Chapter 6

Representation and Recognition within QDL

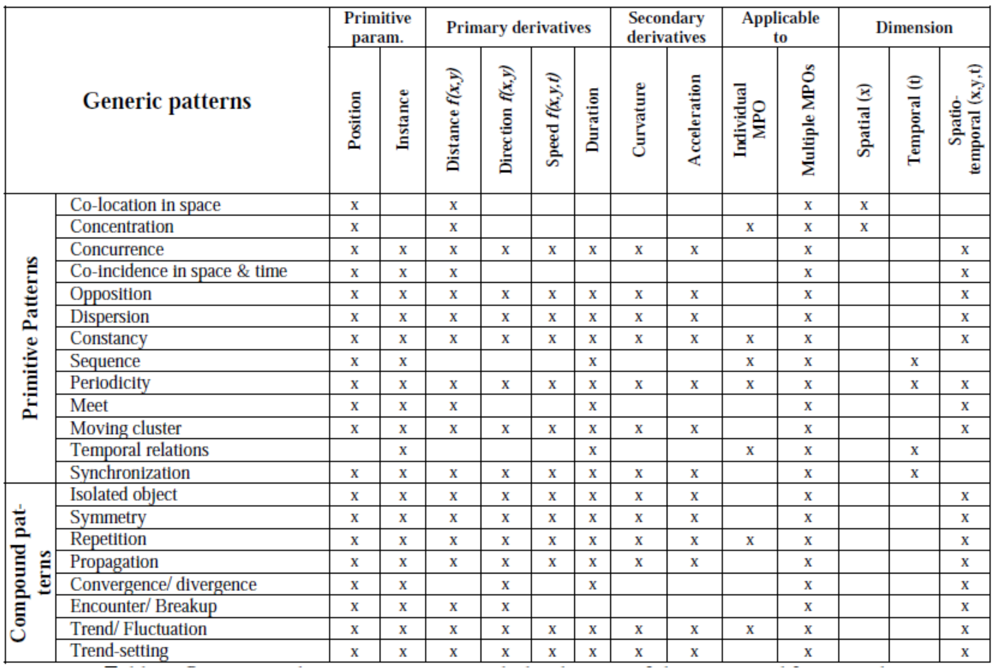
In this chapter, we have shown two applications of the proposed language based framework. The first application is about representation of a standard taxonomy (shown in Figure 2.3.2) in GIScience using **QDL**. For this, we have proposed certain extensions to the basic language constructs. Representation of a standard taxonomy suggests applicability of the framework in such a domain. Motion patterns like moving cluster, trend setting, encounter, breakup etc. have been represented using **QDL** descriptions.

The second application is about learning and representation of a binary motion pattern from video. A *Follow* **BMOP** between two persons is learned and represented.

6.1 Representation of a Standard Taxonomy

6.1.1 Movement Parameters

In Table 6.1 (cited from [1]), it is shown that each movement pattern may involve a number of movement parameters. It is necessary to discuss how each of these movement parameters can be represented in the proposed framework. The **QDL** based framework uses binary qualitative relations for representation of movement



	Generic patterns	Primitive param.		Primary derivatives				Secondary derivatives		Applicable to		Dimension		
		Position	Instance	Distance *f(x,y)*	Direction *f(x,y)*	Speed *f(x,y,t)*	Duration	Curvature	Acceleration	Individual MPO	Multiple MPOs	Spatial (x)	Temporal (t)	Spatio-temporal (x,y,t)
Primitive Patterns	Co-location in space	x		x							x	x		
	Concentration	x		x						x	x	x		
	Concurrence	x	x	x	x	x	x	x	x		x			x
	Co-incidence in space & time	x	x	x							x			x
	Opposition	x	x	x	x	x	x	x	x		x			x
	Dispersion	x	x	x	x	x	x	x	x		x			x
	Constancy	x	x	x	x	x	x	x	x	x	x			x
	Sequence	x	x				x			x	x		x	
	Periodicity	x	x	x	x	x	x	x	x	x	x		x	x
	Meet	x	x	x			x				x			x
	Moving cluster	x	x	x	x	x	x	x	x		x			x
	Temporal relations		x				x			x	x		x	
	Synchronization	x	x	x	x	x	x	x	x		x		x	
Compound patterns	Isolated object	x	x	x	x	x	x	x	x		x			x
	Symmetry	x	x	x	x	x	x	x	x		x			x
	Repetition	x	x	x	x	x	x	x	x	x	x			x
	Propagation	x	x	x	x	x	x	x	x		x			x
	Convergence/ divergence	x	x		x		x				x			x
	Encounter/ Breakup	x	x	x	x						x			x
	Trend/ Fluctuation	x	x	x	x	x	x	x	x	x	x			x
	Trend-setting	x	x	x	x	x	x	x	x		x			x

Figure 6.1: Elements of Movement Patterns(taken from [1])

parameters. There are certain elements in the taxonomy that demand special attention. These elements can not be represented elegantly by binary qualitative relations. The first of these is the primitive parameter *position*. Position refers to absolute physical locations in space and therefore, this parameter is more quantitative than qualitative. The same holds for the parameter *instance*. This parameter refers to a particular instance of time.

The primary derivative parameters like *distance*, *direction*, *speed* and *duration* can be represented in the standard representation of our framework i.e. using binary qualitative relations. *Curvature* and *acceleration* are secondary derivative parameters and can also be represented using binary qualitative relations.

Another important aspect is the number of objects involved in the pattern. When multiple moving point objects are involved, one of this can be chosen as the reference and qualitative relations can be defined with respect to this reference object. When only one object is present in the motion pattern, no such reference object can be identified. When a single object is involved, it is still possible to define qualitative relations for movement parameters like *direction*, *speed* and *acceleration*. For the movement parameter *direction*, we can define qualitative relations with respect to an external frame of reference even when a single moving point object is involved. Qualitative relations in absolute terms can also be defined for parameters like *speed* and *acceleration*. Distance of an object is always measured with respect to a reference point. When only one object participates in the movement pattern, some external object has to be implicitly taken as the reference and all distance measurements should be done with respect to this reference.

6.1.2 Representing the Parameters

Since a motion pattern is defined in terms of movement parameters, it is necessary to discuss how each movement parameter can be represented in our framework. We identify two different cases. The first is the case where a movement parameter can be represented using qualitative relations. The second is the case where there is reference to quantitative information and as such, qualitative relations can not be used. *Position* and *instance* are two such parameters. For representing these parameters, we extend **QDL** by introducing new types. We discuss below how each of these movement parameter can be represented:

Position and Instance

A position refers to a spatial location. The positional data are in the form of triples *(x, y, t)*, where x and y are coordinates of the spatial point and t is the time when these coordinates are recorded. We introduce two new types. One is for a position i.e. a location in space and the other is for an instance of time. The first type is named as **LocationPoint** and the second as **TimePoint**. Declarations can be done like:

BMOP Example {

LocationPoint p1=(10, 20), p2=(30, 40);

TimePoint t1=15;

}

Movement of a single moving point object results in a trajectory. Such a trajectory is a sequence of location points. A type is introduced for taking care of trajectories. This type is named as **Trajectory**. Sequence of timed location points is treated as a different type and named as **TimedTrajectory**. Declarations can be done like:

BMOP Example2 {

LocationPoint p1=(10,20), p2=(30,40), p3=(50,60);

TimePoint t1=15,t2=20,t3=25;

Trajectory tr1 = p1 p2 p3;

TimedTrajectory tr2=(p1, t1) (p2, t2) (p3,t3);

}

Speed and Acceleration

Speed and acceleration are used as movement parameters in a number of movement patterns in the taxonomy. We would like to discuss how these parameters can be represented by qualitative relations. In a relative representation, we will have to compare speed (or acceleration) of one object with respect to another. In this case, qualitative labels like *faster*, *slower* etc. are appropriate. In an absolute representation, qualitative labels can be introduced for each discrete quantity range.

Table 6.1: Relative Speed Relations

Sl. No	Relation Name	Meaning
1	*Slower*	Speed of primary less than speed of reference
2	*Equal*	Speed of primary equal to speed of reference
3	*Faster*	Speed of primary more than speed of reference

In Table 6.1, a set of qualitative speed relations are enumerated. These speed relations are based on a comparison of the speed of the primary with respect to the reference and therefore, relations are defined in relative terms. In Table 6.2, qualitative speed relations are defined in absolute terms. Such absolute relations can be used for representation of motion of a single object.

Table 6.2: Absolute Speed Relations

Sl. No	Relation Name	Meaning
1	*Stationary*	Speed = 0 kmph
2	*Slow*	Speed > 0 kmph, but <= 50 kmph
3	*Normal*	Speed > 50 kmph, but < 80 kmph
4	*Fast*	Speed > 80 kmph

Duration

Duration is a derived parameter and it is derived from time instances. In our framework, a duration can be modeled as an interval of time. Qualitative relations between such intervals of time can be expressed using operators based on Allen's interval algebra. Temporal constraints between two movement patterns are expressed by stating an Allen relation between their hold intervals.

There is another aspect of duration that needs special attention. *Duration* in the taxonomy refers to some absolute terms like *monthly*, *yearly*, *quarterly* etc. These terms can not be represented by Allen relations. In the proposed framework, we handle this issue by defining a set of qualitative relations for *duration*. The hold interval of a motion pattern can be compared with a such a qualitative relation label using the = relational operator. In Table 6.3, such a set of qualitative relations for *Duration* is listed.

Skip Transitions

Let us consider the movement pattern *colocation* in the taxonomy. This pattern is concerned with trajectories of motion. A trajectory is a sequence of location points. In this pattern, trajectories of moving objects have some points in com-

Table 6.3: Qualitative Relation for Duration

Sl. No	Relation	Meaning
1	*up-to-a-week*	days <= 7
2	*up-to-a-month*	days > 7, but <= 30
3	*up-to-a-quarter*	days > 30, but <= 120
4	*up-to-a-year*	days > 120, but <= 360

mon; but the pattern of movement of an object between two location points does not have any significance. To handle this type of requirement, we have introduced a *skip transition.* Symbolically, we denote a skip transition by $\circ$. For example, let us consider the following program segment:

```
BMOP Example3 {
LocationPoint p1=(10,20), p2=(30,40), p3=(60,80);
Trajectory tr1=p1 p2 p3;
var A: tr1;
}
```

The object A is at the point p1, then at the next observation it is at point p2 and so on. If we use the *skip transition* operator instead of concatenation operator between two points, the semantics will be different. Using *skip transition*, the above program is rewritten as:

```
BMOP Example4 {
LocationPoint p1=(10,20), p2=(30,40), p3=(60,80);
Trajectory tr1 = p1 ∘ p2 ∘ p3;
var A: tr1;
}
```

In this representation of the trajectory of A, after encountering p1, we skip the input location points so long as we do not encounter the location p2. Similarly, after encountering p2, input locations are skipped so long as p3 is not encountered. Location points are absolute physical points in space and the concept of spatio-temporal continuity of qualitative relations does not relate to them. On the contrary, primitive patterns exhibit continuity and it is interesting to analyse whether *skip transition* should be allowed with terminals of **QDL**. Apparently, use of *skip transition* with terminals is against the notion of continuity because after a terminal, its neighbouring terminal should always occur in input. It is important to mention here that a *skip transition* never states that neighbouring terminal does not occur after a given terminal. It only states that we want to skip certain terminals and look for the occurrence of a particular terminal. Therefore, we allow the use of *skip transition*s even with terminals of **QDL**.

Modification in Syntax

In **QDL**, trajectories are treated as type declarations. A trajectory represents motion of a single object. Variables can be declared to be of *trajectory* types. The type $< bmptype >$, introduced in chapter 5, takes care of binary motion patterns. Analogously, a unary type $< umptype >$ is introduced to handle the case of trajectories. This type is defined as:

$< umptype > ::= < location\_point\_expr > \mid < timed\_location\_expr >$

$< umptype > ::= < umptype > \bullet < umptype > \mid < umptype > \circ < umptype >$

$< location\_point\_expr > ::= (< integer >, < integer >)$

$< timed\_location\_expr > ::= (< integer >, < integer >, < integer >)$

Using $< umptype >$, type definitions can be performed in the following manner:

$< traj\_decl > ::=$ **Trajectory** $< traj\_name > = < umptype >;$

$< traj\_decl > ::=$ **Trajectory** $< traj\_name > = < umptype >; < traj\_decl >$

$<$ *traj_decl* $>$ ::= **TimedTrajectory** $<$ *timed_traj_name* $>$ = $<$ *umptype* $>$;

$<$ *traj_decl* $>$ ::= **TimedTrajectory** $<$ *timed_traj_name* $>$ = $<$ *umptype* $>$;
$<$ *traj_decl* $>$

$<$ *traj_decl* $>$ ::= ϵ

The framework that is presented in this thesis is flexible in terms of qualitative relations that can be used for modelling. Different models can be used for representation of movement parameters. By saying this, we note the fact that the qualitative relations that we have proposed in section 6.1.2 are not the only possibility. Any other relevant qualitative relations for modelling motion can be used. For example, a relevant work in GIS domain for qualitative modelling of trajectories is Qualitative Trajectory Calculus [137]. In this work, the motion of objects is represented by describing the distance between pairs of objects changing over time. QTC can represent and reason about movements of objects in Euclidean space. It is envisaged that QTC calculi may be useful in representing movements of objects in Geographical Information Systems [137]. It would interesting to explore how QTC relations can be used in our proposed framework for representation of patterns in the taxonomy.

6.1.3 Representing Patterns in The Taxonomy

In this section, examples are presented to show how each movement pattern can be represented in **QDL**. Examples are confined to generic patterns, both primitive and compound, because behavioural patterns can be constructed out of these generic patterns. In many of these patterns, the concept of a reference object is necessary for computation of binary qualitative relations. When we specify the pattern using a **QDL** program, the reference is expressed as a variable. During recognition, there has to be some mechanism for choosing the reference object. We would like to present a brief discussion on how the reference object can be selected during recognition.

Selection of Reference during Learning and Recognition

When the reference object is changed, the representation of a pattern changes. When an expression like *Reference S* is used in a program, the semantics is that there is some object S with respect to which we want to observe the pattern coded in the program. During recognition, any of the participating objects may qualify as this S. There are different ways to handle this reference selection problem.

Firstly, in certain cases, a reference may not be required at all. For example, when absolute qualitative relations are used to model the parameters, we do not have any reference.

Secondly, if objects have unique identities and if we know these identities during learning and recognition, then specification of reference can be done using these unique identities. In this case, when we write *Reference S*, S is the unique identity of some object.

Thirdly, a conceptual strategy may be used. For example, during learning we may always use the rectangle with the smallest X-coordinate value of the lower left point as the reference. Then, during recognition also, the same strategy has to be followed. Selection of a good conceptual strategy is an open research problem.

Fourthly, if the number of objects is small and volume of collected data is not large, then each object can be taken as reference in turn and all combinations can be computed.

Fifthly, an elegant solution to this problem would be to select some object as reference and to compute binary motion patterns for each object with respect to this selected reference object. If recognition succeeds, then the process can stop here. Otherwise, some other object has to be selected as reference. Instead of recomputing everything with respect to the new reference, we would outline a technique to transform the first computation into the new one. For this technique to work, the binary qualitative relations must be closed under composition and converse. We would like to explain this transformation using an illustrative example. Let there be four objects, symbolically denoted as A, B, C and D. Out of these, the object D is expressed as the reference. During recognition, we arbitrarily pick one of the four participating objects as the reference D. Now, binary motion patterns for the objects A, B and C with respect to D are computed. Along with each relation in a binary motion pattern, the start and finish time of the interval for which the relation holds is also recorded. If recognition with respect to this reference fails, then this representation has to be transformed with respect to a new reference. We denote the converse of a relation r_i as $r_i^{'}$ and the composition of two relations r and s as $r \odot s$. Moreover, we denote the binary motion pattern of an object X with respect to a reference Y as XY. Let us assume that following binary motion patterns are computed with respect to the reference D:

AD: $r_1r_2r_3$

BD:r_4r_5

CD:$r_6r_7r_8r_9$

Let the new reference be symbolically denoted as C. We would explain how binary motion pattern AD can be transformed. When r_1 holds in AD, we need to find out the relation(s) that hold(s) between D and C. We already know the relations between C and D. Therefore, converse of the relations between C and D need to considered here. These converses can be composed with r_1 to find the relations that hold between A and C when r_1 holds in the binary motion pattern AD. Temporal relationships between intervals of relations must be considered. Converses will be taken for the relations which have Allen relation *starts*, *overlap*, *during* and *finishes* with the relation r_1. These converses are composed with r_1 to get the list of relations that hold between AC when the relation r_1 holds in the binary motion pattern AD. After this, the next relation r_2 in the binary motion pattern AD is picked and the process is repeated to find out relations that hold between A and C when the relation r_2 holds between A and D. If the converse of a relation is a set, then composition has to be done with each relation in the set. Therefore, in such a case, we may get a set of binary motion patterns with respect to the newly selected reference and we need to check each for a match.

Colocation in Space

This pattern refers to absolute coordinates of places or positions. Trajectories of moving objects have locations in common. The exact form of the trajectory is not significant. Similarly, exact changes in the movement parameters are not important. *Skip transition*s can be used to represent this movement pattern. Three new operators are introduced in **QDL** for handling colocation in space. These are *OrderedColocate*, *UnorderedColocate* and *SymmetricColocate*. The first is for expressing ordered colocation, the second is for unordered colocation and the third is for symmetrical colocation. A program segment expressing an ordered colocation in space is presented below.

BMOP Example5 {

LocationPoint p1=(10,20), p2=(30,40), p3=(60,70), p4=(80,90);

Trajectory tr1 = p1 ∘ p3 ∘ p4;

Trajectory tr2 = p2 ∘ p3 ∘ p4;

var o1:tr1;

var o2:tr2;

o1 **OrderedColocate** o2;

}

We have declared four physical location points $p1$, $p2$, $p3$ and $p4$. $tr1$ is a trajectory type that has the occurrences of the points in the order $p1$, $p3$ and $p4$. $tr2$ is another trajectory type showing the occurrences of a set of locations. We have chosen direction as the sole movement parameter and since we are not concerned with how movement parameter values change in the trajectory, we have indicated this by using the syntactic representation of the skip transition i.e. the $\circ$. The object $o1$ is declared to be of type $tr1$ and $o2$ is associated with the trajectory $tr2$. Using the operator *OrderedColocate*, we specify that the trajectories of $o1$ and $o2$ have a ordered colocation pattern.

Concentration

In spatial concentration, the principle parameter is distance. Concentration of entities indicate their spatial closeness. An example is presented below to represent concentration of four objects. The object **S** is the reference.

BMOP Concentration(α,β,γ,λ) {

Reference λ;

type CLOSE_TO = (close) ;

var α: CLOSE_TO;

var β: CLOSE_TO;

var γ: CLOSE_TO;

α.HI **eq** β.HI;

α.HI **eq** γ.HI;

}

In the above program segment, the objects represented by type parameters α,β and γ are close to the reference object λ and the closeness of the three objects with the reference persist for the same time duration.

Concurrence

A set of entities exhibit the same values of movement parameters for a certain duration. A **QDL** representation of this pattern is given below.

BMOP Concurrence(α,β,γ) {

Reference γ;

type MOVEMENT = (very close, Same) (close, Same+) (near, lr+);

var α: MOVEMENT;

var β: MOVEMENT;

α.HI **eq** β.HI;

}

Movement parameters in this program are qualitative distance and direction. Both the objects α and β exhibit the same values for these parameters during the same time interval. Additional movement parameters like speed and acceleration can also be used. The changes in direction define the path curvature expected in the input trajectories.

Colocation in Space and Time

In full colocation, similar positions are attained at same time points whereas in lagged case, similar positions are attained after a time delay. There is reference to specific time points. So, we use $TimedTrajectory$ for representation of this pattern. Following example represents a full co-location pattern.

BMOP Full_Colocation {

TimePoint t1=10, t2=20, t3=30;

LocationPoint p1=(10,20), p2=(40,50), p3=(80,100), p4=(90,120);

TimedTrajectory tr1 = (p1, t1) ∘ (p2, t2) ∘ (p3,t3);

TimedTrajectory tr2 = (p1, t1) ∘ (p2, t2) ∘ (p4,t3);

var o1: tr1;

var o2: tr2;

o1 **FullColocateSpacetime** o2;

}

Timed trajectories $tr1$ and $tr2$ have a full colocation in space and time. The points $p1$ and $p2$ are common to both the trajectories and these are reached at same instances of time. This is represented by newly introduced **QDL** operator $FullColocateSpacetime$.

Opposition

In this pattern, there is a bipolar or multi-polar splitting of spatial entities. Direction of movement and distance are the principle movement parameters. As an example, let us consider movement of six objects having object identities from A to F. Suddenly these objects split into two groups. One group consists of A, B and C. The other group contains the rest. Movement of the objects before the split can be represented using a **BMOP**. Prior to the split, the objects are close and move in similar direction. This characteristics is retained after the split i.e. objects within each split group are close and move in similar direction. An important aspect to represent is the fact that objects in one group move in a direction opposite to the direction of movement in the other group. In the present form, **QDL** can not express qualitative relation between two **BMOP**s. Therefore, we take help of absolute direction relations to overcome the problem. In Table 6.4, a set of absolute qualitative direction relations is enumerated.

Table 6.4: A Set of Absolute Direction Relations

Serial No.	Relation Name	Serial No.	Relation Name
1	North	5	South
2	North East	6	South West
3	East	7	West
4	South East	8	North West

Initially, we assume that all the six objects move in north direction. After the split, objects A, B and C move in east whereas D, E and F move in west direction. Following is a program segment for representation of *opposition*.

BMOP Before_Split(α,β,γ,ω,λ,δ) {

Reference δ;

type Movement = (north, close);

var α: Movement;

var β: Movement;

var γ: Movement;

var ω: Movement;

var λ: Movement;

}

BMOP Group1(α,β,γ) {

Reference γ;

type Group1_Movement=(east, close);

var α: Group1_Movement;

var β: Group1_Movement;

}

BMOP Group2(α,β,γ) {

Reference γ;

type Group2_Movement=(west, close);

var α: Group2_Movement;

var β: Group2_Movement;

}

MOP Opposition {

var x: Before_Split(A,B,C,D,E,F);

var g1: Group1(A,B,C);

var g2: Group2(D,E,F);

x.HI **b** g1.HI;

x.HI **b** g2.HI;

}

Dispersion

A set of moving point objects performs a non-uniform or random motion. This randomness is not formalised in the taxonomy. A random pattern does not exhibit any uniformity in terms of movement parameters. Using **QDL** types, it is possible represent such random patterns.

Constancy

Movement parameters remain same or change insignificantly over a time interval. A **BMOP** can represent such a pattern. Let us assume that there are three objects, namely, P, Q and R, with object R acting as the reference. Speed, distance and direction as chosen as movement parameters. Following example is a **QDL** representation of a constancy pattern.

BMOP Constancy(α,β,γ) {

Reference γ;

type MOVEMENT = (slow, close, Same);

var α: MOVEMENT;

var β: MOVEMENT;

α.HI **eq** β.HI;

}

Spatio-temporal Sequence

It is an ordered sequence of visits to a series of locations. The taxonomy defines a sptaio-temporal segment between two locations. Each spatio-temporal segment has a start point and an end point in space as well as in time. A *spatio-temporal sequence* similar to *TimedTrajectory* in **QDL**. A *TimedTrajectory* can represent a spatio-temporal segment. Moreover, spatio-temporal segments are linearly ordered in time. This linear temporal relationship between spatio-temporal segments is expressible using operators based on Allen's interval algebra. An example, with direction as the sole movement parameter, is presented below.

BMOP Sequence {

TimePoint t1=10, t2=20, t3=30;

LocationPoint p1=(10,20), p2=(40,50), p3=(80,100);

TimedTrajectory tr1 = (p1, t1) ∘ (p2, t2) ∘ (p3,t3);

TimedTrajectory tr2 = (p1, t1) ∘ (p2, t2) ∘ (p4,t3);

var o1: tr1;

var o2: tr2;

o1.HI **b** o2.HI

}

Meet

In this pattern, objects remain within the radius of a disk for a certain duration. When spatial objects meet, they come close together. The notion of a disk of some radius expresses closeness. A numerical value of this radius can be easily represented by some absolute qualitative distance relation. For example, if the radius of the disk mentioned in the taxonomy is 3 meters and in our qualitative distance model, an absolute distance of 3 meters is assigned the qualitative label *near*, then the qualitative relation *near* can be used in the *meet* pattern as a representation of closeness. In the following program segment, a fixed *meet* pattern is encoded.

BMOP Meet(α,β,γ) {

Reference γ;

type CLOSE_TO = (near) ;

var α: CLOSE_TO;

var β: CLOSE_TO;

α.HI **eq** β.HI;

}

When objects meet, they remain close to each other for a certain time interval i.e. the pattern is not instantaneous. In the taxonomy, it is not specified whether the objects are stationary or in motion when they meet. If the objects are stationary during the meeting period, then speed should also be used as a movement parameter. For a stationary meet, the value of the qualitative speed relation will be *stationary*. In the above program segment, three objects, represented by type parameters, α, β and γ meet during an interval. Since qualitative speed relation is not used as a movement parameter in the above program, it covers both the cases of *meet* pattern.

Moving Cluster

Objects stay close to each other while taking the same path for a certain duration. The term *same path* can be interpreted to mean *same direction*. This direction can be absolute in an allocentric spatial reference frame or it can be relative in an egocentric reference frame. The exact reference frame to be used depends on the requirement of the application. Representation of a moving cluster is similar to that of *meet* pattern. Since the cluster is inherently *moving*, we should use a qualitative speed relation to indicate this. Following example illustrates a moving cluster:

BMOP Moving_Cluster(α,β,γ,λ) {

Reference λ;

type MOVE = (slow, close, Same);

var α: MOVE;

var β: MOVE;

var γ: MOVE;

α.HI **eq** β.HI;

β.HI **eq** γ.HI;

}

In the **BMOP** described in the above program, the objects α, β and γ are close to the reference during the same time interval. This makes it a *meet* pattern. The objects are not stationary because the qualitative speed has a value of *slow* for the objects. Moreover, these objects take the same path by moving in the *Same* direction. These two features make the *meet* pattern a moving cluster.

Full and Lagged Synchronisation

In full synchronisation, movement parameters show similar changes at same instances of time. In lagged synchronisation, similar changes are shown after a time delay. For example, in the first case, if speed was *slow* for the objects, then after a duration speed of all the objects may change to $fast$. The entire duration of observation can be broken down into different segments. In each segment, the movement parameters are similar for all the objects. At the end of a segment, *similar* changes occur and the next segment is initiated. Each of these *observation segment* can be represented as a **BMOP**. A full synchronisation pattern is illustrated in the following program.

BMOP S1(α,β,γ,λ) {

Reference λ;

type MOVE1 = (slow, Same);

var α: MOVE1;

var β: MOVE1;

var γ: MOVE1;

}

BMOP S2(α,β,γ,λ) {

Reference λ;

type MOVE2 = (medium, Same+);

var α: MOVE2;

var β: MOVE2;

var γ: MOVE2;

}

BMOP S3(α,β,γ,λ) {

Reference λ;

type MOVE3 = (fast, rl-);

var α: MOVE3;

var β: MOVE3;

var γ: MOVE3;

}

MOP Synchronisation {

var x: S1(A,B,C,R);

var y: S2(A,B,C,R);

var z: S3(A,B,C,R);

y.HI **m** x.HI;

z.HI **m** y.HI; }

In lagged synchronisation, changes do not occur at the same point of time. Still, changes are similar. Therefore, in this case, each *BMOP* should express the motion pattern of a single object so that the lag can be represented. An example program is given below.

BMOP S(α,β) {

Reference β;

type MOVE = (slow, Same)(medium, Same+)(fast, lr-);

var α: MOVE;

}

MOP Lagged {

var x: S(A,R);

var y: S(B,R);

var z: S(C,R);

x.HI **p** y.HI;

y.HI **p** z.HI;

}

Isolated Object

An individual moving point object pursues its own path. This object was part of a group. It got detached from the group and took a different path on its own. An important aspect to represent here is the fact that this individual object was part of a group. The group movement for objects other than this isolated object can be represented either by absolute or relative qualitative relations. For the individual isolated object, we will have to use absolute relations only. A **QDL** program is presented below.

```
BMOP Group($\alpha$,$\beta$,$\gamma$,$\lambda$) {
Reference $\lambda$;
type MOVE = (close,Same);
var $\alpha$: MOVE;
var $\beta$: MOVE;
var $\gamma$: MOVE;
$\alpha$.HI eq $\beta$.HI;
$\beta$.HI eq $\gamma$.HI;
}
BMOP Individual($\alpha$) {
type single=(north) $\circ$ (north east) $\circ$ (east);
var $\alpha$: single;
}
BMOP Group_Afterwards($\alpha$,$\beta$,$\gamma$) {
Reference $\gamma$;
type MOVE = (close,Same);
var $\alpha$: MOVE;
var $\beta$: MOVE;
}
```

```
MOP Isolated {
var x: Group(A,B,C,R);
var y: Individual(C);
var z: Group_Afterwards(A,B,R);
x.HI p y.HI;
z.HI pi x.HI;
}
```

Objects A and B maintain group behaviour by remaining close and moving in the same direction as the object R. After detachment, the movement of C is not defined with respect to the reference object R. It has been defined using absolute direction relations. The name of the object i.e. C in this example, binds the group with the individual.

Symmetry

In a symmetry pattern, the same patterns are arranged in reverse order. For representing symmetry, we create two patterns. In one, we create a bigger pattern by concatenating a set of movement patterns in some order. In the other, this concatenation is done in the reverse order. If there is any temporal relationship involved, then it can be represented by operators from Allen's interval algebra. In the following example, $C1$, $C2$ and $C3$ are binary motion patterns whose definitions are not shown in the program.

```
BMOP Symmetry($\alpha$,$\beta$) {
type CMP1 = C1 C2 C3;
type CMP2 = C3 C2 C1;
var $\alpha$: CMP1;
var $\beta$: CMP2;
```

α.HI **p** β.HI;

}

Repetition

The same pattern occurs again at a later point of time. We may be interested in knowing what exactly went on between two such occurrences of the pattern. It is equally possible that what is in between is not important to us. In the latter case, *skip transition*s can be used. In the example below, we have indicated that what occurs between two repetitions of the movement pattern $S1$ is not important for us.

BMOP S(α,β,γ) {

Reference γ;

type MOVE = (close,Same);

var α: MOVE;

var β: MOVE;

α.HI **eq** β.HI;

}

MOP Repeat {

var x: S(A,B,R);

var y: S(A,B,R);

x.HI **p** y.HI; }

Propagation

One object starts to show a movement parameter value. This means that movement parameter value for one of the objects starts changing. Gradually this change is propagated to other objects. There is a time delay i.e. other objects can not show the change simultaneously as the first one. For the remaining objects, this change may be simultaneous or may be lagged in any arbitrary order. In the example program given below, speed and direction are taken as movement parameters. Object X is assumed to show a change in movement parameter value. This change occurs in direction. Direction of the object changes from north to east. This change is later shown by the other objects A and B.

BMOP S1(α) {

type MOVE = (slow, north) (slow, north east) (slow, east);

var α: C;

}

BMOP S2(α,β) {

type MOVE = (slow, north) (slow, north east) (slow, east);

var α: C;

var β: C;

}

MOP S {

var x: S1(A);

var y: S2(B,C);

x.HI **p** y.HI;

}

Convergence and Divergence

In convergence, a set objects move to a common location and the direction of movement of these objects remains same. Objects need not arrive at the location at the same point of time. In divergence, objects disperse from a common location. There is no temporal constraint during dispersion. We introduce two **QDL** types for handling convergence and divergence. The type for convergence is named as *ConvergentTrajectory* and that for divergence as *DivergentTrajectory*. A formal definition of these types is given below:

$< ConvergentTrajectory > ::= < btype\_list > < location\_point\_expr >$

$< DivergentTrajectory > ::= < location\_point\_expr > < btype\_list >$

$< btype\_list > ::= < btype > \bullet < btype\_list > \mid \epsilon$

A convergent trajectory is a sequence of $< btype >$s. We know that a $< btype >$ can take the value of a single qualitative relation or it may be constructed out of several qualitative relation values using type induction operator $*$. In convergence and divergence patterns, these qualitative relations need to be absolute because a trajectory specifies movement of a single object. In convergent trajectory, we have a location point at the end. This point specifies the location of the place where objects finally arrive. In divergent trajectory, this point occurs right in the beginning. This is the point from which dispersion takes place. An example of specification of convergence is given below:

BMOP Convergence(α,β,γ,ζ,ι,ψ) {

LocationPoint p=(α,β);

ConvergentTrajectory CT=(γ)p;

var ζ: CT;

var ι: CT;

var ψ: CT;

}

Three objects,represented by parameters ζ, ι, ψ are moving in the direction γ and they have the point p in their trajectory at the end. There is no restriction on the point of time at which p is reached. This **BMOP** can be instantiated as:

Convergence(10,20,east,A,B,C)

Encounter and Breakup

These are very similar to convergence and divergence. The difference is that these have a temporal dimension. In encounter, objects arrive at the location at the same point of time. In breakup, they disperse at the same time point. Encounter may be represented as:

BMOP En(α,β,γ,ζ,ι,ψ) {

LocationPoint p=(α,β);

ConvergentTrajectory ET=(γ)p;

var ζ: ET;

var ι: ET;

var ψ: ET;

ζ.HI **f,fi,eq** ι.HI;

ψ.HI **f,fi,eq** ι.HI;

}

An example of breakup may be given as:

BMOP Br(α,β,γ,ζ,ι,ψ,ρ,μ) {

LocationPoint p=(α,β);

DivergentTrajectory BK1 = p (γ) ;

DivergentTrajectory BK2 = p (ζ);

DivergentTrajectory BK3 = p(ι);

var ψ: BK1;

var ρ: BK2;

var μ: BK3;

ψ.HI **s** ρ.HI;

ρ.HI **s** μ.HI;

}

Instantiation of these patterns may be done like:

En(10,30,west,A,B,C)

Br(10,20,west,south,southeast,A,B,C)

Trend and Fluctuation

Trend refers to consistent changes in the movement parameters. Fluctuation means inconsistent changes. Spatio-temporal continuity is a central notion in our framework. Consistency of change means that the amount of change is same whenever it occurs. For example, if direction is a movement parameter and if direction is changing consistently, we understand that the direction of the moving object changes by the same degree and this change is small. When change occurs in this manner, it is very likely that the spatio-temporal continuity of the qualitative relations will be obeyed by this change and we will gradually move from one relation to another in response to change. In case of fluctuation this change is irregular i.e. the change in the parameter fluctuates. Because of this, there may be *jumps* and spatio-temporal continuity of binary qualitative relations may be violated by observed values. Truly speaking this situation may occur in the case of trend pattern also if the change takes us forward crossing many qualitative relations. This type of situation can be represented by using skip transitions.

Trend Setting Pattern

Trend setter object triggers a movement pattern that is followed by a subset of other objects later on. These other objects may not be in spatial or temporal proximity of the trend setter object. Direction is a principle consideration in this pattern. Other movement parameters such as speed or acceleration may also be considered. One important characteristic of this pattern is that other objects will show the pattern only after the trend setter. An example is discussed below.

```
BMOP Trend_Setting($\alpha$,$\beta$,$\gamma$) {
type C1 = (slow, N) (slow, NE) (slow, E);
var $\alpha$: C1;
var $\beta$: C1;
var $\gamma$: C1;
$\alpha$.HI p $\beta$.HI;
$\alpha$.HI p $\gamma$.HI;
}
```

In the example above, α is the trend setter object. It keeps moving slowly, but the direction of motion, stated in an absolute frame of reference, changes gradually towards east. This pattern is exhibited later by β and γ also. The hold interval of the variable α precedes that of β and γ. So the trend setter shows the pattern first, then it is taken up by the other two objects. The temporal relation between β and γ need not be considered. A trend setting pattern can be more complex than this. We may want to represent a situation where β and γ may not wait for α to show the complete pattern. As soon as α changes its direction from north to north-east, β and γ too may do this. Then, α changes its direction from north-east to east and this is followed by β and γ. In such a case, the patterns $C1$, $C2$ and $C3$ will have to be broken up into smaller patterns so that the temporal relationship among these smaller patterns can be modeled to match the requirement.

Extensions, introduced in **QDL** in this chapter, are in the forms of new types and new operators. The type recognition task for these newly introduced types like **Trajectory**, **TimedTrajectory** etc. is different from the same for **bmps**. The efficiency of recognition of these types depends on the efficiency of algorithms used. Similarly, operators like **OrderedColocate** etc. require algorithms for their evaluation and time complexity of the evaluation task depends on the efficiency of the algorithms.

6.2 Learning and Representing a Motion Pattern from Video

In this section, we describe how a binary motion pattern can be learned and represented from real video. Two movement parameters, namely, spatial orientation and direction, are used for characterising the pattern. These movement parameters are qualitatively represented by the formalisms introduced in Chapter 3.

Input

For learning from video, we consider a binary motion pattern where a kid wearing a yellow sporting is followed by another wearing a red sporting. Two instances of this motion pattern are considered. A set of frames for the two instances of this *Follow* pattern are shown in Figure A.1 and in Figure A.2. Each video has approximately three hundred fifty frames.

Spatial relations are computed in the image plane. Definition of the pattern depends on the position of the camera. If the camera is placed at a different position, the pattern can be described completely differently. In computation of the spatial relations, the kid wearing yellow shirt is taken as the reference. It is possible that, in some test video during recognition, these kids may be switched. This is the reference selection problem. One solution is that during learning, we store binary motion patterns taking each object as reference in turn and during recognition check for a match in both the cases. Otherwise, we can take any of these as the reference and compute binary motion pattern during learning. During recognition, we can match with the stored one. If no match is found, then an automatic transformation is performed on the stored string to convert it to a form with respect to the other reference selection. Then, we check for a match again. Agents in a particular video frame are identified using morphological operations on images. For correspondence of blobs (say in frame t1) with those in another frame (say t2), a Kalman filter based prediction is used. We note that with change of camera position occlusion may occur and this is particularly true when we consider movement away from the camera by placing the camera at the back. In our implementation, all blobs need to be fully visible always.

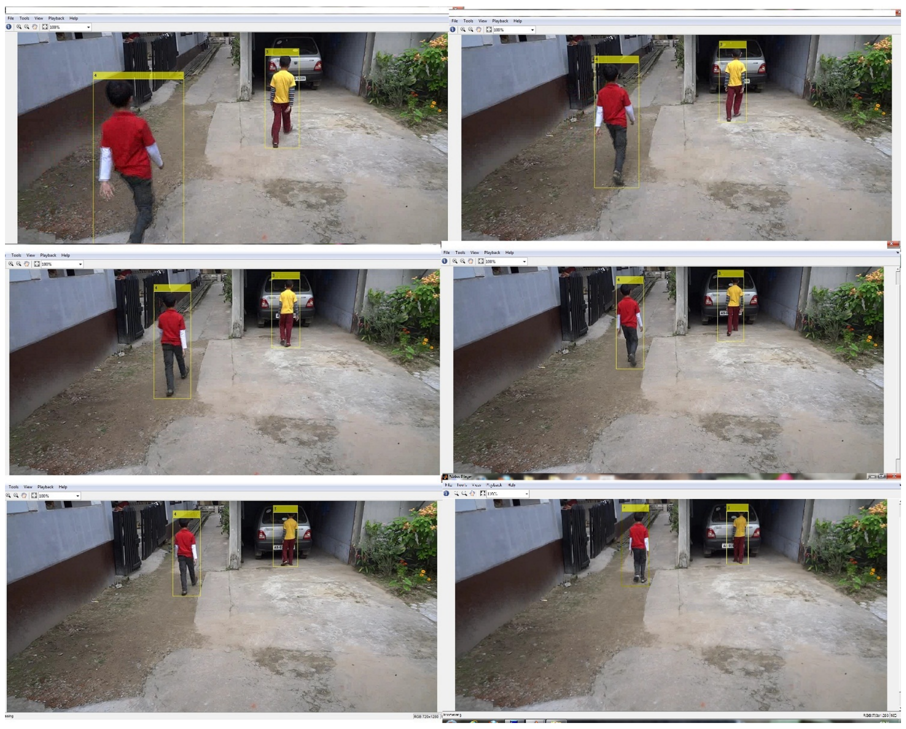


Figure 6.2: First Instance of Follow Pattern

These frames show some tracked objects from their corresponding videos. Objects are tracked using a constant velocity Kalman filter and Minimum Bounding Rectangles (MBR) are drawn around each tracked object across frames. The frames should be viewed row-wise. In Figure A.1 and in Figure A.2, the kid wearing the yellow sporting is considered as the reference object. In each frame, we need to compute the spatial orientation relation between the reference MBR and the primary MBR.

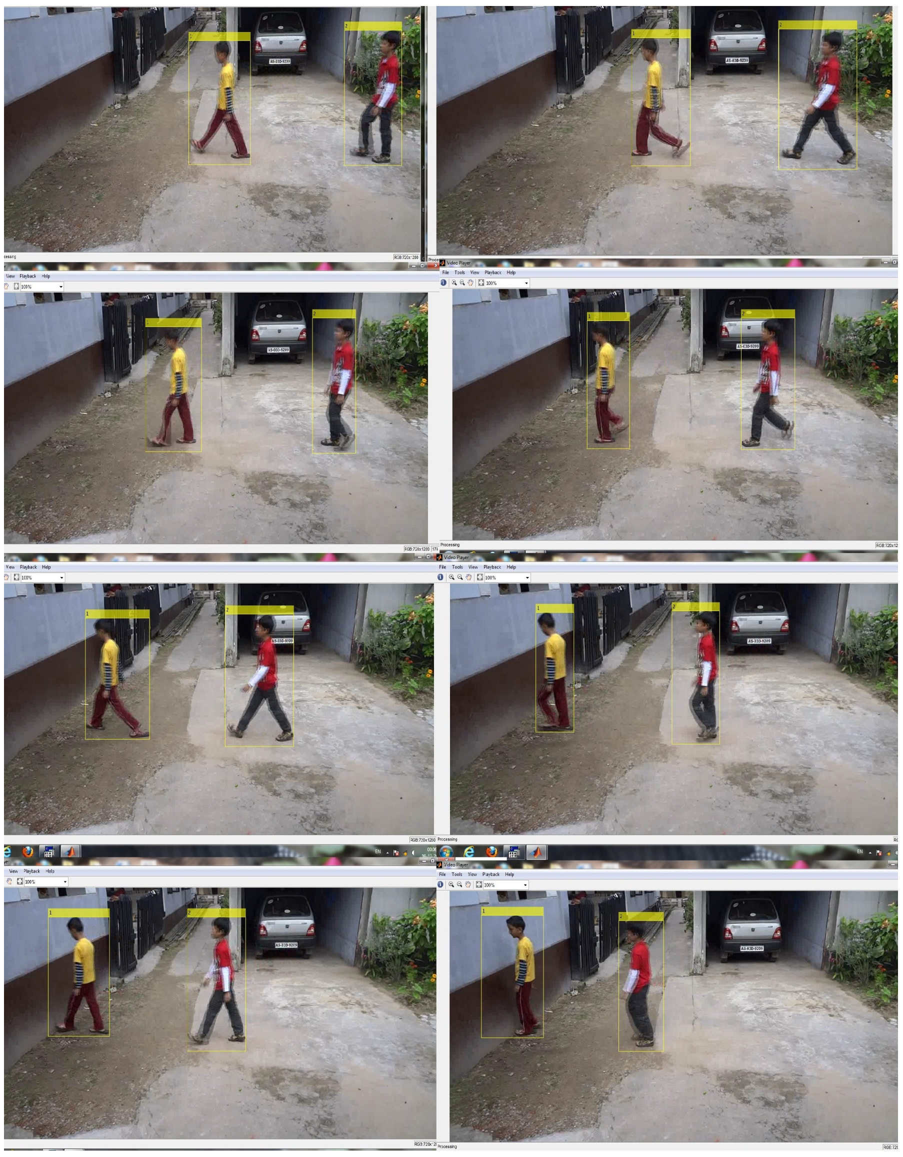


Figure 6.3: Second Instance of Follow Pattern

Finding the Orientation and Direction Relations

In Figure 6.4, two rectangles are shown. The rectangle with end points $R1$, $R2$, $R3$ and $R4$ is the reference and the other one with end points $P1$, $P2$, $P3$ and $P4$ is the primary. The orientation relation can be computed by considering the position of these points with respect to the lines AB and CD. For example, for the $Front$ relation to hold, the points $P1$, $P2$, $P3$ and $P4$ must be on the left of the line AB and on the right of the line CD. Moreover, the y-coordinate of the point $P3$ (or $P4$) must be less than the y-coordinate of the point $R1$ (or $R2$).

An MBR alone does not convey any information regarding direction of mo-

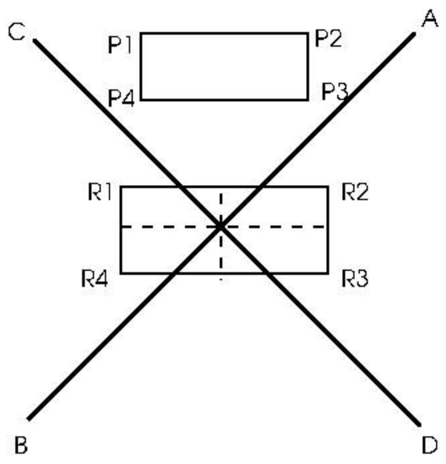


Figure 6.4: Computation of Spatial Orientation Relation

tion. For finding direction of motion of the primary, we consider the positions of the primary in the current frame and in the subsequent frame. Same procedure is adopted for finding the direction of motion of the reference MBR. This idea is illustrated in Figure 6.5.

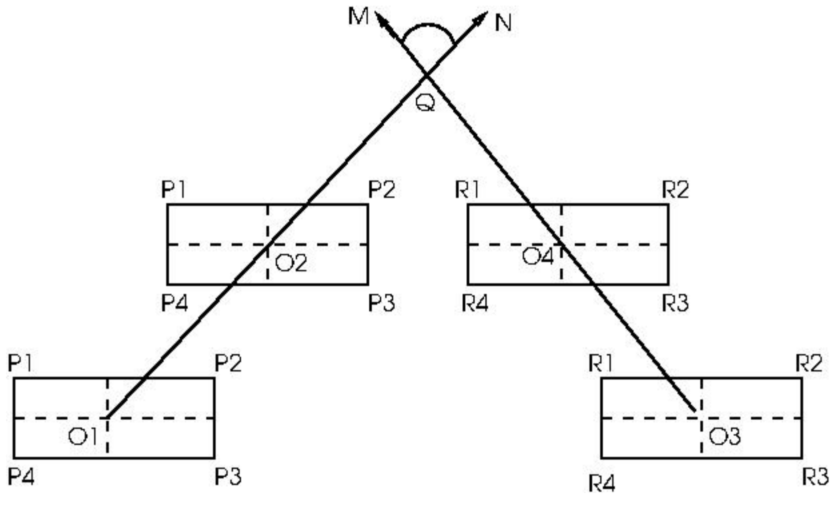


Figure 6.5: Computation of Direction Relation

The end points of the primary MBR are $P1$, $P2$, $P3$ and $P4$. In the current frame, the intersection of the lines joining mid points of opposite sides of the primary MBR is denoted as $O1$. The same in the subsequent frame is denoted as $O2$. The line joining the points $O1$ and $O2$ gives the direction of motion of the primary MBR. Similarly, the line joining the points $O3$ and $O4$ gives the direction of motion of the reference MBR. These two lines are extended and they intersect at the point Q. MQN is the angle between the direction lines of the two MBRs and the qualitative direction relation can be found after computation of this angle.

Results and Analysis

In Appendix A, the results of learning are shown. For each instance, the binary motion pattern is learned. The learned binary motion patterns before and after removal of discontinuities are shown. The first instance is learned and represented as a regular grammar. After learning the second instance, it is stored disjointly from the first using a different set of productions.

We observe discontinuity in both the learned instances. For example, in the first learned binary motion pattern, two consecutive terminals are $(B\&LO, \mathit{Opposite})$ and $(B\&LO, \mathit{Same}+)$. These are not neighbouring terminals because *Same*+ is not a conceptual neighbour of *Opposite*. As there is no occlusion in both the videos, the reasons for occurrence of this form of discontinuity need to be analysed. It is observed that discontinuity occurs more in the case of qualitative direction relations. One reason may be tracking errors of the underlying program. Computation of direction relations depends on the positions of the primary and reference MBRs at different time points. Discontinuity will occur if certain positions are missed in tracking. Spatial orientation relations are be affected by such misses in *boundary cases*. This means that these misses will be more prominent when a transition is about to occur from one spatial orientation relation to another. In other cases, the same orientation relation may hold even if certain positions are missed. Qualitative direction relations are more sensitive to such misses because misses in tracking of objects at certain positions will change the direction line of the objects, thus affecting the direction relation.

After removing the discontinuities, both the instances are represented as regular grammars. The productions of these grammars are listed. The productions of the grammar after learning the second instance are enumerated separately in order to show how each of the instances is learned and represented.

Since both are learned instances are different, they are recorded as two different ways of performing the event. This is taken care of by using two non-terminals *A0* and *B0*. The productions for the first instance use *A0* as the first left hand side non-terminal. Similarly, productions for the second instance use *B0* as the first left hand side non-terminal.

6.3 Recognition of Motion Patterns

In this section, we have attempted to recognise motion patterns using data from different sources. The basic construct that we attempt to recognise is a **BMOP**. A **BMOP** is a conceptual grouping of binary motion patterns observed among a set of objects along with temporal constraints. Algorithm 5, presented in chapter 4, try to recognise a **BMOP** from data. This algorithm is based on an approach where we first learn a set of examples and try to recognise an unseen case by matching against the stored examples. Moreover, it looks for a perfect match of binary motion pattern strings starting from the first position. Temporal constraints, expressed using Allen relations, must also be satisfied. The principle that we have followed is that any motion pattern string, learned as example or recorded as new instance, respects spatio-temporal continuity and accordingly, we modify the string into a spatio-temporally-continuous form as outlined in section 4.4.2.

Data were collected from three sources. The first is GPS tracking data for recognition of motion patterns in the GIS domain. The second source is synthetic data that were collected as cognitive input of a set of persons using an application. The third one is real video data.

For the second case i.e. for synthetic data set, the learning and recognising approach using the recognition algorithm was somewhat encouraging. An important point here is that synthetic data set constructed out of cognition may have some limitations. This may be due to the fact that in defining an event using a tool, people will perhaps try to make this event *happen*. As a result, unpredictability of input, that is typical of real data may not be simulated fully.

In the GIS domain, we focused on recognition of patterns mentioned in [1].

The recognition algorithm 5 is not directly applicable in this case as we do not have learning examples for these patterns. Each GIS pattern has some sort of an informal definition. We have encoded this definition inside implementation programs. Typically, in these encodings, we match the sequences of cardinal directions observed for the participating objects, inspect the value of the *distance* relation in each sequence and check the temporal constraints in terms of Allen relations.

Recognition of motion patterns in real video was the most challenging task because of presence of noise. The term *noise* can cover a number of issues. Typically, in video processing we encounter problems like non-tracking of objects across subsequent frames, merging of blobs, occlusion and tracking errors. Presently in the proposed framework, we can not handle issues like non-tracking, merging and occlusion. What we have attempted to process is discontinuous tracking that may occur as a result of the inability of the tracking program to track the correct object at the correct position and also to track the full object without including any extra details from the background. We recorded a number of videos. These videos have lot of noise and finally we could extract six videos which can be used for analysis as per the limitations of the framework mentioned above. Four out of these videos were for an *overtake-on-right* event and two for an *overtake-on-left* event.

We started our analysis with four videos of *overtake-on-right* type. We tried to first learn and then to recognise a new instance. Out of four, we picked the first as a new instance to be recognised and the remaining as training examples. Algorithm 5 was then invoked, but we could not find any match for recognition. Then, similarly, second, third and fourth videos were selected as new instances in turn. Only for the second video, we could get a match i.e. it was recognised. This is inspite of the fact that videos were recorded in identical environment one after another. The conclusion is perhaps the fact that perfect matching is very difficult to achieve in such a noisy environment. Therefore, the approach of learning set of examples and then trying to recognise using algorithm 5 that was encouraging for synthetic data was not effective for a small set of four videos in identical environment, but variable noise set-up. For modelling the motion parameters, the direction-orientation model introduced in chapter 3 was used. The recorded binary motion patterns for these four videos are presented at serial number 1 in Appendix D.

Since results were not encouraging for the earlier approach, a different approach was tested for video recognition. Definition of an *overtake* event is expressed as a **BMOP** in **QDL**. This definition is based on our common perception of a person overtaking another while walking along a road. This time, instead of going for a perfect match, we have looked for salient points. The **BMOP** definition results in string(s) that define(s) salient points to look for in the recorded continuous new instance string. An operator with such semantics was introduced in section 6.1.1 and named as *skip transition* (was denoted by operator symbol $\circ$). A skip transition takes a list of primitives to look for in another string. Then, it takes the first primitive in the list and looks for the first occurrence of it in the second string. Once found, it takes the second primitive in the list and starts looking for it from the position where it encountered the first primitive. This process continues and if the last primitive also can be located in the string, a match is announced. The semantics of $\circ$ operator can be described by the following algorithm (In this algorithm, the parameter l contains the list of primitives to look for in the second parameter string s. The variable m stores the index of the last primitive in s, n stores the index of the last primitive in l and l is assumed to be like $a_1a_2...a_n$ where each a_i is a primitive):

Algorithm 6 Skip_Transition(l,s)

```
1: if n > m then
2:     Exit
3: end if
4: j:=0
5: status:=true
6: for i:=1 to n do
7:     Look for $a_i$ in $s$ from index $j+1$ to index $m$
8:     if $a_i$ is found in $s$ then
9:         j:=Index of the position where $a_i$ is found
10:    else
11:        status:=false
12:        break
13:    end if
14: end for
15: if status=true then
16:     print('Recognised')
17: else
18:     print('Not Recognised')
19: end if
```

In this approach, we started with the direction-orientation model introduced in chapter 3. Dependence of spatial orientation on direction of motion proved to be a problem in this approach. It turned out that direction relations are very sensitive and change rapidly. Since orientation labels are dependent of direction in this egocentric FoR, orientation labels may be difficult to comprehend. For example, at the point of overtake (when the person has moved to front), if the direction of the reference person changes, the orientation relation may be computed as *back* instead of *front*. In such a case, a program coded using commonsense knowledge of an overtake will not match with the recorded instance in most of the cases.

We detached orientation from direction and used a different model. In this new model, direction relations were same as the ones introduced in chapter 3; but for orientation, rectangular cardinal directions [32] [2] were used. Among the four videos of overtaking on right (where learning and recognition approach could have only one hit), the $\circ$ operator could recognise all four as instances of right overtake.The recorded binary motion patterns for these four videos using the new model are presented at serial number 2 in Appendix D.

For analysis, we carried out one more experiment taking two videos of left overtake and two other videos of right overtake. These two right overtake videos were included in the earlier experiment. We attempted to recognise the events as per the definition. In this case, we could recognise three events. The binary motion patterns for this experiment are presented at serial number 3 of Appendix D. As a final analysis for event recognition in video, we performed one more experiment for recognition of a *follow* event using four video samples. Out of these, in two videos, a person walking along a street was being followed by another on the left side; whereas in the other two videos, the person behind was on the right side. We could successfully recognise the event using *skip* transition operator.

The skip transition operator is not free from limitations. It may give recognitions which perhaps were not intended. For example, while overtaking on left, if one comes to a position on left and then goes back and makes an overtake on right and comes to front, this will also be recognised as an *overtake-on-left*. At present, **QDL** does not have adequate operators to eliminate such cases. There is a scope to extend **QDL** with more operators for performing intricate substring matching operations along temporal dimension. For example, in the above false recognition case, we should be able define a substring and state that this substring should not occur in the future after occurrence of a given primitive (the one for being on *left*).

6.3.1 Recognition of Motion Patterns in GIScience

Data Collection

Data were collected using a GPS tracking application. Five persons were assigned the responsibility of collecting movement data along thirteen routes. Out of these, one two route is traversed twice.

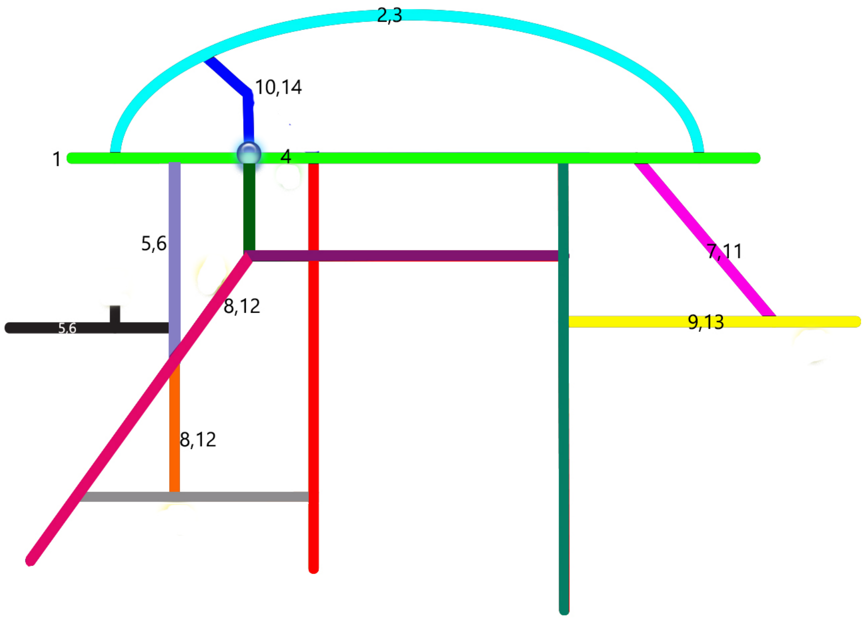


Figure 6.6: The Route Network in GIS Data Collection

We have treated it as a separate instance of a route and therefore, we take the number of routes as fourteen. In Figure 6.6, these routes have been illustrated. These routes were decided in advance. Sometimes, the persons travelled together;but this was not the case always. Moreover, the collection of data spread over several days. At the end of the day, individuals submitted the collected data along with their starting and finishing time for the route. This time is dependent on the length of the route, speed of the person and traffic conditions along the route. Persons used motorbikes where the piggy rider was responsible for using the GPS tracking application and collecting data. Persons never stopped and therefore, the possibility that they may be stationary is not taken into account in our analysis.

Data, so collected, are in the form of latitudes and longitudes. For our analysis, we need to convert data into Cartesian coordinate form. Therefore, from latitudes and longitudes, data are converted into *UTM* form (*Universal Transverse Mercator*). In this form, there are many *UTM* zones for the entire globe and within a zone, longitude and latitude are represented in the form of *easting* and *northing*. For this part of the world, all routes fall in the *UTM* zone *46R*. In our analysis, *easting* is taken as the X-coordinate and *northing* is taken as the Y-coordinate.

Implementation Issues

Movement data along with temporal information for the routes are listed at serial number two of Appendix B. Distance and cardinal directions are taken as movement parameters. Since the persons did not stop during their movement, we have not used any *speed* relation to check whether they were stationary. Four qualitative relations, namely, *veryclose*, *close*, *near* and *far* are used to model distance. Eight cardinal direction relations, namely, *east*, *west*, *north*, *south*, *north-east*, *north-west*, *south-east* and *south-west* are used to model direction of motion. These are abbreviated as *E*, *W*,*N*,*S*,*NE*,*NW*,*SE* and *SW* respectively.

At serial number three of Appendix B, motion patterns extracted from data for each participating object are listed. For each route, four combinations are computed. Each combination is with respect to a particular reference selection. For example, in *Combination One*, the first object is selected as reference. So, *Object A*, listed in that combination, gives the recorded motion pattern of the second object with respect to the first, *Object B* indicates the pattern of the third object with respect to the first and so on. Similarly, *Combination Two* is for the selection of the second object as reference. Since the number of objects is small, we have chosen to compute patterns taking each object as reference in turn. Since cardinal relations give absolute direction, they remain same even if the reference is changed.

Five different types of patterns have been recognised. These are *Moving Cluster*, *Concurrence*, *Propagation*, *Trend Setting* and *Opposition.* The patterns are studied principally in the context of sequence of absolute direction relations recorded. This sequence gives us the concept of a *path* that is traversed or a *similar* value for a motion parameter or a pattern that may be *propagated* or a *trend* that is followed. So, the main idea during recognition is that objects should exhibit identical sequence. Along with this, distance and temporal information are used to distinguish patterns. Some patterns need proximity among objects. Therefore, once sequences are identical, distance relation is verified to identify this proximity. If the distance relations are *close* or *veryclose*, we have considered the objects to have proximity. The last check in recognition process is regarding time. Time intervals are computed from start and finish times for each object and Allen relations between intervals are computed from these. These Allen relations are analysed for conformance to temporal constraints.

Results and Analysis

In Appendix B, at serial number 1, we have tabulated the motion patterns recognised along each route. Along *ROUTE 1*, *Concurrence* pattern is recognised. For such a pattern, proximity is not a necessity. Entities need to show similar motion parameter values over a duration. In none of the four combinations for this route, we observe patterns where objects are always either *veryclose* or *close* to the reference. At some point or other, they have come *near* the reference. Because of this, movement along *ROUTE 1* did not qualify as a *Moving Cluster.* The recognised pattern is written as a **BMOP** below:

BMOP Concurrence(α,β,γ,δ,ψ) {

Reference α;

type β_MOVEMENT=(veryclose,NE)(close,NE)(close,N)(close,NW)

(close,W)(veryclose,NW)(veryclose,N)(veryclose,NE)(veryclose,E);

type γ_MOVEMENT = (veryclose,NE)(close,NE)(veryclose,N)

(veryclose,NW)(veryclose,W)(close,NW)(close,N)(close,NE)(close,E);

type δ_MOVEMENT = (close,NE)(veryclose,NE)(close,N)(close,NW)

(close,W)(near,NW)(near,N)(near,NE)(near,E);

type ψ_MOVEMENT = (close,NE)(veryclose,NE)(close,N)(close,NW)

(close,W)(near,NW)(near,N)(near,NE)(near,E);

var β: β_MOVEMENT;

var γ: γ_MOVEMENT;

var δ: δ_MOVEMENT;

var ψ: ψ_MOVEMENT;

α.HI **eq** β.HI;

β.HI **eq** γ.HI;

γ.HI **eq** δ.HI;

δ.HI **eq** ψ.HI;

}

This **BMOP** is based on the selection of the first object as reference. Actually, the definition of the recognised **BMOP** can be based on any of the four combinations because during recognition, all the reference selections are explored.

Along *ROUTE 2* and *ROUTE 3*, *Moving Cluster* pattern is recognised. Along these routes, objects start and finish at the same time. We find at least one combination for which the objects are always *veryclose* or *close* to the reference. Moreover, objects take the same path as shown by the sequence of absolute direction relations.

Along *ROUTE 4*, the recognised pattern is *Propagation*. Objects have different start and finish times for this route. An object (according to temporal information, it is the first object i.e. object number 1) starts to show a movement parameter value i.e. movement along north-east direction. Other objects start to show this value at later time points.

Along *ROUTE 5* and *ROUTE 6*, the pattern is *Moving Cluster* again. Along *ROUTE 7*, we have an interesting case. The recorded patterns and temporal information indicate that this can qualify as a *Propagation* pattern also. According to definition, in a *Propagation* pattern, the trend-setter object does not have any influence on the movement of other objects that start to exhibit the same motion parameter value. In our framework, we do not have any method to formalise such *influence*. We have listed it as *Trend Setting* because in *Propagation*, we observe value of a particular movement parameter whereas in this case the *trend* is something more than a single value of a movement parameter.

Similar considerations were involved in recognition of motion patterns from *ROUTE 8* to *ROUTE 14*. An *Opposition* pattern was observed involving routes *1*, *10* and *12*. In this case, we used our knowledge about the routes. Routes one, ten and twelve meet at a junction and as per information of the persons, they split at this point. Three of them moved along route ten and two took the path along route twelve. Therefore, we used the recognition algorithm to explore the possibility of an *Opposition* pattern involving these routes. Objects were moving in north-east direction before splitting. After split, some of them moved in north-west direction whereas the remaining moved in south-west or south-east direction. Since routes are not always straight and involve turns, we can consider anything north-bound (NW,N,NE) as opposite direction of anything south-bound (SW,S,SE). In recognition, we use this concept of *opposite* directions and find that objects along route ten are moving in a direction *opposite* to the one in which objects along route twelve are moving. The recognised *Opposite* pattern is written as a **MOP** below:

BMOP Along_Route1(α,β,γ,ω,λ) {

Reference α;

type Movement = (NE);

var β: Movement;

var γ: Movement;

var ω: Movement;

var λ:Movement;

}

BMOP Group_Along_Ten(α,β,γ) {

Reference α;

type Group1_Movement=(NW);

var β: Group1_Movement;

var γ: Group1_Movement;

}

BMOP Group_Along_Twelve(α,β) {

Reference α;

type Group2_Movement=(SW);

var β: Group2_Movement;

}

MOP Opposition_Recognised {

var x: Along_Route1(α,β,γ,ω,λ);

var g1: Group_Along_Ten(α,β,γ);

var g2: Group_Along_Twelve(ω,λ);

x.HI **m** g1.HI;

x.HI **m** g2.HI;

}

Another information we have about routes is that route two and route three are actually along the same road. We use this knowledge to explore the possibility of occurrence of a *Repetition* pattern for these routes. There is at least one *combination* in each of these routes where the the sequence of absolute direction relations is same. This sequence is *NW, W,SW*. This recognised *Repetition* pattern between route two and route three is written as a **MOP** below:

BMOP S(α,β,γ,δ,ω) {

Reference α;

type MOVE = (NW,W,SW);

var β: MOVE;

var γ: MOVE;

var δ: MOVE;

var ω: MOVE;

α.HI **eq** β.HI;

β.HI **eq** γ.HI;

γ.HI **eq** δ.HI;

δ.HI **eq** ω.HI;

}

MOP Repetition_Recognised(α,β,γ,δ,ω) {

var x: S(A,B,C,D,E);

var y: S(A,B,C,D,E);

x.HI **p** y.HI; }

What could not be recognised

Certain routes are opposite in terms of movement along a road. For example, *ROUTE 5* is the movement from a place (say *s1*) to some other place (say *d1*) and *ROUTE 6* is movement from *d1* to *s1.* Obviously, we expect to observe *Symmetry* pattern in these cases;but this did not happen. Perhaps, this is due to the fact that the concept of *reverse* of a sequence of primitives has not been formalised in the framework. This *reverse* of a sequence may not always be obtained by reversing the string from end to beginning. This is particularly true in the case of directions.

Convergence pattern could not be recognised. Along each route, objects are finally approaching the same destination. So, in each case, there is always a possibility of a *Convergence* pattern. For recognition of such a pattern, some abstraction of a *place* is necessary. Convergence to a point is difficult to recognise because the coordinates of that point may vary with time during GPS tracking. Same arguments hold for patterns like *Colocation in Space*, *Colocation in Space and Time* and *Spatio-Temporal Sequence.*

6.3.2 Recognition of Movement Patterns using Synthetic Data

We tried to recognise two different motion patterns. The first one is a two-object *overtake* pattern. The second one is a four-object pattern. In this pattern, the first object is followed by the second. The third object approaches from left and the fourth object approaches from right. The data set is generated from cognitive input by a set of persons.

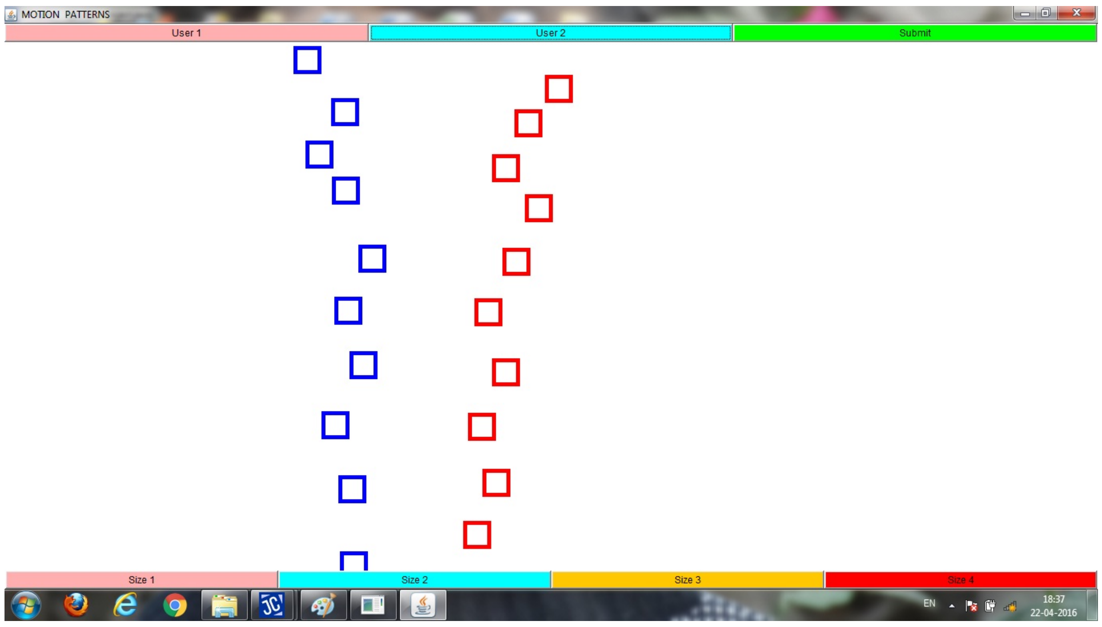


Figure 6.7: User Interface for Two Object Data Collection

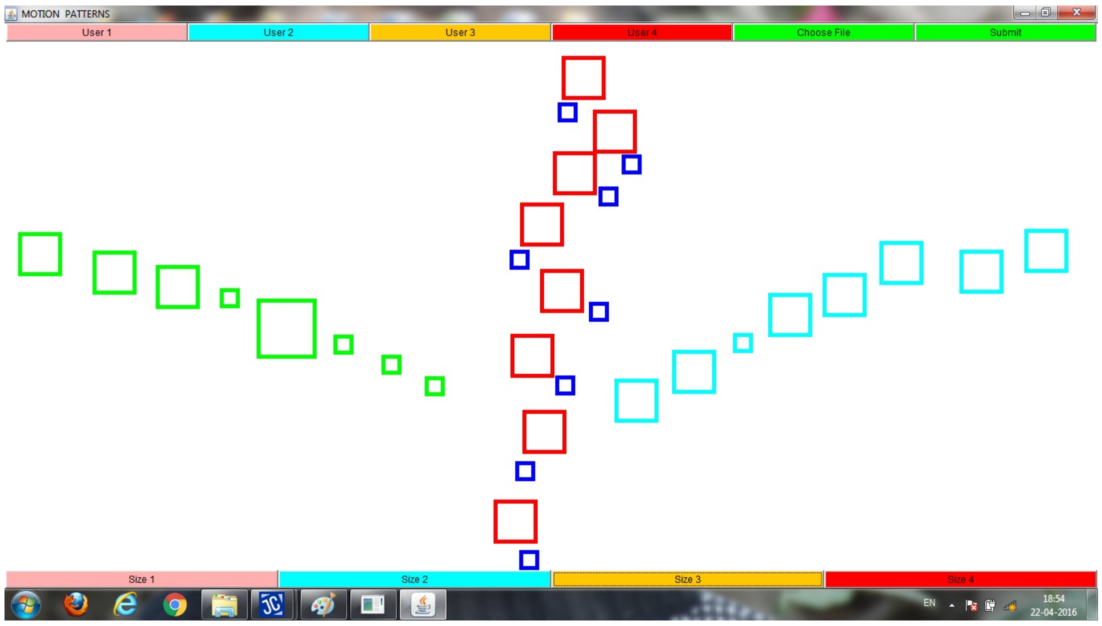


Figure 6.8: User Interface for Four Object Data Collection

Data Collection

Two applications were developed for collecting data. The user interface of the application for two-object data collection is shown in Figure 6.7 and the same for four-object data collection is shown in Figure 6.8. A group of ten persons were explained about the events and were requested to generate data using the interfaces. For example, in Figure 6.7, the button *User1* is clicked first and then a rectangle size is selected by clicking on one of the *size* buttons. Then on clicking the canvass, the positions of the first object is specified from bottom to top. Sizes of rectangles can be made to vary during this process.The red rectangle is for the first object. After this, a user clicks on *User2* button for indicating the positions of the second rectangle. The blue rectangle is for the second object. On submitting, rectangle coordinates get saved along with object identifiers. Such input about the overtake event is based entirely on user's perception of the event. Each person participating in the experiment provided ten input for this overtake event.

Similarly, in Figure 6.8, there are four buttons named as *User1*,*User2*,*User3* and *User4*. The red rectangle is for the first object which is being followed by the blue rectangle. Both are moving from bottom to top. The object (the green one) approaches from left to right whereas the fourth one approaches from right to left. On submitting, rectangle coordinates get saved along with object identifiers. The same set of persons provided cognitive input for this event also and as before, we collected hundred and one data samples. For both these cases, temporal information was not gathered as we felt that this may make the entry process tedious for the persons.

Results and Analysis

For implementation, we used the direction and orientation model introduced in Chapter 3. Qualitative direction relations listed in Table 3.1 are used for modelling direction of motion and orientation relations enumerated in Table 3.3 are used for modelling spatial orientation. Discontinuity is present in user input. For example, the two-object event shown in Figure 6.7 is recorded as:

```
(B,Same-)(L,Same-)(L,lr+)(L,Same-)(L,Same-)(L,lr+)(L,lr)(L,Same+)(L,lr)
```

Here, a discontinuity is present between primitives *(L,lr)* and *(L,Same+)*. This is processed as:

```
(L,lr)(L,lr+)(L,Same-)(L,Same)(L,Same+)
```

Another discontinuity is present between *(L,Same+)* and *(L,lr)*. This is processed as:

```
(L,Same+)(L,Same)(L,Same-)(L,lr+)(L,lr)
```

Therefore, after removing discontinuity, the final continuous string becomes:

```
(B,Same-)(B&L,Same-)(L,Same-)(L,lr+)(L,Same-)(L,lr+)(L,lr)(L,lr+)
(L,Same-)(L,Same)(L,Same+)(L,Same)(L,Same-)(L,lr+)(L,lr)
```

We note that, after overtake, the observed relation may not be *Front* always. Since direction of motion affects the orientation relations, some other relation may also be observed.

Among hundred examples collected, we pick the first one as a new instance and take the remaining as the training set. We try to recognise this instance using the training set. We invoke algorithm 5 for recognition. We repeat this process by taking each example as a new instance and then trying to match it against the rest. The results are presented in Appendix C. In the first column of this table, we list the instance number and in the second column the status of recognition is shown.

The results of four-object recognition are also presented in Appendix C. The process of selecting new instance and training set is identical to what was done for two-object case. An event is recognised only when the status of each participating object becomes *Recognised*.

Though the results for recognition using synthetic data may seem encouraging, it can not be compared to real video data. For example, in synthetic generations, users will not perhaps click the last rectangle in the bottom most position even if the movement is upward. Similarly, he/she will perhaps will not click a subsequent rectangle that is so big that it extends far below the current rectangle. This simulation too has *noise*. This *noise*, we feel, arises principally out of the inability to click at the right positions using such a tool. Therefore, this analysis is centred around some particular type of *noise* and does not cover all the unpredictability encountered in object tracking in video.

6.3.3 Recognition of Motion Patterns from Video Input

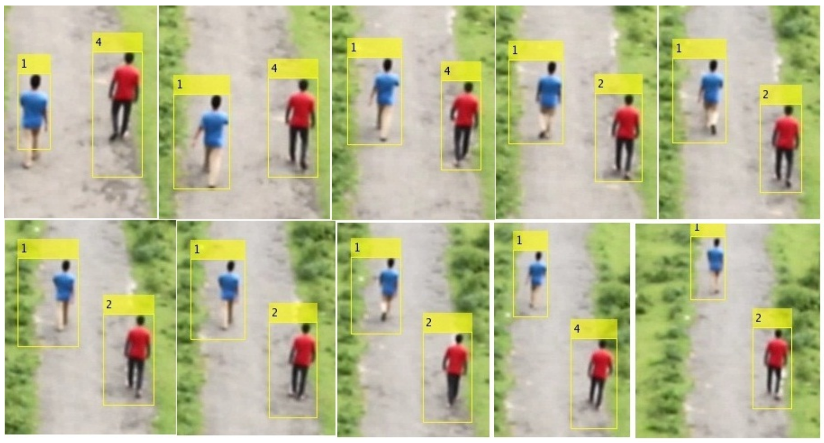


Figure 6.9: Overtake on Left 1

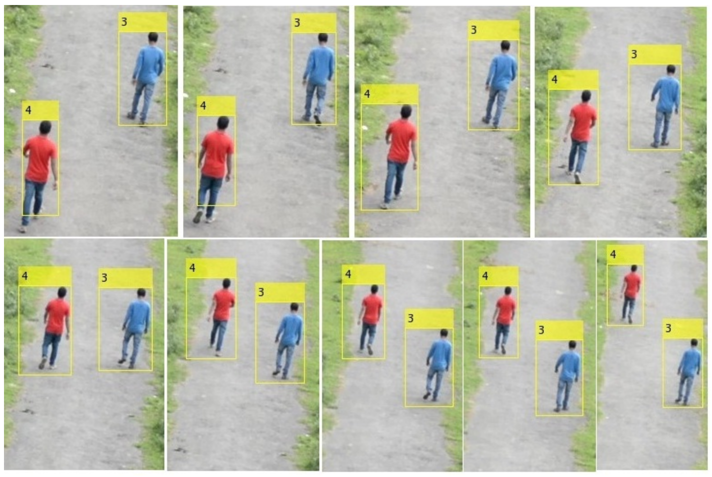


Figure 6.10: Overtake on Left 2

For an analysis,we selected an *overtake* event between persons walking along a road. A number of videos were recorded. In this thesis, we have proposed a language based framework which, at present, does not have constructs for handling certain issues typically encountered in video processing. Presence of noise makes object identification and tracking a complex operation. Problems like non-tracking of objects in certain frames, merging of blobs, occlusion etc. are common. Another issue is that noise varies with video i.e. each recorded video may have different issues to handle. Presently, **QDL** can not handle problems mentioned above. So, we need the objects to be tracked in each frame. Moreover,

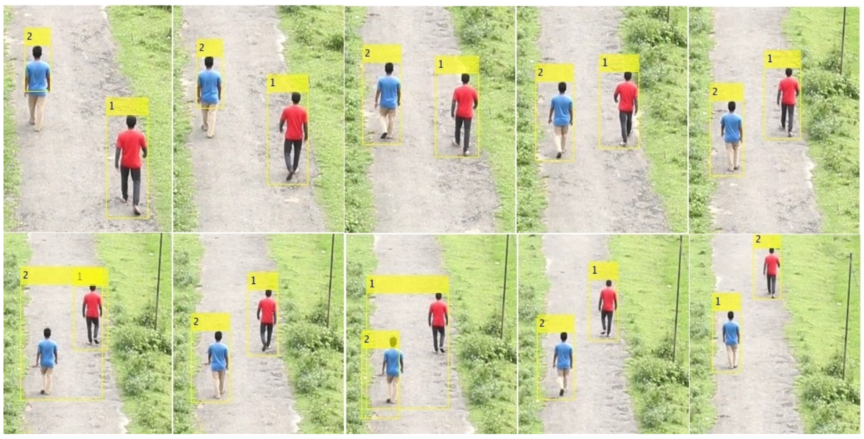


Figure 6.11: Overtake on Right 1

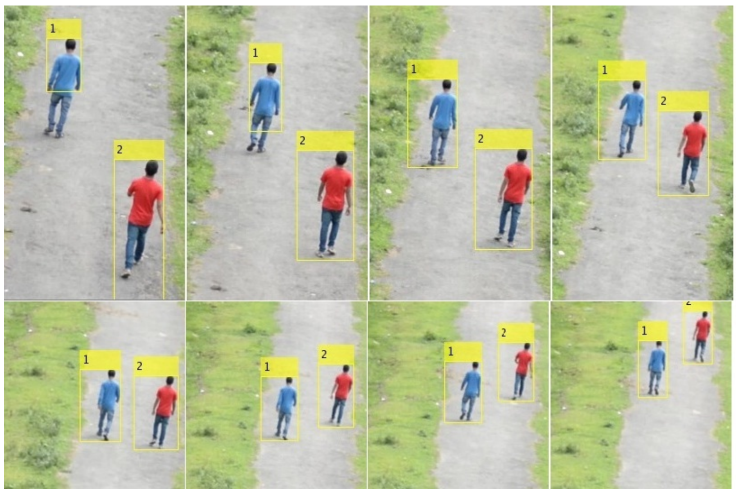


Figure 6.12: Overtake on Right 2

we have selected video where issues like merging of blobs and occlusion are not present.

We have picked four videos for analysis. Out of these, in two, *overtake* has taken place on left. In the remaining two, *overtake* is on right. Videos are of short duration. Each is approximately one minute long with around 1500 frames. Identification of objects has been done using morphological image operations and tracking across frames is done using a constant velocity Kalman filter. A set of tracked images for each video is shown in Figure 6.9 to Figure 6.12. In Figure 6.9, the person wearing the blue shirt has overtaken the red one whereas in Figure 6.10, the red one has overtaken. In Figure 6.11 as well as in Figure 6.12, the red person has done the overtaking on right.

For a video analysis case, we have attempted to use a *definition* of an *overtake* event from cognitive input. This definition is coded as a **QDL** program and then we try to recognise the event using this **QDL** definition.

We need to choose sets of binary qualitative relations for modelling motion parameters. Direction of motion and spatial orientation are identified and used as motion parameters. At first, we started to use the direction and orientation model that introduced in Chapter 3 for modelling these parameters. We computed the relations for the video shown in Figure 6.9. The computed binary motion pattern initially started like:

```
(B&L,Same)(B&L,lr+)(B&L,Same-)(B&L,rl-)(B&L,Same+)
```

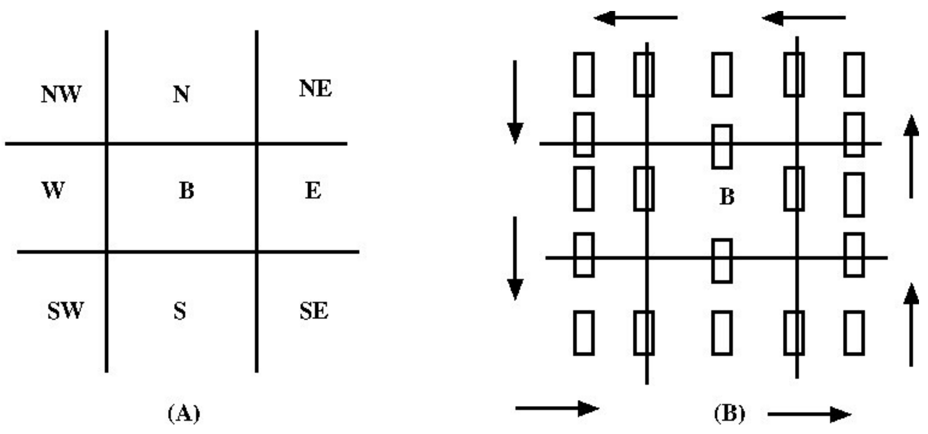


Figure 6.13: Rectangular Cardinal Directions (taken from [2])

This is quite expected as the blue person was behind on left initially. The problem is that direction relations are very sensitive. If at the end of *overtake*, the direction relation changes to, say, *lr+*, the blue person's spatial orientation may be computed to be somewhere in the back (instead of the common perception of being in front). In a learning and recognition set-up, this may not be a problem because in both the cases the event will be recorded in the same way; but such a description will be difficult to comprehend and also difficult to code as a definition in a program. The main issue here is the dependence of orientation on direction in an egocentric FoR and sensitivity of direction in video processing. We decided to explore another model where these two (direction and spatial orientation) are detached. For direction, we used the same model introduced in chapter 3. For orientation, we chose to use the rectangular cardinal directions [2] [32]. This model uses absolute labels like **N** for north, **S** for south etc. In part (A) of Figure 6.13, single tile relations of this model are shown. The reference occupies the tile marked as **B**. Nine relations are shown in the figure. In these relations, the primary object occupies a single tile. Multi-tile relations result when the primary rectangle occupies more than one tile. These relations are assigned labels constructed using names of the spanned tiles. For example, **N:NW:NE** is a relation where the primary rectangle occupies three tiles. These multi-tile relations are listed in Table 6.5. In part (B) of Figure 6.13, conceptual dependencies of rectangular cardinal directions, that are used for handling discontinuity in this implementation, have been shown. When the primary rectangle (the smaller one in the figure) moves in the direction of arrowheads, a relation changes to its conceptual neighbour. For example, if the discontinuity is from **W** to **SE**, then we make it continuous by the sequence: **W**, **W:SW**, **SW**, **S:SW**, **S**, **S:SE**, **SE**. Accordingly, if the discontinuity is from **SE** to **W**, the sequence will be reverse. In writing these relation names in implementation, we have sometimes overlooked the order i.e. we have taken that NE:N and N:NE mean the same relation.

In an intuitive description of an *overtake*, the person doing the overtake will be behind initially. Then, he/she will move either to the right or to the left of the other person and finally he/she will be in the front. Moreover, they should move roughly in similar direction. These terms can be expressed in terms of binary qualitative relations used for modelling the motion parameters. The concept of being *behind* can be represented by the relations **S**,**SW** and **SE**. *Left* can be represented by **W** relation and *right* by **E** relation. The concept of *front*, can be represented by **NW**, **N** and **NE** relations. Concept of similar direction can be represented by *Same*, *Same+* and *Same-*. Considering the sensitivity of these relations, even *lr+* and *rl+* can also be included.The important point here is the fact that at the time of writing the **QDL** definition, we can not foresee the entire binary motion pattern string that is going to be recorded for recognition. Exact matching of these two (definition and recorded string), perhaps, will be very

Table 6.5: Spatial Orientation Relations

Sl. No.	Relation	Sl. No.	Relation
1	S:SW	15	B:W:E
2	S:SE	16	B:N:S
3	N:NW	17	W:NW:SW
4	N:NE	18	E:NE:SE
5	B:W	19	B:S:SW:W
6	B:E	20	B:W:NW:N
7	B:S	21	B:S:E:SE
8	B:N	22	B:N:NE:E
9	W:SW	23	B:S:SW:W:NW:N
10	W:NW	24	B:S:SE:E:NE:N
11	E:SE	25	B:S:SW:W:E:SE
12	E:NE	26	B:W:NW:N:NE:E
13	S:SW:SE	27	B:S:SW:W:NW:N:NE:E:SE
14	N:NW:NE		

difficult to achieve. We try to overcome this problem by using *skip transitions.* Skip transitions try to match salient points in both the strings. Accordingly, a definition of the *overtake* event is coded as:

BMOP Overtake_Left(α,β) {

Reference α;

type MOVE = ((Same | Same+| Same-| lr+ | rl+) • (S | SW | W:SW | S:SW | S:SE) | E:SE | SE)

∘((Same | Same+ | Same- | lr+ | rl+) • (W | NW:W))

∘ ((Same | Same+ | Same- | lr+ | rl+) • (NW | N | NW:N));

var β: MOVE;

}

BMOP Overtake_Right(α,β) {

Reference α;

type MOVE = ((Same | Same+| Same-| lr+ | rl+) • (S | SW | W:SW | S:SW | SE | SE:S | E:SE))

∘((Same | Same+ | Same- | lr+ | rl+) • (E | NE:E)) ∘

((Same | Same+ | Same- | lr+ | rl+) • (NE | N | NE:N));

var β: MOVE;

}

In the above definition, we have defined each basic type expression by taking a concatenation (• operator) of two basic types. Each basic type is expressed as a union of binary qualitative relations. For example, (NE | NW)is a basic type and (Same | Same+ | Same- | lr+ | rl+) is another basic type and their concatenation results in a binary motion pattern. For joining two such binary motion patterns, we have used skip transition operator (∘ operator). Skip transitions were introduced in section 6.1.2. The semantics is that we do not try for a perfect match of strings; instead we take the primitives in the definition and check for their presence in the string to be recognised. The order of occurrence must be same. This is like looking for salient points in the string to be recognised. In **QDL** encoding of the *overtake* pattern, we have used parenthesis to disambiguate the issue operator priority.

In the first video, we first locate the primitive $(Same, W : SW)$. Then, $(Same, W)$ is observed and finally we see the primitive $(Same-, NW)$. This conforms to the definition of *overtake* coded in the **QDL BMOP** definition. In the second video, we first see $(Same+, W : SW)$, then somewhere in the middle $(Same, W)$ is observed and finally $(rl+, NW)$ is recorded. In the third video, we observe $(Same+, S)$, then $(Same+, E)$ is seen at a later point and finally we get to $(Same-, NE)$. In the fourth video, $(Same+, SE)$ is seen, then at a later point, $(Same, E)$ is observed.In this case, the primary does not attain the defined orientation at the end. So, as per definition, the pattern is not recognised in this case. In all these cases, we have used spatial information only; temporal relationships were not used in the implementation.

There are certain points that we would like to discuss about this approach using $\circ$ operator (skip transition operator). Sometimes, a recognised pattern may not be what we actually intended to see. For example, let us take the case of *overtake on left*. A person may go upto west and then he/she may make a right overtake finally ending in north-eastern side. Then also, it will be recorded as an overtake on left. This is a false detection. At present, the $\circ$ operator is not powerful to eliminate such cases. Inclusion of additional operators for performing intricate string matching operations will make the recognition more robust. Another issue is related with continuity networks. We assume that a motion pattern respects spatio-temporal continuity and fill in missing details with the help of conceptual neighbourhood graphs (CNG). The choice of path in a CNG is an important issue. For example, in the overtake case, if the padded sequence takes us to the front artificially, then also false recognition may occur.

In Appendix D, binary motion patterns recorded from each video are listed at serial number 3 with the heading *Recorded Binary Motion Patterns in Left Overtake and Right Overtake using QDL Program Definition.* Under this heading, the string for Figure 6.9 (to be referred to as first video) is listed at serial number (1), the string for Figure 6.10 (to be referred to as second video), at (2), string for Figure 6.11 (third video) is enumerated at (3) and finally the recorded binary motion pattern string for Figure 6.12 (fourth video) is listed at serial number (4).

In the first video, we first locate the primitive $(Same, W : SW)$. Then, $(Same, W)$ is observed and finally we see the primitive $(Same-, NW)$. This conforms to the definition of *overtake* coded in the **QDL BMOP** definition. In the second video, we first see $(Same+, W : SW)$, then somewhere in the middle $(Same, W)$ is observed and finally $(rl+, NW)$ is recorded. In the third video, we observe $(Same+, S)$, then $(Same+, E)$ is seen at a later point and finally we get to $(Same-, NE)$. In the fourth video, $(Same+, SE)$ is seen, then at a later point, $(Same, E)$ is observed.In this case, the primary does not attain the defined orientation. So, as per definition, the pattern is not recognised in this case.

There are certain points that we would like to discuss about this approach using $\circ$ operator. Sometimes, a recognised pattern may not be what we actually intended to see. For example, let us take the case of *overtake on left*. A person may go upto west and then he/she may make a right overtake finally ending in north-eastern side. Then also, it will be recorded as an overtake on left. This is a false detection. At present, the $\circ$ operator is not powerful to eliminate such cases. Inclusion of additional operators for performing intricate string matching operations will make the recognition more robust. Another issue is related with continuity networks. We assume that a motion pattern respects spatio-temporal continuity and fill in missing details with the help of conceptual neighbourhood graphs (CNG). The choice of path in a CNG is an important issue. For example, in the overtake case, if the padded sequence takes us to the front artificially, then also false recognition ma occur. We have chosen paths so that this does not happen.