5

Results for `cars` video is followed by results of `flame` video. Also checkout the attached code files.

a)

Extracted Frames are displayed in 2, 4.

b)

Coded snapshots are displayed in 1, 3. Each snapshot is displayed in two ways, the top images are required scaled coded snapshots while the bottom images give view more information per pixel to understand the snapshot better.

c)

We know that the final 2-D image $I(x, y)$ is given by

$$I(x,y) = \sum_{t=1}^{T} C_t(x,y) \cdot F_t(x,y) \; \forall \; x,y \tag{1}$$

We can rewrite the expression in terms of matrix-vector product where each pixel signifies an element of the vector as below

$$\underbrace{\begin{bmatrix} \text{diag}(\text{vec}(C_1)) & \text{diag}(\text{vec}(C_2)) & \cdots & \text{diag}(\text{vec}(C_T)) \end{bmatrix}}_{\boldsymbol{A} \in \mathbb{R}^{HW \times HWT}} \underbrace{\begin{bmatrix} \text{vec}(F_1) \\ \text{vec}(F_2) \\ \vdots \\ \text{vec}(F_T) \end{bmatrix}}_{\boldsymbol{x} \in \mathbb{R}^{HWT}} = \underbrace{\begin{bmatrix} \text{vec}(I) \end{bmatrix}}_{\boldsymbol{b} \in \mathbb{R}^{HW}} \tag{2}$$

Here, vec($\boldsymbol{A}$) is conversion of an $m \times n$ matrix into a $mn-$sized vector. As the expression in equation 1 is same for each pixel (x, y), the ordering of entries while conversion doesn't matter.

d)

For each patch i, we can write $\boldsymbol{A} = \boldsymbol{\Phi}\boldsymbol{\Psi}$, where $\boldsymbol{\Phi}$ is the measurement matrix and $\boldsymbol{\Psi}$ is the sparsifying basis matrix (common for all patches).

Now, for each time frame F_i, we can write $\boldsymbol{H} \in \mathbb{R}^{H_pW_p \times H_pW_p}$ such that $\text{vec}(F_t) = \boldsymbol{H}\,\text{vec}(\hat{F}_t)$ where $\boldsymbol{H}$ is taken as the 2-D DCT matrix.

$$\underbrace{\overbrace{\begin{bmatrix}\text{diag}(\text{vec}(C_1)) & \text{diag}(\text{vec}(C_2)) & \cdots & \text{diag}(\text{vec}(C_T))\end{bmatrix}}^{\boldsymbol{\Phi}\in\mathbb{R}^{H_pW_p\times H_pW_pT}} \overbrace{\begin{bmatrix}\boldsymbol{H} & \boldsymbol{0} & \cdots & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{H} & \cdots & \boldsymbol{0}\\ \vdots & \vdots & \ddots & \vdots\\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{H}\end{bmatrix}}^{\boldsymbol{\Psi}\in\mathbb{R}^{H_pW_pT\times H_pW_pT}}}_{\boldsymbol{A}} \underbrace{\begin{bmatrix}\text{vec}(\hat{F}_1)\\ \text{vec}(\hat{F}_2)\\ \vdots\\ \text{vec}(\hat{F}_T)\end{bmatrix}}_{\boldsymbol{\theta}\in\mathbb{R}^{H_pW_pT}} = \underbrace{\begin{bmatrix}\text{vec}(I)\end{bmatrix}}_{\boldsymbol{b}\in\mathbb{R}^{H_pW_p}} \tag{3}$$

Here, the dimensions of each patch is $(H_p, W_p) = (8, 8)$.

Note that here $\boldsymbol{x}$ is given by $\boldsymbol{\theta}$, also, similar to previous part, due to the independent nature of the equation 1 for each pixel, we can again convert the relevant part of the patch from each these matrices to vectors and the equation will hold true.

e), f)

Figure 2, 4 display the reconstruction results and the RMSE values using ISTA with $\lambda = 25$. This value was achieved using trial-and-error

g)

Images from both videos were cropped to a suitable cropping area of size 120×240.

h)

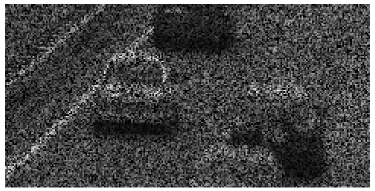
Refer to figure 2, 3 for all required data.

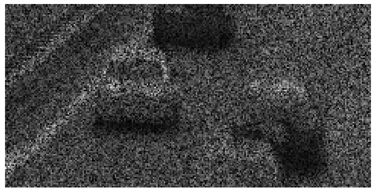
Observations

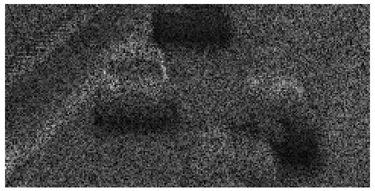
- As T increases, the number of pixels never measured decreases
- As T increases, the smoothness in the snapshot increases
- As T increases, the RMSE value and the smoothness in the reconstructed images increases, which results in loss of detail. An adaptive value of λ in ISTA, might reduce these effects to some level
- As λ increases, the smoothness of reconstructed images increases and the number of artifacts reduce

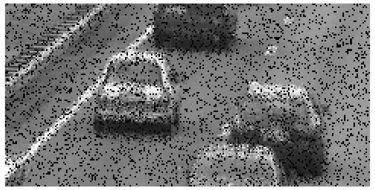
- As λ increases, the threshold value of ISTA is reached quicker and hence the speed of reconstruction increases
- Reconstruction of `flame` frames is much better than `cars` as can be seen using RMSE, a possible reason could be low amount of details in the former

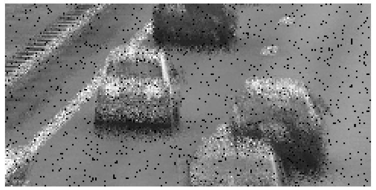
Results


(a) $T = 3$


(b) $T = 5$


(c) $T = 7$


(d) $T = 3$


(e) $T = 5$

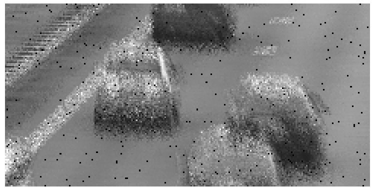
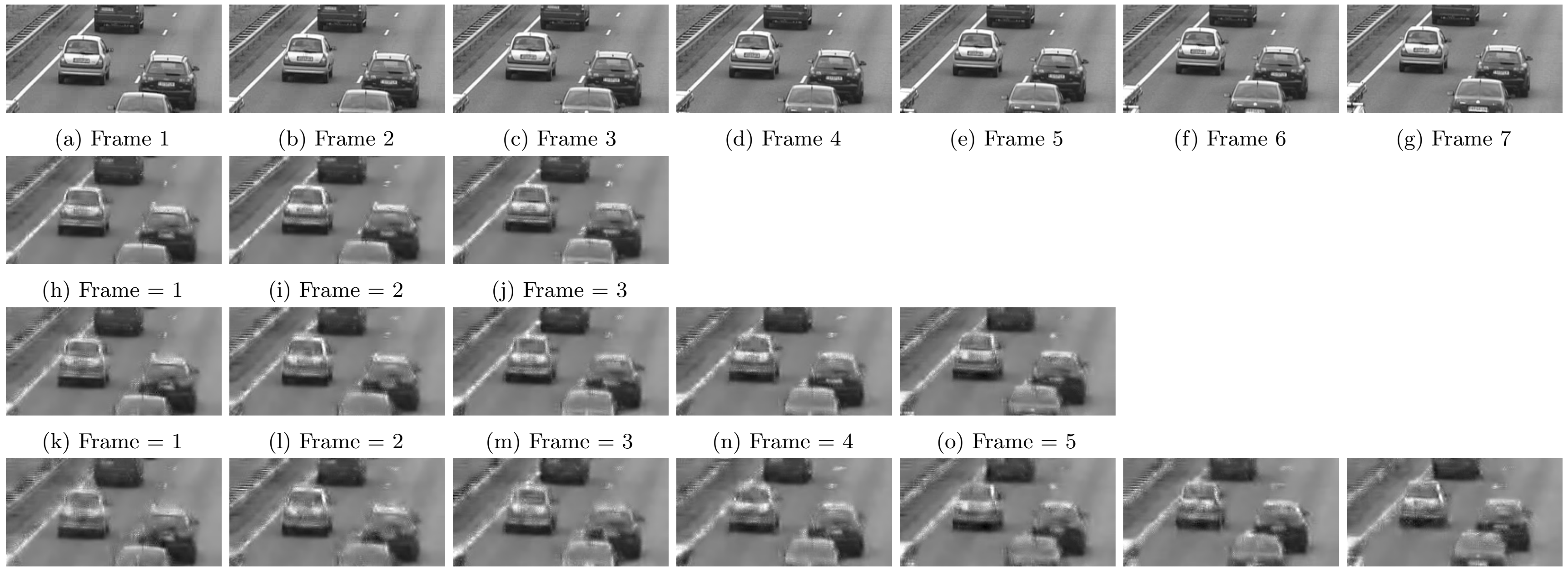

(f) $T = 7$

Figure 1:
Row 1: Coded Snapshot, displays average value of each pixel
Row 2: Coded Snapshot, displays weighted average value of each pixel using the code as the weight



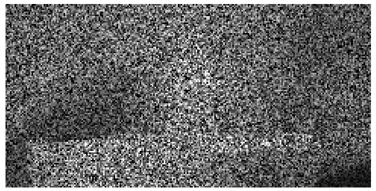
(a) Frame 1 (b) Frame 2 (c) Frame 3 (d) Frame 4 (e) Frame 5 (f) Frame 6 (g) Frame 7

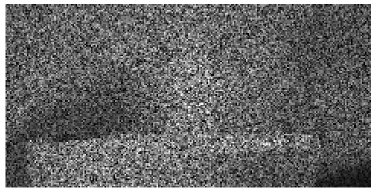
(h) Frame = 1 (i) Frame = 2 (j) Frame = 3

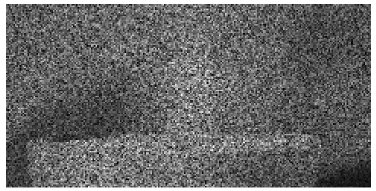
(k) Frame = 1 (l) Frame = 2 (m) Frame = 3 (n) Frame = 4 (o) Frame = 5

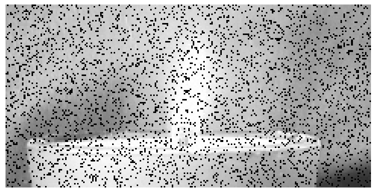
(p) Frame = 1 (q) Frame = 2 (r) Frame = 3 (s) Frame = 4 (t) Frame = 5 (u) Frame = 6 (v) Frame = 7

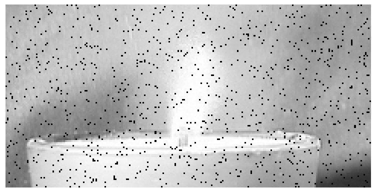
Figure 2:
Row 1: Original Images
Row 2: Reconstructed Images for $T = 3$, RMSE = 0.12107
Row 3: Reconstructed Images for $T = 5$, RMSE = 0.14484
Row 4: Reconstructed Images for $T = 7$, RMSE = 0.16047


(a) $T = 3$


(b) $T = 5$


(c) $T = 7$


(d) $T = 3$


(e) $T = 5$

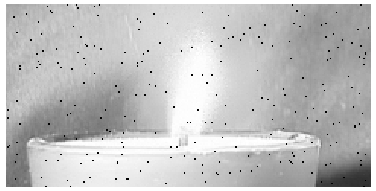
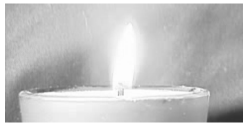
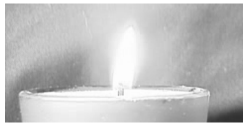
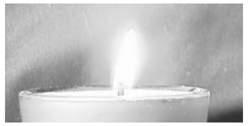
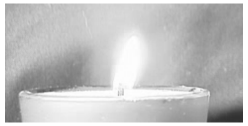

(f) $T = 7$

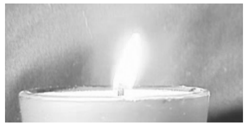
Figure 3:
Row 1: Coded Snapshot, displays average value of each pixel
Row 2: Coded Snapshot, displays weighted average value of each pixel using the code as the weight

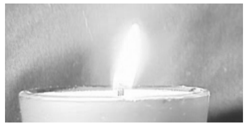

(a) Frame 1

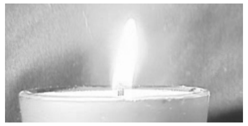

(b) Frame 2


(c) Frame 3


(d) Frame 4


(e) Frame 5


(f) Frame 6


(g) Frame 7

(h) Frame = 1

(i) Frame = 2

(j) Frame = 3

(k) Frame = 1

(l) Frame = 2

(m) Frame = 3

(n) Frame = 4

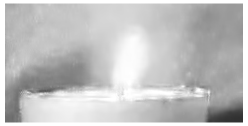
(o) Frame = 5

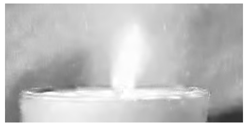
(p) Frame = 1

(q) Frame = 2

(r) Frame = 3

(s) Frame = 4


(t) Frame = 5


(u) Frame = 6

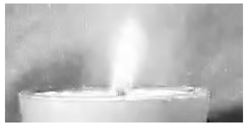

(v) Frame = 7

Figure 4:
Row 1: Original Images
Row 2: Reconstructed Images for $T = 3$, RMSE = 0.025309
Row 3: Reconstructed Images for $T = 5$, RMSE = 0.028312
Row 4: Reconstructed Images for $T = 7$, RMSE = 0.032038