Multi-Agent Behaviour Segmentation via Spectral Clustering

Bálint Takács and **Simon Butler** and **Yiannis Demiris**
Intelligent Systems and Networks Group
Electrical and Electronic Engineering
Imperial College
South Kensington Campus London SW7 2AZ

Abstract

We examine the application of spectral clustering for breaking up the behaviour of a multi-agent system in space and time into smaller, independent elements. We extend the clustering into the temporal domain and propose a novel similarity measure, which is shown to possess desirable temporal properties when clustering multi-agent behaviour. We also propose a technique to add knowledge about events of multi-agent interaction with different importance. We apply spectral clustering with this measure for analysing behaviour in a strategic game.

Introduction

Segmenting complex joint behaviours of multi-agent systems into smaller independent elements is of utmost importance since such segmentation can be used for automatic plan extraction, plan recognition or subgoal selection in learning tasks.

In most cases, the behaviour of a multi-agent system is built up from independent, smaller elements. For example, consider the RoboCup domain: if a goal was scored, we could determine which agents participated in performing the goal through simple rules, for example, examining single indicators of interaction (like who touched the ball prior to the goal). We can then effectively break up a match into smaller parts. The problem is obviously much more complicated: a single agent generally can participate in multiple consequent elements in the same match, or can fill multiple roles in two parallel ongoing elements. Examining single indicators of interaction is not enough to separate these elements, because not all interactions tie agents together (for example, an agent may see all the other agents from a distance, but usually interacts only with the nearest ones). It means we need to identify temporal and spatial boundaries between these elements in a very noisy environment. We also need to consider *scaling* of the separation: a specific agent or agent group may affect other agents only for a short time and in a specific role, and it is possible that no further description is required about the internal structure of the agent group.

These problems are crucial in analysing multi-agent system behaviour in general and are strongly connected to plan recognition (see e.g. (Devaney & Ram 1998; Nair *et al.* 2004; Sukthankar & Sycara 2006) and the references therein). Plans are different from the elements we are trying to identify here: a plan is a higher-level concept which may span multiple element of the joint behaviour. Nevertheless, elements can be considered as low-level plans which were extracted automatically.

A major question here is how to define independence between elements. One may think in terms of probability theory, where methods do exist to deal with noisy environments. For example, "blind" statistical algorithms, like independent component analysis, are capable of separation by probabilistic independence without a known model. However, we think that measuring event densities and thus probability-based separation cannot be applied in most of the cases, because estimating probabilistic models require lots of samples usually not available.

An alternative approach is to use *clustering* (Xu & Wunsch 2005). The goal of clustering is to separate a finite unlabeled data set into a finite and discrete labelled set of "natural", hidden data structures. We may try using clustering algorithms to label agents or group of agents in time and space, and then view the cluster boundaries as spatio-temporally separated elements. Clustering is an ill-posed problem from the mathematical point of view, because what is "natural" is usually decided by human judgement as no labelled data available, but despite of this, clustering algorithms are frequently applied to gain insight to the structure of data.

Multi-agent systems are frequently operate in the two-dimensional spatial domain, like the already mentioned RoboCup. In these domains, agent positions form a set of two dimensional points. An emerging approach to clustering is *spectral clustering* (Shi & Malik 2000; Ng, Jordan, & Weiss 2001; Bach & Jordan 2006), which is shown to be able to catch human impression about grouping two-dimensional point formations very efficiently (Shi & Malik 2000; Zelnik-Manor & Perona 2004). Spectral clustering is also appealing because it is related to manifold learning (Lafon, Keller, & Coifman 2006; Ham, Ahn, & Lee 2006), which is about finding low-dimensional representations of data along geometrical constraints. We may expect that two dimensional multi-agent behaviour is subject to such constraints and therefore manifold learning may be able to reduce dimensions effectively.

Clustering methods require an *affinity* (or similarity) measure over the points to be clustered, which is a local measure telling the expectations about whether two point should be labelled similarly or not. Affinity measures are usually a function of the distance between the points (here distance not always equivalent with the Euclidean metric, see e.g. (Xu & Wunsch 2005)). If u and v are points in the full set V, the point-to-point affinities form the $N \times N$ sized $w(u, v)$ affinity matrix if we have N points to cluster.

Spectral clustering concerns the minimal *normalised cut* of the affinity graph, which is the graph corresponding to the affinity matrix with edge weights defined between points u and v as $w(u, v)$ (Shi & Malik 2000). If the set of graph nodes, noted by V, are to be cut into two separated sets A and B, the normalised cut is defined as follows:

$$Ncut(A, B) = \frac{assoc(A, B)}{assoc(A, V)} + \frac{assoc(A, B)}{assoc(B, V)} \quad (1)$$

where

$$assoc(A, B) = \sum_{u \in A, v \in B} w(u, v). \quad (2)$$

Since $V = A \bigcup B$ and $A \bigcap B = \emptyset$, $assoc(A, V) = assoc(A, B) + assoc(A, A)$. In other words, normalised cuts are trying to find cuts of the graph where the $assoc(A, B)$ cut value is minimal, but the $assoc(A, A)$ intra-cluster affinities of the two sets are maximal. Normalised cuts are superior to minimum cut approaches because they take into consideration the intra-set dependencies as well.

It was shown that the eigenvector decomposition of the affinity matrix can be used for finding approximate solutions to the minimal normalised cut problem, which is NP-complete (Shi & Malik 2000). This is why this method is called spectral clustering. As explained by Markov random walks (Lafon, Keller, & Coifman 2006), when one considers only the first few eigenvectors of this decomposition, the number of eigenvectors retained can be seen as a scaling parameter to the clustering, thus we can hope to treat the scaling problem as well. Other recent works showed that spectral clustering is deeply connected to non-linear dimensionality reduction methods like ISOMAP or locally linear embedding (see (Saul *et al.* 2006) for a review) or kernel PCA methods (Bengio *et al.* 2004), and can be seen as a method searching for block-diagonal groupings of the affinity matrix (Fischer & Poland 2005).

Finding the Affinity Measure

Based on the convincing perceptual grouping performance of the algorithm, spectral clustering is a good start for finding spatial groupings in multi-agent systems. However, it is still an open question how to extend the clustering into the temporal domain.

Defining what to cluster is the first step. By a simple extension of the two-dimensional problem space of clustering agent positions into the temporal domain, we get that each point is defined in the form of $u : (x, y, t)$, where t is its owner agent's time spent from the beginning of the engagement, and x, y its 2D coordinates at that time. We will call a normalised cut found by the clustering *temporal* if the cut includes edges that only connect nodes which have different temporal value (but belong to the same agent), and *spatial* if the cut has edges which do belong to different agents at the same time. Let assume that we record an agent's position with a prescribed frequency, for example, we measure it once in each second and the time is measured by the number of these „ticks" spent. This also means that we record discrete times but continuous spatial positions, which is usually the case in real-life problems.

Defining an affinity measure over these points is not obvious. The original spectral clustering approaches typically use Gaussian affinities (Shi & Malik 2000; Zelnik-Manor & Perona 2004):

$$w(u, v) = \exp\left(-\frac{d(u, v)^2}{\sigma^2}\right) \quad (3)$$

where $d(u, v)$ is the Euclidean distance between points u and v and σ is a scaling parameter.

The most straightforward extension of this affinity measure to the spatio-temporal domain is if we extend the Euclidean distance to the temporal dimension as well. However, this kind of extension has strange properties: a single agent standing still will have the temporal cut value decreasing to zero as time progresses. It is easy to see if we consider that in this case the graph of the points to be clustered forms a simple chain in time. The minimal normalised cut is in the middle of the chain. If we extend the chain on the two ends, all extra nodes will contribute to the intra-cluster edge weights, but they will contribute almost nothing to the cut weight because the value of a Gaussian far from its center is almost zero. It means any cut involving the single, standing agent will prefer only temporal cuts as time progresses, which is not a desirable property of this affinity measure. If we wait long enough, any normalised cut which involves this agent will unavoidably split the agent's history in time and never in space (or, if we repeat cutting to get more clusters, the first few cuts will be always temporal, at least until the temporal length of the segments become small enough). In other words, affinities good for spatial clustering are not giving the expected result in time.

It looks like we have to modify the affinity measures for the purpose of clustering spatio-temporal multi-agent behaviour. We may try weighting in the different modalities (space or time), but it could introduce a lot of parameters. To reduce the number of parameters, we should start with defining sensible constraints on the measure, like

1. the measure should be local,
2. it should be equivalent to the affinity measure of Eq. (3) when there is no temporal difference between two points,
3. if $u : (x_1, y_1, t_1)$ and $v : (x_2, y_2, t_2)$, and we define $\Delta\mathbf{x} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$ and $\Delta t = |t_1 - t_2|$, it should be monotonically decreasing with increasing $\Delta\mathbf{x}$

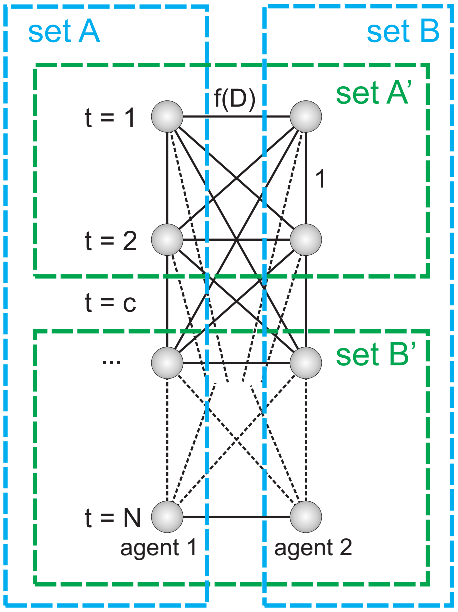


Figure 1: **Two-agents-standing scenario.** This is the (full) affinity graph that illustrates the case where two agents are standing at D distance from each other. We assume that the agents' position is measured N times in equal intervals. The two sets belonging to spatial cuts are marked by A and B, while the sets marked by A' and B' in the case of temporal cuts. The $t = c$ parameter is the time index of the boundary between sets A' and B'. The temporal cut is balanced if $c = N/2$ (assuming N is even). The shown edge values $f(D)$ and 1 are of the proposed measure of Eq. (10) if we set the time spent between two measures to 1.

and Δt, that is, points with increasing distance should be less similar,

4. it should be translation invariant and symmetric in time as well, that is, $w(u, v)$ should depend only on the temporal difference Δt (and not directly on t_1 or t_2),
5. it should be robust against time spent in the case of "standing" agents. One possible way to concretise this is the following: normalised cut values for standing agents should not depend on the total time of the graph neither in the case of spatial nor temporal cuts.

A possible way to provide theoretical analysis of the arising affinity measures is to examine simple scenarios like the single agent case was, and select parameters of possible measures according to the expected behaviour in these cases. The next simplest case is when two agents standing at D distance from each other (Figure 1). This case can be seen as an approximation to the general case, where we consider each (closest) pair of agents individually and neglecting all the effects of other agents.

In the case of the two-agents-standing scenario, if we search for the affinity measure in the form of

$$w(u, v) = f(\Delta \mathbf{x}) \cdot g(\Delta t), \tag{4}$$

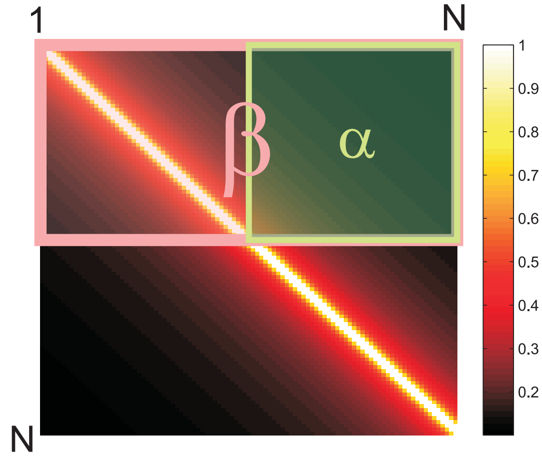


Figure 2: **The affinity matrix of the one-agent-standing case.** The example here shows the affinity values of $g(\Delta t) = \Delta t^{-1/2}$. Diagonal values were set to 1. The sum of the affinities below the area marked by α equals to $assoc(A', B')$ and the sum of the affinities below the rectangle marked by β equals to $assoc(A', V)$.

the spatial normalised cut values will not depend on the total time spent at all. That is because after substituting Eq. (4) into Eq. (2) and inserting it into Eq. (1), we get that $Ncut(A, B)$ does not depend on $g(\Delta t)$ and

$$Ncut(A, B) = 2f(D)/(1 + f(D)). \tag{5}$$

Here A and B are the two sets of a *spatial* cut (Fig. 1) and D is the distance of the agents. By equating $f(\Delta \mathbf{x})$ with the right-hand side of Eq. (3), that is, by defining

$$f(\Delta \mathbf{x}) := \exp(-\frac{\Delta \mathbf{x}^2}{\sigma^2}), \tag{6}$$

we have $f(D)$ changing between 0 and 1, which in turn means that the normalised spatial cut value changes between 0 and 1, and is equal to zero only if the points are totally spatially dissimilar (they are infinitely far away). That means that spatial cuts will fulfill our last requirement.

The values of temporal cuts are harder to calculate in an explicit form. First, we may exploit the fact that by assuming a form of the measure as in Eq. (4), the temporal cut values of the two-agents-standing scenario are equivalent to the values of temporal cuts in the case when one agent is standing. It again can be seen from substituting Eq. (4) into Eq. (2) and inserting it into Eq. (1).

Let assume that the temporal cut is balanced in the one agent case, that is, it prefers cutting N points into two parts consisting $N/2$ points each (later we will check whether this assumption really holds). Let us try the following temporal measure: $g(\Delta t) = \Delta t^{-r}$, where r is a constant, and define $g(\Delta t) := 1$ if $\Delta t = 0$. It is obvious that this measure fulfills our requirement above for decreasing affinity by increasing temporal distance. The normalised cut value then can be approximated by 2x the sum of elements of the affinity matrix

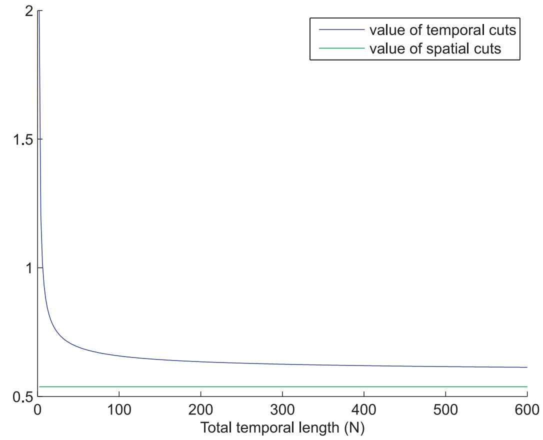


Figure 3: **Cut values as a function of N in the case of the two-agents-standing scenario with $D = 1$ and $\sigma = 1$.** As can be seen, the value of spatial cuts is constant, while the value of temporal cuts converge rapidly with time. By tuning the σ parameter, we can determine the smallest distance where we start prefer spatial instead of temporal cuts.

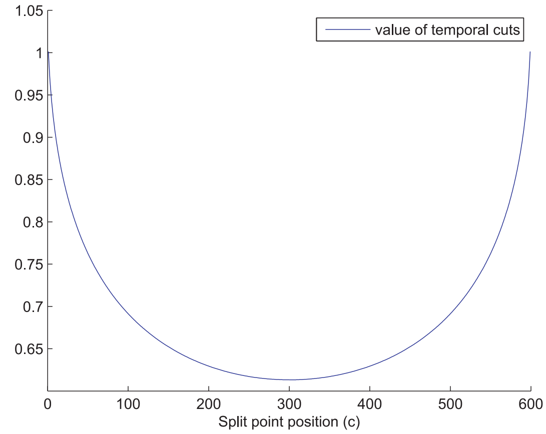


Figure 4: **The temporal cut value as the function of the cutting point in the case of the two-agents-standing scenario with $D = 1$ and $\sigma = 1$.** Here N = 600 was fixed and the impact of c cutting point of Fig. 1 was examined with the affinity measure of Eq. (10). The measure prefers balanced cuts, that is, cuts which split the temporal domain into two same-sized parts.

in the upper right quadrant divided by the sum of elements in the upper half of the matrix (see Fig. 2). We may assume without the loss of generality that N is even, so these sums can be written as

$$assoc(A', B') = \sum_{i=0}^{N/2-1} \sum_{j=N/2-i}^{N-i-1} j^{-r} \tag{7}$$

$$assoc(A', V) = assoc(B', V) =$$
$$= assoc(A', B') + \frac{N}{2} + 2 \sum_{i=1}^{N/2-1} \sum_{j=1}^{N/2-i} j^{-r}, \tag{8}$$

where A' and B' are the two sets of the *temporal* cut (Fig. 1). These sums can be substituted with integrals as N increases, and Eq. (1) can be approximately calculated in an explicit form for different r values in the case of the two-agents-standing scenario. It turns out that for $r \geq 1$ integers, $\lim_{N\to\infty} Ncut(A', B') = 0$ (although at $r = 1$, the cut value decreases very slowly for all practical N values). For $r = 1/k$ where $k = 2, 3, \cdots$, after tedious but elementary calculations one can see that

$$\lim_{N\to\infty} Ncut(A', B') = 2 - 2^{1/k}. \tag{9}$$

For example, in the case of $k = 2$, $Ncut(A', B') > 0.585$ for all N values (see Fig. 3). Higher k values result in a faster convergence, but time differences also have a lower impact on cut values.

The convergence of temporal cut values means that for sufficiently big N values, temporal cuts are practically independent of the total time spent. Because temporal cut values are bounded from below, we can set up the spatial cut values in a way that their value compared to temporal cuts depends on the D distance between the agents only. Thus the prerequisite for being clustered in the same group because of spatial distance is determined solely by the σ parameter, that is, we can set σ to determine a minimum distance where two agents never will be clustered into the same group, no matter how much time was spent. We can conclude that the combined measure approximately fulfills the last requirement in the case of two agents.

To summarise, the proposed affinity measure for spatio-temporal spectral clustering of agent behaviour is as follows:

$$w(u, v) = \begin{cases} \frac{1}{\sqrt{\Delta t}} \exp\left(-\frac{\Delta \mathbf{x}^2}{\sigma^2}\right) & \text{if } \Delta t \neq 0, \\ 1 & \text{otherwise.} \end{cases} \tag{10}$$

The single parameter of this measure is σ, just like it was in the case of the original affinity measure. Figure 3 shows the cut values with this measure in the case of the two-agents-standing scenario as a function of N. Based on this and the analysis given in this section, we can therefore give a natural interpretation to σ as – if there are two agents and no special events present – being about two times the distance between the agents where the algorithm starts to prefer spatial cuts instead of temporal ones. This means agents are thought to be far away enough to be clustered as a separate group at distances of about $\sigma/2$.

We still have to check that the affinity measure of Eq. (10) prefers balanced temporal cuts, that is, it has the smallest cut value at $c = N/2$ where c is the cut point of Figure 1 (we can assume without the loss of generality that N is even), or, in other words, a temporal cut is smallest when we cut a temporally symmetric graph into two equally sized parts. It can be justified by a simple numerical test (Figure 4) or in a more rigorous way, via elementary calculus with relaxing c to allow taking continuous values and taking the derivative of $Ncut(A', B')$ with respect to c.

Adding Events

It is obvious that using only agent positions is not enough for catching the behavioural aspects of the multi-agent system. Fortunately, we may assume that a log of *events* is also available, which can be seen as a history of low-level interactions. We may assume that each event has a relative importance measure and we know the participants of that event. For example, in the case of a strategic computer game, we may assume that we can record when a unit fires at an enemy and who this enemy was. Lots of similar low-level events are possible to define, and a human expert can set a relative importance to each of these events in quite a natural way. It is advantageous to treat positional information like events as well, which occur with a fixed frequency. These positional events can have much lower importance than events covering some important interaction.

Adding events with different importance can be done by exploiting the fact that with spectral clustering the affinity measure is only required to be symmetric and positive. We also know that affinities created by Eq. (10) are falling between 0 and 1, therefore we can think of simple spatial connections as having a value of 1 when scaling event importance. When adding an event for two agents into the affinity matrix, we do not have to introduce new nodes: we may calculate the closest temporal representation of both agents and set the affinity between these nodes to the event importance. It is reasonable to add a temporal "depth" to the event as well, which means setting the affinities to the agents' next and previous nodes to the importance as well. If we need to superimpose multiple events, we can simply add their importance measures.

When events are available, instead of clustering spatio-temporal positions of each agent, one may consider clustering in the space of events with combined spatial positions of the participants and its time. This choice ends up with more dimensions, which may cause problems for clustering (Xu & Wunsch 2005). This option also has the problem that it is not too easy to incorporate event importance. Multiplying the number of events by their importance is possible, however, it may rapidly increase the number of points. Additionally, this clustering is not equivalent to two dimensional clustering of agent positions when neglecting the temporal dimension.

Experiments

We tested our ideas on a multi-agent system developed to simulate small-scale conflict of armoured units. The program can be compared to real-time strategy games in computer entertainment, where human operators can move the units according to their commands. Two players and only one type of unit was present, which were modelled to the extent of movement and firing of the main gun. A very simple damage model was employed, where the actual unit damage status is described by a scalar value, and the unit is destroyed

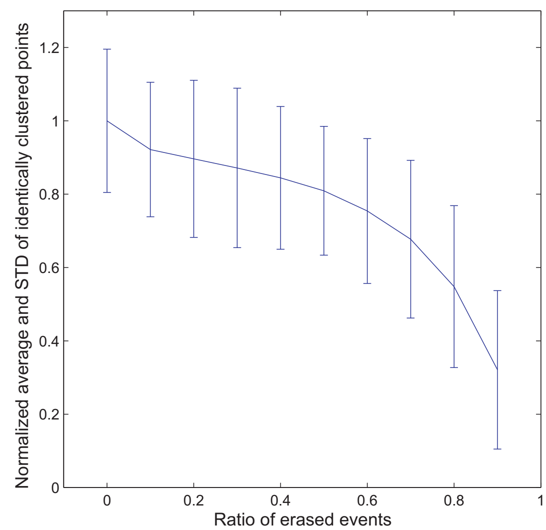
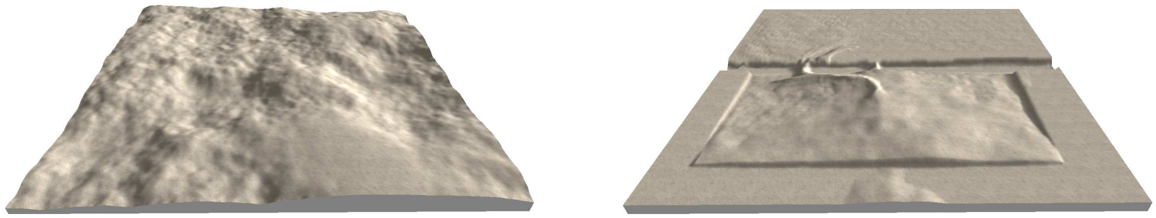


Figure 6: **The stability of the results as a function of the ratio of events deleted from the history.** The first scenario of Fig. 7 was used. We gradually increased the amount of data deleted from the history and calculated the number of differently clustered points, ten times each. The curve shows the measured average deviance and the standard deviations from the case when all the events were retained. In each case, the comparison is made in the "fairest" cluster assignment (we considered that the algorithm may output clusters in different order).

if this indicator reaches zero. The amount of damage dealt with a hit was dependent on the accuracy of the hit. We modeled the unit movement and behaviour by real physical dynamical modeling. The environment was described with a two-dimensional heightmap. This approach is satisfying to depict the most important aspects of the game from the planning point of view, like entrenchment, open ranges and line-of-sight. The goal of the engagement was to destroy all units of the enemy. The units were initially placed in pre-defined positions. Units fired automatically on the nearest target when it was available. Human operators were able to select target positions for movement for each unit independently. To ease operation by humans, the simulation accepted commands in a "paused" state when the flow of time was frozen, thus a human operator is able to substitute multiple human operators, thus effectively modelling an all human-controlled multi-agent system. The software is based on the Delta3D engine (http://www.delta3d.org/) and is open-source and multi-platform.

We created multiple maps (two of these can be seen on Figure 5) and played battles on these maps to produce multi-agent behavioural data. We used the spatio-temporal affinity measure proposed in the previous section to create the affinity matrix. σ was set to 200 and spatio-temporal points were assigned to each unit by 1 update/sec frequency. We defined 4 events: 1. SEE: a unit becomes visible to an other one, 2. HIDE: a unit becomes invisible to an other one, 3. FIRE: a unit fires, 4. HURT: a unit's health is decreased by a hit.



(a) Map 1 **(b) Map 2**

Figure 5: **Two example maps created for testing purposes.** The first map was created by a random landscape generator. The second one has a separating gorge across the map, which is crossed by two "bridges". There are potential entrenchment sites on both sides of the gorge. Both maps are 5 km $\times$ 5 km in size.

We assigned an importance of 20 to SEE and HIDE events, while FIRE/HURT got 40.

After the normalisation step first described by (Shi & Malik 2000), we created a 15 dimensional eigenvector projection retaining the largest eigenvalues (Shi & Malik 2000). We then used the algorithm of (Zelnik-Manor & Perona 2004) to automatically determine the number of clusters. This method works by performing rotations of the matrix of eigenvectors by defining a cost function which penalises configurations which do not conform to the expected block-diagonal form. The cost function can be used to measure the fitness of clustering when different number of clusters are selected, thus automatic selection of the number of clusters is possible.

Results

As mentioned in the introduction, the unsupervised nature of clustering makes hard to validate the results. In some cases, even humans will not agree how to split up engagements. Therefore we created scenarios where the behaviour was governed by simple plans, and compared the result of the algorithm on these scenarios with our original intentions.

Figure 7 shows two results on the maps of Fig. 5. 5 blue and 4 red units were placed on the first map. 4 blue units were grouped initially in the upper left corner, 2 red units were placed in the upper right corner, a red unit was placed into the lower right corner and a red and a blue unit was placed in the lower left corner. The four blue units first engaged the two reds in the upper half of the scene, which resulted in the destruction of both red units and the corresponding loss of a single blue unit. Meanwhile, the red unit in the lower right took a defensive position and the single red and blue units tried to destroy each other in a duel. The three blue units took a right turn and destroyed the defending red one, and joined the last blue unit to destroy the last red one.

The scenario had 4399 points to cluster in total. The algorithm found that 3 clusters are optimal. It can be seen that the main elements of this description are captured by the clustering algorithm by separating the elimination of the two red units, the elimination of the single red unit and finally the endgame. The output demonstrates that the algorithm is capable of differentiating between parallel ongoing elements, because the temporal length of the yellow segment is equal to the length of the whole battle.

On the second map, 7 blue and 8 red units were placed in the two opposing sides of the gorge. The red units started behind the ridge of the southern side, so initially they were behind cover. The blue units took reinforced positions behind small hills and in depressions on the northern side of the gorge. The red units first tried to destroy the blue ones with a direct frontal attack which resulted in the loss of two red units and no blue units. Three red units then tried to get into the back of the defending blues with a flanking manoeuvre using the left-hand side "bridge" over the gorge, while the remaining 3 units tried to divert the blue units' attention. Finally, red tried to attack from two sides simultaneously.

The scenario had 4038 measure points. The algorithm split the engagement into two, which correspond to the main engagement and the flanking manoeuvre. We can also notice that the clustering of the final points belonging to the flanking red units is noisy. A smoothing post-processing phase which prohibits clusters to be changed frequently may help to draw more stable boundaries between clusters.

We also tested the algorithm's robustness against changes in the history of events. We ran the algorithm on the first scenario while randomly chosen events were erased from the history with a gradually increasing proportion of all events. The calculations were done ten times for each ratio applied. The results are summarised on Fig. 6. The number of points clustered equivalently to the case when the full history is used deteriorates significantly only when $> 60\%$ of the events were deleted.

Discussion

Regarding computational complexity, the algorithm's hardest part is the eigenvector decomposition, which takes $\mathcal{O}(n^3)$ time and $\mathcal{O}(n^2)$ space in the most general case, where n is the number of points to be clustered. If we can approximate affinities with sparse structures, the spectral clustering algorithm can be kept linear in the number of points to be clustered, as explained in (Shi & Malik 2000). The results shown on Fig. 7 took less than 1 minute on an average desktop computer to compute with MATLAB.

Spectral clustering has been applied previously for improving learning systems by subgoal selection (Mahadevan & Maggioni 2006; Şimşek, Wolfe, & Barto 2005; Zivkovic, Bakker, & Krse 2006), however, all of these works addressed

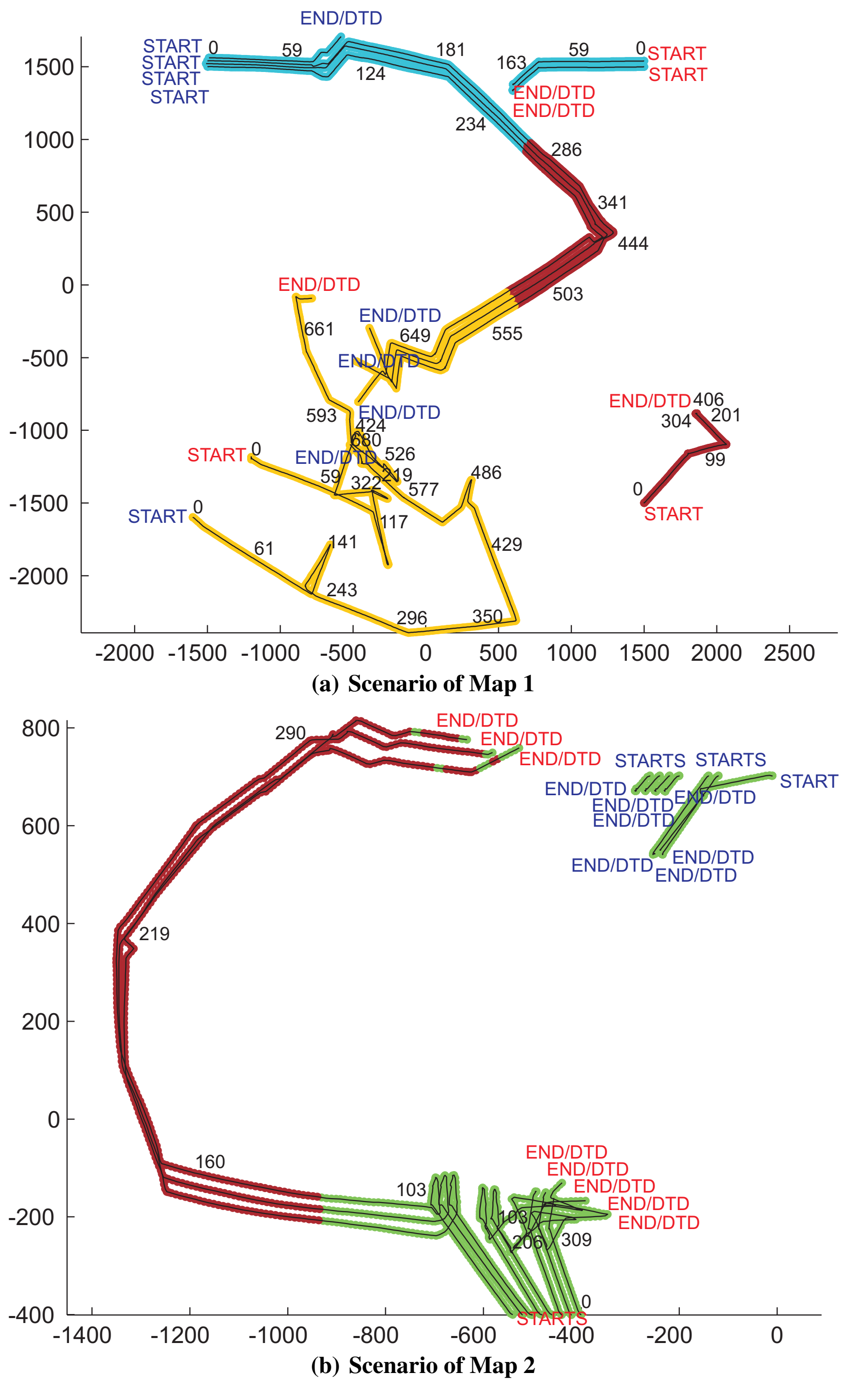


Figure 7: **Two example results of the segmentation algorithm.** The figures depicts a top-down view of the maps. Unit trajectories in space are marked by a continuous line and starting at the point marked by "START(S)" and ended or destroyed at the point marked by "END/DTD". These labels has a color appropriate to the side of the unit (red or blue). The numbers on the trajectories are the time spent for the trajectory point nearby since the start of the engagement in seconds. Colored points placed on unit trajectories are marking different clusters found by the algorithm. Distance is shown in metres; the point $(0, 0)$ corresponds to the centre of the maps shown on Fig. 5.

only the spatial and not the temporal domain. There exists works which use spectral clustering (and related methods) for grouping spatio-temporal actions (Yuan, Zhang, & Lin 2005; Park, Zha, & Kasturi 2004; Jenkins & Mataric 2004; Porikli 2005; Zelnik-Manor & Irani 2006), but all of these are applied to different problem domains, like video segmentation, and therefore do not consider the requirements outlined in the previous sections. For example, (Zelnik-Manor & Irani 2006) clusters the points of spatio-temporal gradients of video sequences. This method is not applicable in our case because we would like to keep the excellent grouping performance of spectral clustering over static positions as well, while extending it into the temporal domain at the same time.

The proposed spatio-temporal measure can be enhanced in lots of ways. A possible extension could, instead of modifying the affinity values directly, adjust the local scaling of the affinities to incorporate events into the affinities, similarly as performed in (Zelnik-Manor & Perona 2004) for spectral clustering over homogenous dimensions.

Conclusions

We proposed a novel affinity measure to extend spectral clustering into the temporal domain for automatical segmentation of multi-agent behaviour. The measure is equivalent to the well-known Gaussian affinities in the case of static problems, but shown to be superior for classifying multi-agent behaviour when the problem space extends to the temporal domain. We also proposed a technique to incorporate events with different importance. The ideas were demonstrated with segmentating multi-agent behaviour in a strategic game, where we found that the output of the algorithm coincides with the human-provided segmentations. The technique can be used in analysing multi-agent behaviour, for automatic subgoal extraction or may help with plan extraction or recognition.

References

Bach, F. R., and Jordan, M. I. 2006. Learning spectral clustering, with application to speech separation. *Journal of Machine Learning Research* 7:1963–2001.

Bengio, Y.; Delalleau, O.; Le Roux, N.; Paiement, J.-F.; Vincent, P.; and Ouimet, M. 2004. Learning eigenfunctions links spectral embedding and kernel PCA. *Neural Computation* 16(10):2197–2219.

Devaney, M., and Ram, A. 1998. Needles in a haystack: Plan recognition in large spatial domains involving multiple agents. In *AAAI/IAAI*, 942–947.

Fischer, I., and Poland, J. 2005. Amplifying the block matrix structure for spectral clustering. In *Proceedings of the 14th Annual Machine Learning Conference of Belgium and the Netherlands*, 21–28.

Ham, J.; Ahn, I.; and Lee, D. 2006. Learning a manifold-constrained map between image sets: applications to matching and pose estimation. In *CVPR06*.

Jenkins, O. C., and Mataric, M. J. 2004. A spatio-temporal extension to isomap nonlinear dimension reduction. In *Proceedings ICML*, 441–448.

Lafon, S.; Keller, Y.; and Coifman, R. R. 2006. Data fusion and multicue data matching by diffusion maps. *IEEE Transactions on pattern analysis and machine intelligence* 28(11):1784–1797.

Mahadevan, S., and Maggioni, M. 2006. Proto-value functions: A Laplacian framework for learning representation and control in Markov decision processes. Technical Report 2006-35.

Nair, R.; Tambe, M.; Marsella, S.; and Raines, T. 2004. Automated assistants for analyzing team behaviors. *Autonomous Agents and Multi-Agent Systems* 8(1):69–111.

Ng, A.; Jordan, M.; and Weiss, Y. 2001. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, volume 14.

Park, J.; Zha, H.; and Kasturi, R. 2004. Spectral clustering for robust motion segmentation. In Pajdla, T., and Matas, J., eds., *ECCV 2004*, number 3024 in LNCS, 390–401. Springer-Verlag Berlin, Heidelberg.

Porikli, F. 2005. Ambiguity detection by fusion and conformity: a spectral clustering approach. In *International Conference on Integration of Knowledge Intensive Multi-Agent Systems*, 366–372.

Saul, L. K.; Weinberger, K. Q.; Sha, F.; Ham, J.; and Lee, D. D. 2006. *Semisupervised Learning*. MIT Press: Cambridge, MA. chapter "Spectral methods for dimensionality reduction".

Shi, J., and Malik, J. 2000. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22(8):888–905.

Şimşek, Ö.; Wolfe, A. P.; and Barto, A. G. 2005. Identifying useful subgoals in reinforcement learning by local graph partitioning. In *Proceedings of the Twenty-Second International Conference on Machine Learning*, volume 119 of *ACM International Conference Proceeding Series*, 816–823.

Sukthankar, G., and Sycara, K. 2006. Robust recognition of physical team behaviors using spatio-temporal models. In *Proceedings of AAMAS*, 638–645.

Xu, R., and Wunsch, D. 2005. Survey of clustering algorithms. *IEEE Transactions on Neural Networks* 16(3):645–678.

Yuan, J.; Zhang, B.; and Lin, F. 2005. Graph partition model for robust temporal data segmentation. In Ho, T.; Cheung, D.; and Liu, H., eds., *PAKDD 2005*, number 3518 in LNAI, 758–763. Springer-Verlag Berlin, Heidelberg.

Zelnik-Manor, L., and Irani, M. 2006. Statistical analysis of dynamic actions. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 28(9):1530–1535.

Zelnik-Manor, L., and Perona, P. 2004. Self-tuning spectral clustering. Eighteenth Annual Conference on NIPS.

Zivkovic, Z.; Bakker, B.; and Krse, B. 2006. Hierarchical map building and planning based on graph partitioning. In *IEEE International Conference on Robotics and Automation*, 803–809.