Measuring Flow Complexity in Videos

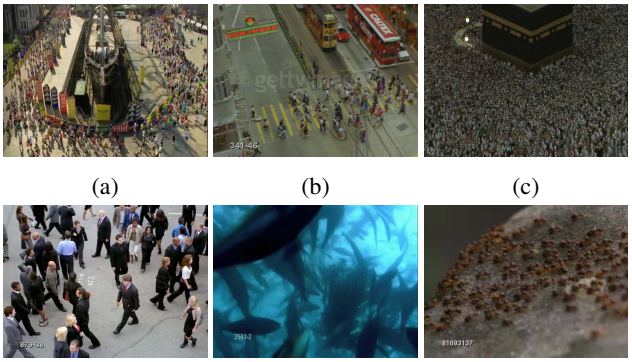
Saad Ali
Center for Vision Technologies, SRI International
201 Washington Rd., Princeton, NJ, USA
saad.ali@sri.com

Abstract

In this paper a notion of flow complexity that measures the amount of interaction among objects is introduced and an approach to compute it directly from a video sequence is proposed. The approach employs particle trajectories as the input representation of motion and maps it into a 'braid' based representation. The mapping is based on the observation that 2D trajectories of particles take the form of a braid in space-time due to the intermingling among particles over time. As a result of this mapping, the problem of estimating the flow complexity from particle trajectories becomes the problem of estimating braid complexity, which in turn can be computed by measuring the topological entropy of a braid. For this purpose recently developed mathematical tools from braid theory are employed which allow rapid computation of topological entropy of braids. The approach is evaluated on a dataset consisting of open source videos depicting variations in terms of types of moving objects, scene layout, camera view angle, motion patterns, and object densities. The results show that the proposed approach is able to quantify the complexity of the flow, and at the same time provides useful insights about the sources of the complexity.

1. Introduction

Motion or flow analysis has been an important area of research in computer vision and over the decades significant advances have been made in solving the problems related to motion estimation [14, 18], tracking of moving objects [15, 16, 17], motion segmentation [19, 20, 21] and motion pattern understanding [22, 23]. In general, all motion analysis methods work under certain assumptions about the nature (or complexity) of motion. For instance, assumptions about a sparse or a dense flow field, planar or non-planar motion, number of independently moving objects, etc., are implicitly encoding author's or practitioner's belief about the complexity of the motion that can be handled by their algorithm. The question arises, is there an objective way to measure this complexity directly from a video sequence, thus making the notion of motion or flow complexity explicit?



(a) (b) (c)

(d) (e) (f)

Figure 1: Frames from videos depicting movements with different level of flow complexity. (a) Athletes participating in a marathon, (b) People crossing a street, (c) Pilgrims performing Hajj, (d) People walking in random directions, (e) A school of fish, and (f) A colony of ants.

In this paper we attempt to address this question by proposing a measure that quantifies the flow or motion complexity. For this purpose, the flow complexity is considered as a measure that captures the amount of collective *interaction* among objects resulting due to the kinematic and mixing properties of the underlying optical flow field. The unique kinematic and mixing properties of a flow could be a result of one of many possible factors that include presence of high density of objects in a scene, frequent spatial alteration between positions of objects, magnitude of object velocities, articulated movement of parts of an object, chaotic motion, or transition between flows of qualitatively different dynamics (e.g. from free to congested flow).

To further elaborate the notion of flow complexity, we show frames from several video sequences in Fig. 1. These videos depict a variety of scenes and motion patterns including athletes in a marathon (Fig. 1a), groups of people merging (Fig. 1b & 1d), thousands of people circling in an extremely dense formation (Fig. 1c), a school of fish (Fig. 1e)

and a colony of ants (Fig. 1f). While many current methods can compute optical flow, segment motion patterns, and track objects in these videos, there is a lack of approaches that can objectively answer the question: what is the complexity of flow in these sequences? Is the flow resulting from a large number of athletes in a marathon more complex than the motion of a small group of people crossing the street? How complex is the flow of a dense crowd in Fig. 1c? Can we compare the complexity of flows generated by different entities, e.g., an ant colony vs. a school of fish?

The approach proposed in this paper to compute flow complexity has several ingredients. First, particle trajectories are used as the representation of the flow in a video. This allows capturing of flow dynamics over longer duration of times. The trajectories may correspond to interest points (e.g. KLT) [17], complete objects or their parts. Second, we make a key observation that 2D trajectories of particles take the form of a braid in space-time due to intermingling (or interaction) among the objects observed in a video. A braid is defined as any complex structure or pattern formed by intertwining three or more strands. The common notion of intertwining in braids and intermingling in trajectories is what allows us to make the connection between the two. As a result, the problem of estimating the flow complexity is mapped into the problem of estimating the braid complexity, which in turn can be computed by measuring the topological entropy of the braid. The topological entropy can be used as a measure of flow complexity because the degree of entanglement of trajectories over time, which signifies the complexity of the underlying flow, is directly proportional to the topological entropy of the corresponding braid. Third, we compute the topological entropy of a braid using several mathematical tools from braid theory [11, 8]. These tools allow rapid computation of topological entropy even with a large number of trajectories [11, 8]. The terms braid entropy, flow entropy and topological entropy will be used interchangeably in the text.

2. Related Work

There are several pieces of work that attempt to characterize video sequences based on their motion content. Primarily these approaches focus on using statistics derived from magnitude, orientation, and regularity of instantaneous optical flow vectors. For instance, Chen et al. [1] developed a motion entropy measure to distinguish high motion intensity frames from low motion intensity frames. Statistical distributions of both direction and magnitude are employed for this purpose and the idea is that a set of regular motion vectors, which will appear more frequently in a low motion intensity frames, would generate a lower entropy (i.e. low motion complexity) while a set of irregular motion vectors, which will appear more frequently in a high motion intensity frames, would generate a higher entropy (i.e. high motion complexity). Similar notion of motion entropy has been used for video watermarking in [3]. On the same lines Ma et al. [2] employed an entropy measure defined on angle distributions of motion vectors while Liu et al. [6] used magnitude between successive frames along the dominant motion direction for characterization of motion in a video sequence. Iyengar et al. [7] also developed motion-based measures for quantifying temporal and spatial activity as part of finding optimal video encoders.

The work of Peker et al. [4] is also closely related where an attempt has been made to measure the subjective perception of motion activity using a set of motion descriptors. The descriptors are again based on optical flow vectors and are designed to capture the mean, variance, and difference of motion vector magnitudes. The motion activity is also one of the motion features included in the visual part of the MPEG-7 standard and has been used to describe the level or intensity of activity, action, or motion in a video sequence [5] .

The proposed algorithm differs from these works in two aspects. First, our approach is based on a trajectory based representation as opposed to the instantaneous optical flow based representation. It has been shown by the body of work on motion pattern analysis ([24][15][23]) that particle trajectories integrate motion information over longer durations of time, and therefore capture the dynamics of the underlying flow in a more reliable and robust way. Second, our approach explicitly takes into consideration the interaction among objects (see Sec. 3.5) while measuring the flow complexity. As interactions among objects emerge overtime, statistics derived from instantaneous optical flow are not suitable to characterize interaction driven motion complexity. In addition, the need to capture interactions naturally leads to a trajectory-based formulation for which sound theoretical framework of braids is well suited.

The main contributions of this work are: (1) a trajectory based formulation of flow complexity that explicitly takes into consideration object *interactions*, and (2) introduction of several techniques from braid theory to the computer vision community which could be used to analyze trajectory representations in several other problem areas (e.g. action recognition, motion segmentation etc.).

3. Algorithm Overview

In this section, we describe various pieces of the proposed flow complexity computation algorithm and discuss their conceptual and implementation details.

Fig. 2 shows a high-level block diagram of the algorithm. Given an input video, the algorithm starts by computing optical flow between consecutive frames of the video (Step 1). The computed optical flow fields are stacked together to generate a 3D flow volume through which a dense grid of

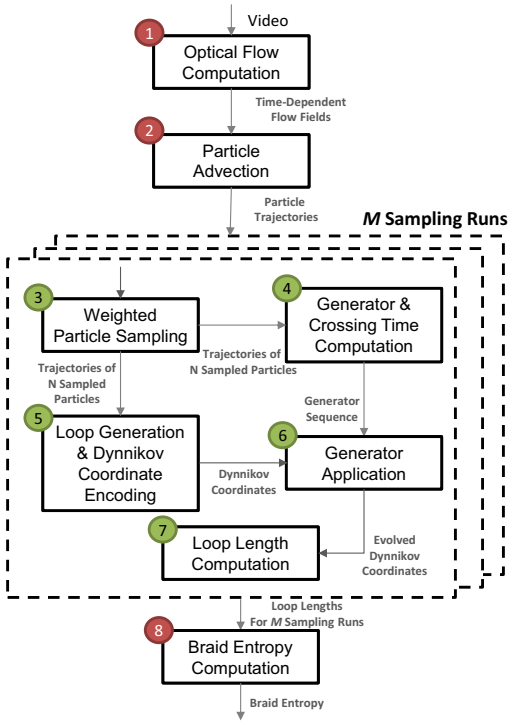
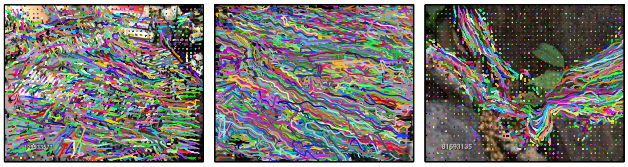


Figure 2: A high-level block diagram showing algorithmic steps of the flow complexity computation.

particles is advected (Step 2). The advection process generates a dense collection of trajectories which are sampled to obtain a subset of N trajectories (Step 3). Next, N sampled trajectories are mapped into a braid and a generator sequence is computed for this braid by locating the precise times at which particles exchange their positions along the chosen projection axis (Step 4). In parallel, a collection of K random loops, encompassing the N sampled trajectories, are initialized and encoded using Dynnikov coordinates (Step 5). The generators computed at Step 4 are applied to these loops to evolve them in time and their final Dynnikov coordinates are obtained (Step 6). The length of each loop is computed over time using Eq. 2 (Step 7). The above process (Step 3 to 7) is repeated for M trajectory sampling runs. Finally, the growth rate is computed for fixed time intervals for each loop. The growth rate is averaged over M runs and K loops (i.e. by $M \times K$) and a line is fitted to the logarithm of the resulting growth rate. The slope of the line is used as a measure of topological entropy or flow complexity (Step 8). Each step is described next.

3.1. Optical Flow Computation

Given a video optical flow between consecutive pairs of frames is computed [26]. From preliminary experiments, it is observed that the exact choice of optical flow algorithm is not critical, however, it is important to obtain both small and large scale motion structures. Often



(a) (b) (c)

Figure 3: Trajectories obtained by advecting a grid of particles. (a) A high-angle view of a street intersection, (b) A zoomed-in view of a street intersection, (c) a colony of ants.

due to explicit flow smoothness constraints small scale motion structures are not preserved during optical flow computation [18]. However, these small scale structures can contribute significantly to the complexity of the flow, especially in scenarios containing a dense collection of objects. Therefore, we lower the weight associated with the smoothness term during optical flow computation which helps in emphasizing both the large and the small scale motion. For a given interval $(t, T+t)$, the optical flow fields, $\mathbf{U}(t), \mathbf{U}(t+1), \ldots, \mathbf{U}(t+T)$, are pooled to generate a 3D flow volume.

3.2. Particle Advection

The trajectory-based representation of the flow needs to capture the kinematics and dynamics of the underlying flow. For this purpose, particle advection is used to obtain a dense collection of trajectories [24, 25]. Using advection has several advantages: First, it provides a comprehensive coverage of the underlying flow, and second it helps in sidestepping the shortcomings of current object tracking algorithms especially in presence of complex motion. The trajectories for the interval $(t, t+T)$ are obtained by overlaying a grid of particles on the first flow field, $\mathbf{U}(t)$. Next, a trajectory $[\mathbf{X}(t+T; t, x_0, y_0), \mathbf{Y}(t+T; t, x_0, y_0)]$ corresponding to a particle at the grid location (x_0, y_0) is computed by numerically solving the ordinary differential equations: $\frac{dx}{dt} = U(x, y, t)$, $\frac{dy}{dt} = V(x, y, t)$, subject to the initial conditions $[X(t), Y(t)] = (x_0, y_0)]$. Here U and V represent two components of the optical flow, and interval $(t+T)$ represents the temporal duration of the 3D flow volume. During advection, cubic interpolation is used to obtain optical flow values at sub-pixel locations. The typical choice of T in our case is 5 seconds (or 150 frames at 30 fps). Currently, particles are not reseeded once they leave the frame boundaries. A visualization of trajectories obtained through this method is provided in Fig. 3.

3.3. Weighted Particle Sampling

Particle advection results in a large number of trajectories (e.g. 76800 trajectories for a 240×320 resolution video). Some of these trajectories originate from areas depicting dominant flows (e.g. pedestrian crossing) while oth-

(a) (b) (c)

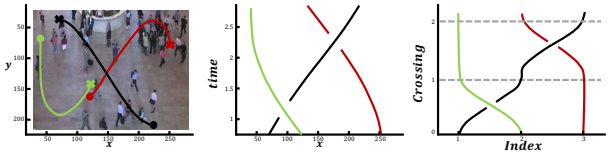
Figure 4: Sampled trajectories using the procedure described in Section 3.3. The trajectories are sampled from the dense collection of trajectories in Fig. 3.

ers may reside in portions of the video that usually do not have significant motion content. Similarly, some trajectories may be fragmented while others may be longer in duration. To reduce the computational complexity of subsequent steps, the number of trajectories are decreased through a sampling step. The sampling is based on the observation that a particle with a large displacement has a better chance of capturing the dynamics of the flow, and therefore should be preferred during sampling. A weight, w, is assigned to each trajectory $[x(t+T;t,x_0,y_0), y(t+T;t,x_0,y_0)]$, where w is directly proportional to the sum of displacements over time, i.e., $\Delta d = \sum_{i=t}^{t+T-1} \text{sqrt}((X(i+1;i,x_t,y_t) - X(i;i,x_t,y_t))^2 + (Y(i+1;i,x_t,y_t) - Y(i;i,x_t,y_t))^2)$. For sampling, the weights are normalized across all trajectories so they sum up to 1 and represent a discrete density. Sampling is performed from this discrete density with replacement. Examples of sampled trajectories are shown in Fig. 4. A typical value of $N = 50$ is used during sampling.

3.4. Mapping of Trajectories to Braid

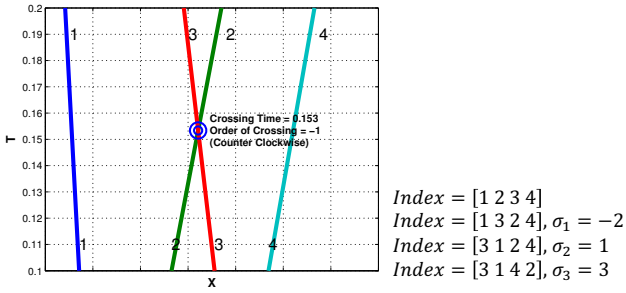
The sampled trajectories are next mapped into a braid. A *physical braid* is defined as a collection of three or more strands that are intertwined. Since trajectories are space-time constructs and their intermingling results in intertwining as well, each trajectory can be considered as a strand of a braid. Using this similarity, the geometric representation of a braid is obtained by projecting each trajectory into the xt−plane [8]. This process is shown in Fig. 5b where three trajectories (or strands) from Fig. 5a are projected onto the plane containing the x- (or any other chosen axis) and the time-axis. This projection captures the organization of trajectories (i.e. which trajectory is next to or behind another trajectory) and also preserves the *crossing* information (along y-axis or the axis perpendicular to the chosen axis). Note that exchange in positions along x-axis in the geometric representation of the braid is called a *crossing*. Another important thing to note is that any braid is uniquely defined by how the trajectories intertwine or cross in the projected view. The trajectories can be perturbed but the braid will be invariant to this perturbation as long as the direction, number, or order of crossing is not changed [9, 8].

From the implementation point of view, the trajectory



(a) (b) (c)

Figure 5: (a) Trajectories corresponding to three particles with cross representing the start and dot representing the current position; (b) Geometric representation of the braid where trajectories are projected onto the xt-plane; (c) Algebraic representation of the braid where trajectories are ordered and indexed in terms of their x-axis coordinate.



(a) (b)

Figure 6: (a) Demonstrates the process of computing crossing times by finding line intersections. (b) Shows how index locations of particles are updated and generator sequence is created from the crossing time information.

projection starts by sorting the sampled trajectories in terms of their position along the projection axis (e.g. x-axis) at initial time t. The index of location resulting from the sorting is assigned to each trajectory as its starting identity. For instance, Fig. 6a shows four trajectories that are projected onto the xt-plane and based on their positions are assigned identities 1 to 4 . During the next time step, t to $t+1$, pairwise crossings between all trajectories are computed using line intersection computation. For each resulting crossing (e.g. the blue marker in Fig. 6a) three pieces of information are saved: the time of the crossing (which is 0.153 in Fig. 6a), the direction which is either clockwise or counter-clockwise based on the value along y-axis at the time of crossing (counter clockwise in Fig. 6a), and the order of the trajectories before crossing (which is $[2, 3]$ in Fig. 6a). After all pairs of trajectories are analyzed, the resulting crossings are collected and sorted by the time of crossing.

3.5. Generator and Crossing Time Computation

Generators are operators that act on a braid by taking a pair of trajectories (or strands) and crossing them. They allow conversion of the geometric representation of the braid into an algebraic representation, as shown in Fig. 5c. Here

two generators act in a sequence where the first generator causes the green trajectory to pass in front of the black trajectory, while the second generator causes the black trajectory to pass in front of the red trajectory.

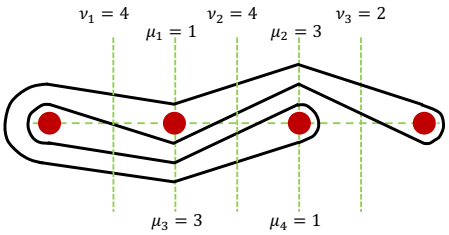
A numerical value of a specific generator is assigned as follows: The trajectories involved in the crossing are indicated by their index location which becomes the magnitude of the generator, and the direction of crossing (clockwise or counter-clockwise) is indicated by the sign (positive or negative) of the generator. Since geometrically it is only possible to cross two adjacent trajectories, the magnitude of the generator is set to the lower index of the trajectory involved in the crossing. Fig. 6b shows a pictorial description where the first generator switches the position of particle indexed 2 with the particle indexed 3. Referring to a generator by σ, the magnitude of the first generator becomes -2 due to counter clockwise change in position. The subsequent generators (σ_2, σ_3) update the index locations further. Each generator is stored along with its crossing time as this information is needed for evolving the loops in Section 3.7. We refer the readers to reference [8] for a comprehensive description of the process of extracting generators from a geometric representation of braids.

3.6. Loop Generation and Dynnikov Coordinate Encoding

Now that trajectories are converted into a geometric representation of braid and a sequence of generators is computed, the next step is to utilize this information for computing the entropy of the braid. This is achieved by using the concept of a *loop* where loop is a non-self intersecting closed curve or a region that passes around or encapsulates a set of particles. An example loop is shown in Fig. 7 where it is surrounding a sample consisting of four particles. The intuition behind using loops is that the length of a loop surrounding a set of particles involved in complex motion will grow exponentially over time, while a loop surrounding particles that exhibit simple motion e.g., particles moving in a straight line, will not grow rapidly. Therefore, the "growth of a loop" can be used to infer entropy of the braid, and in turn the complexity of underlying motion. The loop generation consists of two steps [8]: i) Symbolic Encoding of Loops; and ii) Dynnikov Encoding.

Symbolic Encoding of Loops: The symbolic encoding of a loop surrounding a set of particles is achieved by introducing a coordinate system which uniquely defines a loop. This is possible as any non-self intersecting closed loop, winding around particles, can be reconstructed by counting the number of intersections with fixed reference lines [8], for instance green vertical line in Fig. 7. Intuitively the number of intersections is capturing the information that how many times a loop passes above, below or between particles.

Given reference lines, the encoding scheme for a randomly initialized loop around particles proceeds as follows: Let μ_i counts the number of intersections of boundary of the loop with the reference line *above and below* the particles, and ν_i counts the number of intersections of the boundary of loop with the reference lines *in between* the particles. The number of crossings above and below the first and the last particle (along the projection line) is not required as that can be deduced from the other crossing information [9, 12]. Fig. 7 provides a visualization of this initial loop encoding step where the above and below crossings and their magnitudes are $\mu_1 = 1$, $\mu_2 = 3$, $\mu_3 = 3$ and $\mu_4 = 1$, while the midpoint crossings and their magnitudes are $\nu_1 = 4$, $\nu_2 = 4$ and $\nu_3 = 2$. For an N trajectory sample it requires $2N - 4$ dimensional coordinates to encode one loop.



(a)

Figure 7: A closed loop encapsulating four particles. Vertical reference lines are used to encode the loop in Dynnikov coordinates [12] while the horizontal reference line is used for computing length of the loop (see Eq. 2).

Dynnikov Encoding: The initial loop encoding is further condensed using a minimal coordinate system called Dynnikov coordinates [12]. This is done by taking difference between adjacent μ and ν values, that is following operation is performed for $i = 1 \dots (N-2)$:

$$a_i = \frac{1}{2}(\mu_{2i} - \mu_{2i-1}), b_i = \frac{1}{2}(\nu_i - \nu_{i+1}), \qquad (1)$$

where N is the number of trajectories. The Dynnikov coordinates are essentially a comparison of number of times a loop passes above or below the $(i+1)th$ particle (captured by a_i), and the number of times loop passes to the left and to the right of $(i+1)th$ particle (captured by b_i). Both a_i and b_i are signed integers and they are concatenated to generate Dynnikov coordinate representation of the loop, $u = (a_i, \dots, a_{n-2}, b_i, \dots, b_{n-2})$. This coordinate system uniquely defines any loop by a sequence of integers, and any even number of integers can be represented as a loop.

3.7. Generator Application

The loops encoded in Dynnikov coordinates are evolved using the generators computed from the trajectory crossings. The idea is that as generators are applied to the trajectories, the loop surrounding the trajectories also moves. The benefit of encoding a loop using Dynnikov coordinate

is that there is a set of rules for updating the these coordinates under the application of a generator. These rules allow rapid computation of change in the position of the loop due to a generator and prevent the need for a computationally intensive loop advection process. This also allows the analysis of a large number of random loops, therefore enabling the search of exponentially growing loops. The rules are discussed in detail in [11]. It is pertinent to mention that initialization of a loop is done in a random fashion, and since a loop is represented by $2N-4$ coordinates, a large number of possible loops can be generated for a N trajectory sample. However, in the current implementation only the loops with coordinate values between -1 and 1 are considered. For a single run of the flow complexity computation, K different loops are initialized and their evolutions are tracked under the application of generators.

3.8. Loop Length Computation

The next step is to measure the length of the loop for which we use the result by Moussafir [11]. This result demonstrates that the length of a loop is proportional to the number of times the loop crosses (intersects) a line connecting all the particles (green horizontal line in Fig. 7). In terms of Dynnikov coordinates, the number of intersections L with the reference line for a loop q is [8]:

$$L_q = \mid a_1 \mid + \mid a_{n-2} \mid + \sum_{i=1}^{n-3} \mid a_{i+1} - a_i \mid + \sum_{i=0}^{n-1} \mid b_i \mid, \tag{2}$$

where $b_0 = -\max_{i \leq i \leq n-2}(|a_i| + \max(b_i, 0) + \sum_{j=1}^{i-1} b_j)$ and $b_{n-1} = -b_0 - \sum_{i=1}^{n-2} b_i$.

The growth rate of $L(t)$ (dependence on q is removed for clarity) is a measure of how fast the loop is growing with time. If the rate of growth is slow or remains approximately the same, the loop is surrounding particles exhibiting simple motion. On the other hand if the length is growing at exponential rates, it surrounds particles with complex and entangled motion. The growth of the loop is measured by computing $L(t)$ in intervals of 0.5 seconds (or 15 frames).

3.9. Braid Entropy Computation

The final step is to compute the topological entropy of the braid. However, instead of directly computing the topological entropy we compute the braid entropy which is a lower bound on the topological entropy. As more trajectories are included it converges to the topological entropy of the underlying the flow (or dynamical system) [13, 10]. In general the topological entropy of a dynamical system measures the loss of information under the dynamics, and this loss happens usually in the presence of chaotic (or complex) motion. Given the growth rate of a loop, the braid entropy, S_b, is computed as [10]:

$$S_b = \lim_{t \to \infty} \frac{d}{dt} \log L(t). \tag{3}$$

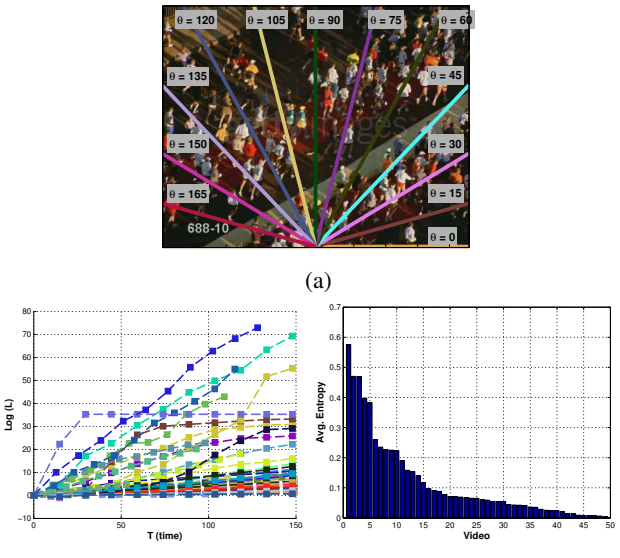
Here S_b is an approximation to braid entropy and captures the the asymptotic growth rate of the logarithm of the length of the loop $L(t)$. Note that S_b is maximized over the choice of loops, K. To obtain a single value, $L(t)$ is averaged over all M sampling runs, and slope of a line fitted to the logarithm of the averaged $L(t)$ is used as the magnitude of the braid entropy. The value also represents the topological entropy or motion complexity of the underlying flow.

4. Experiments and Discussion

4.1. Dataset & Experimental Setup

The approach is tested on a dataset depicting various types of moving objects, scenes, camera viewpoints, motion patterns and density levels. The types of objects include groups of people, vehicles, school of fish, ants and synthetic moving characters. In term of scenes, the dataset contains videos taken at street intersections, religious gatherings, sports stadium and under-water. Similarly, the camera viewpoint exhibits wide variations. In terms of motion patterns, the dataset contains dominant flows of various shapes and orientations, and flows resulting from intermingling of objects, chaotic motion, and multiple independently moving objects. In total, the dataset consists of 50 videos with each having 120 to 150 frames on average. 30 of the videos are originally from the UCF crowd dataset. The amount of variation in this dataset allows a realistic benchmark for evaluating the performance of the proposed algorithm.

For each video the optical flow is computed as described in Section 3.1. OpenCV based implementation is used with default parameters except the value of α (the smoothness term) is set to 0.01. Trajectories are obtained by advecting a dense grid of particles through the flow and $N = 50$ trajectories are sampled for further analysis. It is observed that this choice of N is reasonable to capture the dynamics of the flow in the current dataset. However, a more thorough experimentation is required to understand the dependence of N on the computed flow complexity. In total $M = 10$ sampling runs are performed to compute $L(t)$ (Eq. 2). The final value of $L(t)$ is obtained by averaging growth rate over all M runs. The braid entropy for a particular projection axis is computed by fitting a line to the logarithm of averaged $L(t)$ as described in Section 3.9. The entropy computation is repeated for 12 projection axis oriented from $\theta = 0^\circ$ to 165° with a jump of 15° in between (see Fig. 8a). Finally, the braid entropy averaged over all projection axis is used as a measure of flow complexity in a video.

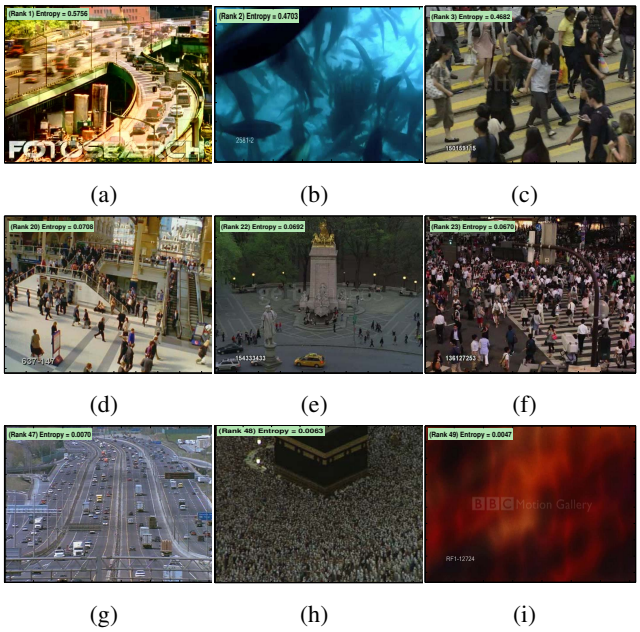


(a) (b) (c)

Figure 8: (a) The 12 projection axis ranging from $\theta = 0°$ to $165°$; (b) The growth rate, $\log L(t)$, of all videos in the dataset; (c) The computed braid entropy of each video (sorted in descending order).

4.2. Flow Complexity Computation Results

This section presents quantitative and qualitative results. Fig. 8b shows a plot of $\log L(t)$ (averaged over all 12 projection axis) as a function of time for all 50 videos in the dataset. Based on the slope of the line fitted to each of these curves, videos are categorized into three types of flow complexity based on high, medium and low growth rates of the $\log L(t)$. It is observed that the videos representing high flow complexity contain dense motion at high speeds and are also captured by a zoomed-in camera (i.e. trajectories cover greater than 80% of image space). In the dataset this corresponds to the videos of school of fish, fast moving vehicles and zoomed-in view of street crossings (Fig. 9a to Fig. 9c). This is intuitively understandable as fast moving trajectories are able to perform many more crossings per unit time thereby increasing the braiding factor. Similarly flow at a street crossing, when observed by a zoomed-in camera, becomes highly complex due to people walking in opposite directions and articulated motion of arms and legs. The medium complexity flows correspond to videos containing zoomed-out views of crowds, usually from a high angle view, where trajectories intertwine primarily due to high density of objects. However, due to the zoomed-out camera view (which translates to low resolution) trajectories travel for much shorter distances per unit time and as a result the number of crossings drop. Fig. 9d to Fig. 9f show keyframes from videos that fall in medium complexity category. Finally, videos with low complexity flows contain motion in a dominant direction (i.e. people running along a path, traffic on a road, motion of synthetic characters in one direction), or flows at extremely high density (almost packing density) where it becomes hard for particles to exchange their positions (see Fig. 9h). This is an interesting result as apparently this flow appears complicated due to the shear number of people in the scene. When there is a dominant motion, high braid entropy is observed only along few projection axis. Fig. 9g to Fig. 9i show keyframes for low complexity flow videos. The bar chart in Fig. 8c summarizes the braid entropy, which is an average of braid entropy over 12 projection axis, for all videos in the dataset.



(a) (b) (c)
(d) (e) (f)
(g) (h) (i)

Figure 9: Videos having high (top row), medium (middle row) and low (bottom row) flow complexity.

Next we discuss the effect of projection axis on flow complexity computation. For this purpose, for each projection axis percentage change in the entropy with respect to the highest entropy value along any other axis is computed. Then a mean of the percentage change is obtained over the 12 axis. The idea is that if the underlying motion is truly complex, the percentage change in entropy should be small (i.e. it is independent of the projection axis). On the other hand if the percentage change in entropy is large, it signals the existence of a motion which is complex only along certain axis. Based on this measure all videos are ranked and a keyframes from one of the highest and lowest ranked videos are shown Fig. 10. The $\log L(t)$ curves for all 12 projection axis is also displayed. It is observed that videos least effected by the choice of projection axis are the ones that usually have medium density of objects (people, ants etc.) with relatively zoomed-in views. The objects in these video move in all directions which results in a large number of crossings along all projection axis. On the other hand,

videos depicting flows with highest amount of percentage change in braid entropy contain flows in a dominant direction (e.g. marathon, vehicles on a road). For example, the video in Fig. 10c has one of the highest percentage change in the braid entropy, with the highest entropy of 0.1 along the projection axis oriented at 60° (see Fig. 8a) i.e., axis slightly slanted with respect to the dominant direction of motion, and the lowest entropy of 0.021 along the axis oriented at 150° (i.e. almost perpendicular to the dominant direction of motion).

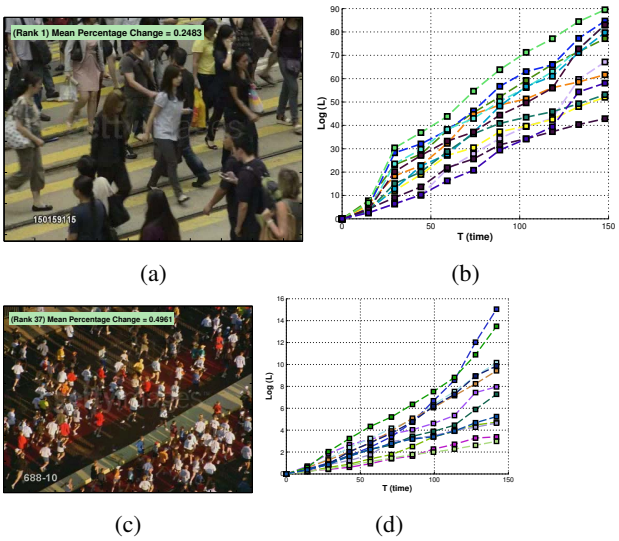


Figure 10: Videos having low (top row) and high mean percentage change in braid entropy with respect to the projection axis.

Finally, Fig. 11 shows the flow complexity based ordering of videos displayed in Fig. 1.

Figure 11: Ordering in terms of complexity of flow (left more complex - right less complex).

5. Conclusion

In this paper, a trajectory based formulation for flow complexity is proposed. This formulation explicitly takes into consideration object *interactions* and demonstrates that braids resulting from intermingling of trajectories can be used to compute the level of flow complexity in video sequences. The future work can take a number of possible directions. For instance, dependence of flow complexity on various design parameters (e.g. number of trajectories, temporal scales, projection axis etc.) can be further explored and validated. In addition, the trajectory representation in combination with braid theory can be used to address several other problem areas that include action recognition, motion segmentation and abnormal behavior detection.

References

[1] Chen-Yu Chen et. al, *Motion Entropy Feature and its Applications to Event-based Segmentation of Sports Video*, EURASIP J. Adv. Signal Process, 2008. 2

[2] Y. F. Ma, and H. J. Zhang, *A New Perceived Motion-based Shot Content Representation*, IEEE ICIP, 2001. 2

[3] S. Suthaharan et. al, *Perceptually Tuned Video Watermarking Scheme Using Motion Entropy Masking*, IEEE Region 10th Conference and Exhibition, vol. 1, 1999. 2

[4] K. A. Peker, A. Divakaran, T. V. Papathomas, *Automatic Measurement of Intensity of Motion Activity of Video Segments*, Proc. SPIE Storage and Retrieval for Media Databases, 2001. 2

[5] S. Jeannin and A. Divakaran, *MPEG-7 Visual Motion Descriptors*, IEEE Transactions on Circuits and Systems for Video Technology, vol. 11, pp. 720-724, 2001. 2

[6] T. Liu et al., *A Novel Video Keyframe Extraction Algorithm based on Perceived Motion Energy Model*, IEEE Transactions on Circuits and Systems for Video Technology, 13(10), 2003. 2

[7] G. Iyengar, and A. Lippman, *VideoBook: An Experiment in Characterization of Video*, IEEE ICIP, 1996. 2

[8] J.-L Thiffeault, *Braids of Entangled Particle Trajectories*, Chaos: An Interdisciplinary Journal of Nonlinear Science, 20(1), 2010. 2, 4, 5, 6

[9] M. R. Allshouse and J.-L. Thiffeault, *Detecting Coherent Structures Using Braids*, Physica D, 241(2), 2012. 4, 5

[10] J. G. Puckett et al., *Trajectory Entanglement in Dense Granular Materials*, Journal of Statistical Mechanics: Theory and Experiment, 2012. 6

[11] J. O. Moussafir, *On Computing the Entropy of Braids*, Func. Anal. and Other Math., 1(1), 37-46, 2006. 2, 6

[12] I. A. Dynnikov, *On a Yang-Baxter Map and the Dhornoy Ordering*, Russian Math Survey, 57, 592-594, 2002. 5

[13] S. M. Cox and M. D. Finn, *Two Dimensional Stokes Flow Driven by Elliptical Paddles*, Physics of Fluids, 19(11), 2007. 6

[14] Thomas Brox and J. Malik, Large Displacement Optical Flow: Descriptor Matching in Variational Motion Estimation, IEEE Trans. on PAMI, 33(3), 2011. 1

[15] Saad Ali and Mubarak Shah, *Floor Fields for Tracking in High Density Crowd Scenes*, In ECCV 2008. 1, 2

[16] Louis Kratz and Ko Nishino, *Tracking Pedestrians Using Local Spatio-Temporal Motion Patterns in Extremely Crowded Scenes*, IEEE Trans. on PAMI, 34(5), 2012. 1

[17] B. Zhou et al., *Understanding Collective Crowd Behaviors: Learning a Mixture Model of Dynamic Pedestrian-Agents*, In IEEE CVPR, 2012. 1, 2

[18] L. Xu et al., *Motion Detail Preserving Optical Flow Estimation*, IEEE Trans. on PAMI, 2012. 1, 3

[19] J. Yan et al., *A General Framework for Motion Segmentation: Independent, Articulated, Rigid, Non-rigid, Degenerate and Nondegenerate*, In ECCV, 2006. 1

[20] N. Vasconcelos and A. Lippman, *Empirical Bayesian Motion Segmentation*, IEEE Trans. on PAMI, 23(2), 2001. 1

[21] Thomas Brox and J. Malik, *Object Segmentation by Long Analysis of Point Trajectories*, In ECCV, 2010. 1

[22] I. Saleemi et al., *Scene Understanding by Statistical Modeling of Motion Patterns*, In IEEE CVPR, 2010. 1

[23] D. Lin et al., *Modeling and Estimating Persistent Motion with Geometric Flows*, In IEEE CVPR, 2010. 1, 2

[24] Saad Ali and Mubarak Shah, *A Lagrangian Particle Dynamics Approach for Crowd Flow Segmentation and Stability Analysis*, In IEEE CVPR, 2007. 2, 3

[25] B. E. Moore et al., *Visual Crowd Surveillance through a Hydrodynamics Lens*, In Commun. ACM, 54(12), 2011. 3

[26] T. Brox et al.,*High Accuracy Optical Flow Estimation based on a Theory for Warping*, In ECCV 2004. 3