mode (Algorithm **4.3**).

ALGORITHM **4.3. Separate foreground and background.**

```
bg = find(abs(omega)<1e-2);
fg = setdiff(1:r, bg);

omega_fg = omega(fg); % foreground
Phi_fg = Phi(:,fg); % DMD foreground modes

omega_bg = omega(bg); % background
Phi_bg = Phi(:,bg); % DMD background mode
```

The foreground and background DMD modes can then be used to reconstruct the DMD solution. In particular, two solutions are achieved: a DMD reconstruction of the foreground video and a DMD reconstruction of the background video (or mode). The following code constructs these two objects.

ALGORITHM **4.4. Foreground/background reconstruction.**

```
%% Compute DMD Foreground Solution
b = Phi_bg \ X(:, 1);
X_bg = zeros(numel(omega_bg), length(t));
for tt = 1:length(t),
    X_bg(:, tt) = b .* exp(omega_bg .* t(tt));
end;
X_bg = Phi_bg * X_bg;
X_bg = X_bg(1:n, :);

figure;
surfl(real(X_bg'));
shading interp; colormap(gray); view(-20,60);

%% Compute DMD Background Solution
b = Phi_fg \ X(:, 1);
```

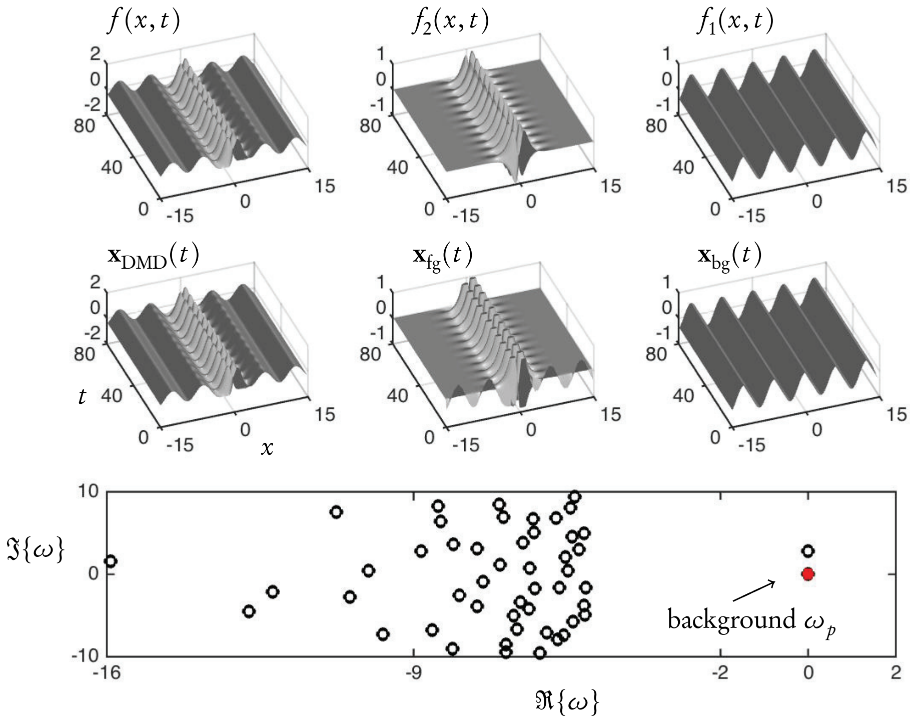

Figure 4.2. *Foreground/background separation using a simple example $f(x,t)$ (top left) constructed from time-dependent dynamics $f_2(x,t)$ and time-independent mode $f_1(x)$ (top middle and top right, respectively). The DMD reconstruction for full dynamics is mirrored in the middle panels, with the full reconstruction, foreground (time-dependent) behavior, and background (time-independent) mode given by $\mathbf{x}_{DMD}(t)$, $\mathbf{x}_{fg}(t)$, and $\mathbf{x}_{bg}(t)$, respectively. The DMD spectral distribution of eigenvalues ω_j is shown in the bottom panel, with the zero mode $\omega_p \approx 10^{-15}$ highlighted in red.*

```
X_fg = zeros(numel(omega_fg), length(t));
for tt = 1:length(t),
    X_fg(:, tt) = b .* exp(omega_fg .* t(tt));
end;
X_fg = Phi_fg * X_fg;
X_fg = X_fg(1:n, :);

figure;
surfl(real(X_fg'));
shading interp; colormap(gray); view(-20,60);
```

The above codes rely on the same algorithm as in § 1.4. The only difference is the separation step in the DMD eigenvalues. Once separated, reconstruction is done in the standard way with two distinct reconstructions based on the eigenvalues. Figure 4.2 shows the original example along with the DMD reconstruction and DMD spectra. Specifically, the top panel of the figure shows the two initial signals, one time-stationary ($f_1(x)$) and the other not ($f_2(x,t)$), along with the combined signal $f(x,t) = f_1(x) + f_2(x,t)$. The middle panel shows the full DMD reconstruction $\tilde{\mathbf{x}}(t)$ along with the DMD reconstruction of the foreground (time-dependent) object $\tilde{\mathbf{x}}_2(t)$

and the background (time-independent) behavior $\tilde{\mathbf{x}}_1(t)$. The DMD spectra is shown in the bottom panel, and the background eigenvalue $\omega_p \approx 10^{-15}$ is shown in red.

4.5 ▪ DMD for video surveillance

This framework easily generalizes to more complex video streams. Here we use the open-source Advanced Video and Signal based Surveillance (AVSS) Datasets, (www.eecs.qmul.ac.uk/ ~andrea/avss2007_d.html), specifically the "Parked Vehicle—Hard" and "Abandoned Bag—Hard" videos. The DMD separation procedure can be compared and contrasted against the RPCA procedure. The original videos were converted to grayscale and down-sampled in pixel resolution to $n = 120 \times 96 = 11520$ to make the computational memory requirements manageable for personal computers. Also, the introductory preambles to the surveillance videos, which constitute the first 351 frames of each video, were removed because they are irrelevant for the following illustrations.

The video streams are broken into segments of $m = 30$ frames each, which were analyzed individually using both the RPCA and the DMD methods. Frame numbers 500, 1000, and 2000 of the entire video stream are depicted in Figures 4.3 and 4.4, along with their separation results for easy comparison. Although one has the option of manually tuning the regularization parameter of the RPCA method to best suit the given application, for consistency λ is set at $\lambda = (\sqrt{n})^{-1} \approx 9.32 \cdot 10^{-3}$, as is suggested by Candès et al. [56] for creating a reliable automatic algorithm. Likewise, the dimensionality reduction parameter of the DMD method is held constant at $r = m - 1 = 29$. For enhanced contrast and better visibility, the sparse results from both methods are artificially brightened by a factor of 10.

Consider Figure 4.3 of the AVSS "Parked Vehicle" surveillance video, which shows various vehicles traveling along a road, with a traffic light (not visible) and a crosswalk (visible) near the bottom of the frame. There are a few vehicles parked alongside the road. Sometimes, in the distance, moving vehicles become difficult to perceive by eye due to limitations of the pixel resolution. Note that some extraneous pixels in the sparse structure can be eliminated by applying a simple thresholding criterion based on pixel intensity.

Next, let's consider Figure 4.4 of the AVSS "Abandoned Bag" surveillance video, which consists of people walking, standing, and sitting as trains come and go in a subway station. Shadows are relevant in this video because they move with their respective foreground objects, and they sometimes change the background significantly enough to be viewed as extensions of the moving objects themselves. Note that, for frames 500 and 1000, both methods struggle with the fact that between the numerous moving objects and their shadows, many of the pixels in the video change intensity at some point. Observe that objects in motion create movement trails that extrapolate the object's motion both forward and backward in time. When the object is dark, its corresponding movement trail is bright, and vice versa.

4.5.1 ▪ Timing Performance

The RPCA and DMD methods have comparably good background/foreground separation results. The difference in the separation quality seems to depend on the specific situation, likely because the assumptions that determine when the RPCA and DMD methods will perform well are also different. Given the relatively consistent quality of the separation results, other factors, such as computational effort, become the dis-

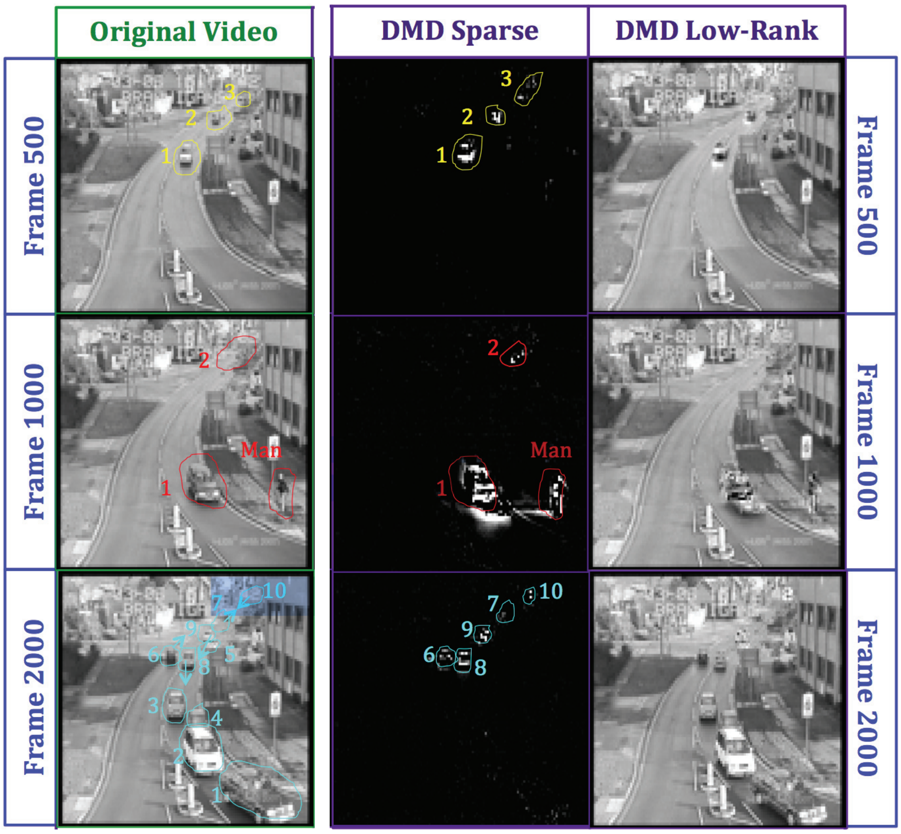

Figure 4.3. *The DMD background/foreground separation results are illustrated for three specific frames in the "Parked Vehicle" video. The* 30*-frame video segment that contains frame* 500 *has three vehicles driving in the same lane toward the camera. The* 30*-frame video segment that contains frame* 1000 *has a man stepping up to a crosswalk as a vehicle passes by, and a second car in the distance, barely perceptible, starts to come into view. The* 30*-frame video segment that contains frame* 2000 *has three vehicles stopped at a traffic light at the bottom of the frame, with another two vehicles parked on the right side of the road, and five moving vehicles, two going into the distance and three coming toward the vehicles waiting at the light, the last vehicle being imperceptible to the eye at this pixel resolution:* $n = 11520$.

tinguishing characteristics that determine which method is more suitable for the given application.

The key difference in effectiveness between the RPCA and DMD algorithms is found in computational time needed to complete the background/foreground separation. Again, consider the AVSS Datasets, with the two videos used previously: "Parked Vehicle" and "Abandoned Bag." For a timing performance experiment, consider having these videos down-sampled to various pixel resolutions n and separated into various video segment sizes m. Figure 4.5 shows the computational time required for RPCA and DMD averaged over both videos, fixing either number of pixels or video segment sizes. Both the exact ALM and the faster inexact ALM convex optimization routines were used to solve the PCP problem (4.2) of the RPCA method.

In Figure 4.5, the empirical computational times are plotted on a logarithmic scale,

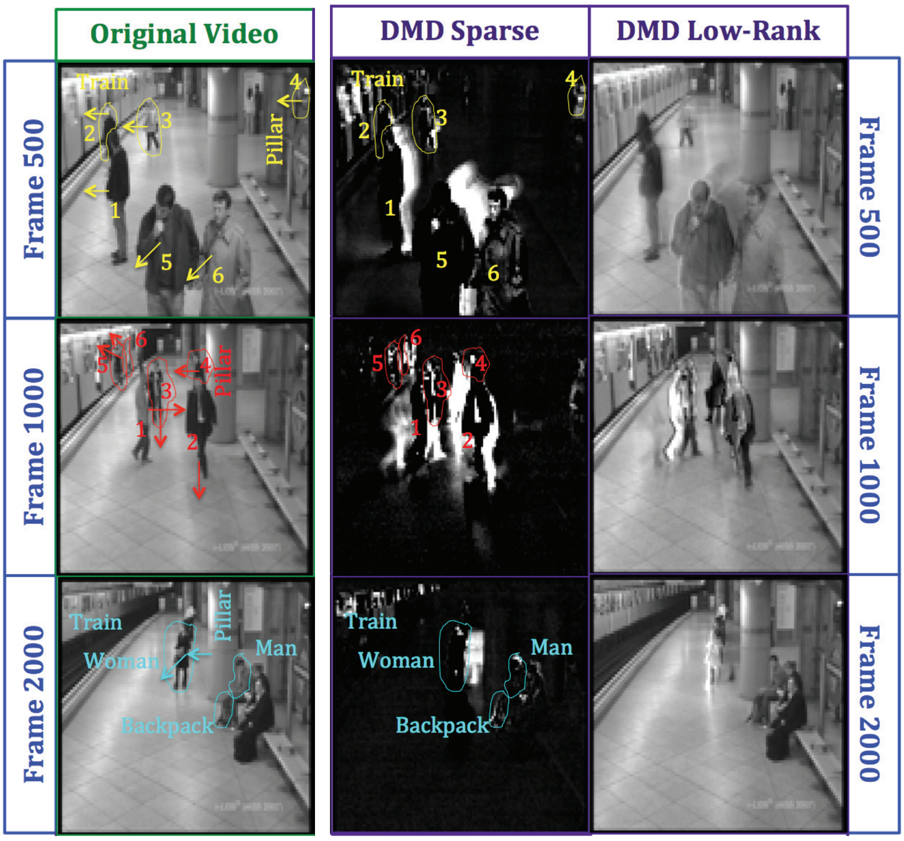

Figure 4.4. *The DMD background/foreground separation results are illustrated for three specific frames in the "Abandoned Bag" video. The* 30*-frame video segment that contains frame* 500 *has three people stepping slightly closer to an arriving train, and another person walking behind a support pillar toward the train in the upper right area of the frame. The two people closest to the camera walk toward the bottom of the frame. The* 30*-frame video segment that contains frame* 1000 *has four people walking in different directions in the middle of the frame, one of whom comes out from behind a support pillar, while, farther down the platform, two people enter the train. The* 30*-frame video segment that contains frame* 2000 *has a woman walk out from behind a support pillar, move to the left, and then turn somewhat toward the camera. The train is moving and the man sitting closest to the support pillar adjusts his backpack.*

along with best-fit curves found by the linear least-squares method. It is clear that the DMD method is about two to three orders of magnitude faster than its RPCA method counterpart using the exact ALM optimization procedure and about one order of magnitude faster when the inexact ALM optimization procedure is employed. In fact, many cameras operate at a rate of about 20 to 30 frames per second, and DMD can be computed for those video segments in about 0.1–0.01 seconds for high- and low-resolution images, respectively. This efficiency makes real-time, online data processing possible, even without down-sampling.

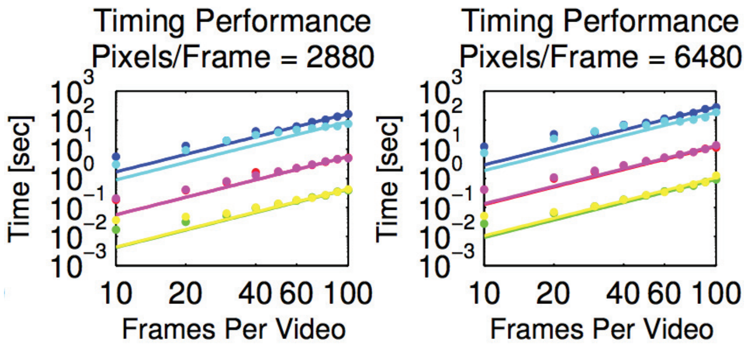

Figure 4.5. *DMD is faster than RPCA at foreground/background separation by orders of magnitude. For the "Abandoned Bag" and "Parked Vehicle" videos, the computational times of the DMD (green and yellow, respectively), inexact ALM RPCA (red and magenta, respectively), and exact ALM RPCA (blue and cyan, respectively) background/foreground separation methods are graphed on a logarithmic scale. The number of pixels per frame is fixed at* $\{2880, 6480\}$*, respectively. This timing data fits reasonably well with a quadratic fit:* $t = cs^2$*, where* t *is the computational time,* $c \in \mathbb{R}$*, and* s *is the video segment size (number of frames per video segment). Timing was assessed on a* 1.86 *GHz Intel Core* 2 *Duo processor using MATLAB.*

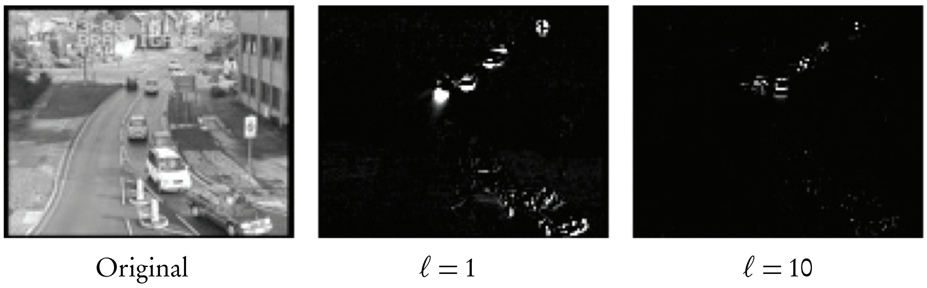

Figure 4.6. *The effects of the parameter* r *that controls the dimensionality reduction of the DMD process are depicted. The original frame, and the DMD sparse reconstructions of that frame for* $r = 1$ *and* $r = 10$*, artificially brightened by* 10 *times the true intensity, are illustrated for frame* 2000 *of the "Parked Vehicle" video* [1] *using a* 30*-frame video segment (see Figure 4.3). Notice that the sparsity increases with the parameter* r *because dimensionality reduction has the effect of blurring the motion between frames, thus smearing the motion out over a greater number of pixels in any given frame. Even under significant dimensionality reduction* ($r = 1$)*, the background/foreground separation is still accomplished reasonably well. By* $r = 10$*, the method approximates the full-rank reconstruction* ($r = 29$)*.*

4.5.2 ▪ DMD and rank truncation

As discussed in detail in Chapter 8, one parameter in the DMD algorithm is r, the rank truncation that decreases the dimensionality of the decomposition. To show the effect that the dimensionality reduction parameter r has on the DMD separation process, consider the foreground DMD results represented in Figure 4.6.

The surprising result here is the quality of the low-rank and sparse separation even when r is set as low as 1. Rank truncation seems to have the effect of blurring the video frames together. It is possible that the results of $r = 1$ are in part because the vehicles

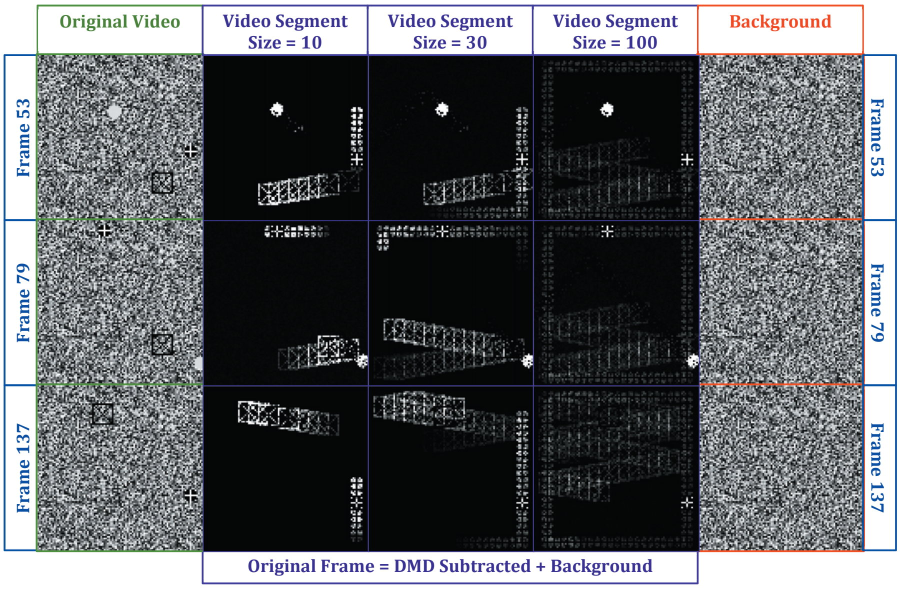

Figure 4.7. *Analysis of the DMD background/foreground separation with random background noise. The left column shows the original frames with the random background noise. The middle three columns show the sparse DMD results using* 10, 30, *and* 100 *frames per video segment, which are artificially brightened by a factor of* 10 *to increase the contrast and visibility of the results against a black background, and the right column shows the true background noise that was added to each frame. The erroneous pixels contribute to the mean pixel intensity error of the background.*

in the video of Figure 4.6 did not move very far within the frame, so the smearing of pixels over the places of movement is very localized. This dimensionality reduction constitutes a *time-saving* procedure that can potentially reduce the video separation costs even further.

4.5.3 ▪ Error analysis

To accurately measure how well the DMD method captures actual foreground movement, and not the stationary background, it is helpful to have an artificially constructed video where the true background and foreground are known. The error could then be measured precisely between the actual and DMD-reconstructed backgrounds. Moreover, the quality of reconstruction can also be compared with the RPCA technique.

One such video was constructed to calculate the error in the DMD low-rank/sparse separation method with 300 frames in total, each being 100×100 pixels. The background to the video is produced by a uniform random grayscale-intensity field, ranging from pure black to pure white. The background remains constant throughout the entire video.

This video contains three moving objects: (1) a black square, 7×7 pixels in size, with four white pixels in each corner and a white "plus" sign centered in the middle; (2) a light gray circle with a diameter of 9 pixels; and (3) a transparent square, 13×13 pixels in size, lined by black pixels and with a black "X" centered within. Object (1) revolves, at a constant rate of 4 pixels per frame, counterclockwise around the inside

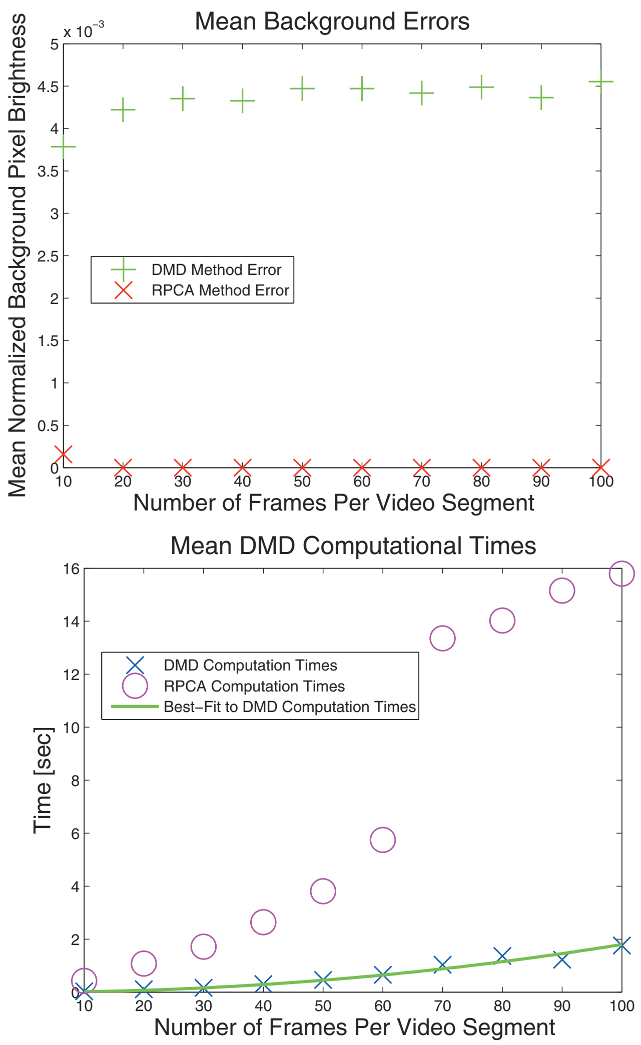

Figure 4.8. *Analysis of the DMD background/foreground separation error. The mean pixel intensity error for the entire video at each video segment size is plotted in the top figure on a normalized intensity scale where* 1 *is pure white and* 0 *is pure black, the target background color. The mean computational times* t*, using MATLAB on a* 1.86 *GHz Intel Core 2 Duo processor, as a function of the video segment sizes* s *is plotted in the bottom figure, yielding a quadratic best-fit:* $t = 1.52 \cdot 10^{-4} s^2$, $R^2 = 0.983$.

edges of the frame starting at the top left corner of the frame for the entire duration of the video. Object (2) enters the video on frame 25 on the left side of the frame, moving up and to the left at a constant rate of 2 pixels per frame in both directions. This object reflects downward and eventually leaves the frame after bouncing off an imaginary wall located a fifth of the way down from the top of the frame. Object (3) enters the video on frame 50 on the right side of the frame, moving down and to the left. Once this object enters the frame, it stays inside the frame for the rest of the video, reflecting off the edges of the frame. Object (3) maintains a constant vertical velocity of 1 pixel per frame and a horizontal velocity of 6 pixels per frame. At various times these objects overlap one another, and the precedence is that object (1) is drawn first, then object (2), then object (3), burying the previously drawn objects under the newly drawn object.

The foreground results of the DMD method applied to this video are illustrated in Figure 4.7 for frame numbers 53, 79, and 137. Note that object (3) cannot be seen in the results because it is pure black. This exemplifies a limitation to the DMD method, in that when there is an object of low pixel intensity, it may be difficult to distinguish it from the zero entries that naturally occur in a sparse matrix. Note that the spurious pixels are both residuals of previous frames and projections of where the objects will be moving in future frames, and have inverted pixel intensities compared to their corresponding objects' intensities. This is much like what is seen in the foreground DMD results of frames 1000 and 2000 of the "Abandoned Bag" video in Figure 4.4, where the walking man (frame 1000) and walking woman (frame 2000) have obvious, erroneous movement trails.

The mean, normalized pixel intensity error for this constructed video is shown in Figure 4.8. This error is calculated on the first $m-1$ frames of the video segments of length m because the last frame is where the DMD procedure accumulates its error. Note that the background to the sparse DMD-reconstructed videos should be pure black, and on an intensity scale of 0 to 1, the average sparse background pixel intensity is on the order of 10^{-3}, compared to the true average background intensity of nearly identically 0.5. Recall that, in some cases, thresholding techniques can eliminate spurious background pixels from the foreground results and improve the measured error.

One might have expected that the DMD algorithm errors would have decreased as the video segment sizes increased, because having more frames means that the DMD method has more information to work with. However, because of this anomaly with black objects leaving white movement trails, there are extraneous pixels for every video segment size. As the videos get longer, these erroneous movement trails also get longer, projecting into both the past and the future, increasing the amount of error in proportion to the increase in number of frames per video segment. However, the longer videos do produce less-bright pixel intensity trails, which helps reduce the error, which is consistent with the idea that they should be able to make more use of the extra information that they have.

The RPCA-reconstructed foreground error results are also presented for comparison, where the suggested $\lambda = (\sqrt{n})^{-1} = 0.01$ is used. In this case, the RPCA perfectly reconstructs every video for segment sizes greater than 10 frames.

Figure 4.8 confirms the quadratic growth in computational times that the DMD and RPCA schemes experience as the video segment sizes are increased. This quadratic growth trails off slightly for very small segment sizes, likely due to inherent processing times that do not scale with data size. For segment size 10, there is, of course, the option of retuning the regularization parameter λ in RPCA. However, in most cases where the true solution is unknown, this must be done manually by time-consuming reruns of the RPCA algorithm.

Finally, it should be noted that as object size increases relative to the frame size, the DMD reconstruction error also increases. Likewise, and not surprisingly, the DMD reconstruction error increases if fewer frames are used to show the same object movement. Not all types of object movement are accurately reconstructed with the DMD method; however, it seems that high speeds and acceleration can still be handled fairly well given enough snapshots to capture the movement. Objects that were once moving and then stop are events that are not well reconstructed by the DMD method.

4.5.4 ▪ Iterative application of DMD

One of the advantages of the RPCA method that is illustrated in the preceding subsection is that it can perform exact low-rank and sparse decompositions; thus the error can be reduced to zero. No such guarantees are available for the DMD algorithm. However, we highlight an interesting observation concerning the iterative use of DMD. Grosek and Kutz [121] demonstrated how the DMD method can be applied iteratively to empirically converge toward the true low-rank and sparse components of the given data. The successive DMD iterations are applied to the approximate sparse structure, creating a better approximate sparse structure and adding more data back into the low-rank structure. In the future, finding the exact rate of convergence and understanding the theoretical basis for this convergence will be important steps toward establishing an analytical connection between the RPCA and DMD separation algorithms. The performance of DMD for video background subtraction may also benefit from streaming DMD algorithms [130].