Parametric Rectangles: A Model for Querying and Animation of Spatiotemporal Databases*

Mengchu Cai, Dinesh Keshwani, Peter Z. Revesz

University of Nebraska-Lincoln, Lincoln, NE 68588, USA

Abstract. We propose *parametric rectangles* — cross products of intervals whose end points are functions of time — as a new data model for representing, querying, and animating spatiotemporal objects with continuous and periodic change. We prove that the model is closed under relational algebra and new spatiotemporal operators and that relational algebra queries can be evaluated in PTIME in the size of any input quadratic non-periodic parametric rectangle database. Finally, we also describe the implementation in our PReSTO database system.

1 Introduction

Many spatiotemporal objects such as clouds, cars, deserts, lakes, planets, ships and tornados change position or shape continuously and also sometimes periodically. Although in the last decade substantial research was done independently in spatial [16, 22] and temporal [18] data modeling, continuously changing objects require new data models that can capture the interdependency of the spatial and temporal extents of these objects.

We introduce a new approach to modeling spatiotemporal objects based on the use of n-dimensional parametric rectangles (or boxes), which are moving objects specified as the cross product of intervals that are parallel to the axes and whose endpoints are functions of time. In our model, each spatiotemporal object is represented by a finite set of parametric rectangles. We provide a PTIME evaluable query language by generalizing the relational algebra to our model. We also add to the query language some spatiotemporal operators like *block, collide, deflect, granulate, scale* and *shift* that facilitate novel applications.

An advantage of our model is the combination of efficient querying with efficient animation of objects. Most other spatiotemporal data models have difficulty combining effectively these two functions (see Section 7). We implemented a system –PReSTO (short for *Parametric Rectangle Spatio-Temporal Objects*)– that proves that the combination is effective in practice as well as theory.

The paper is structured as follows. Section 2 describes the parametric rectangle data model and illustrates how to represent spatiotemporal objects in this model. Section 3 defines the query language by generalizing relational algebra and proves that the evaluation of queries is in PTIME in the size of the database.

* The third author was supported by NSF grant IRI-9625055 and a Gallup Research Professorship. Contacts: revesz@cse.unl.edu and http://cse.unl.edu/~revesz

Section 3 also introduces some new operators for spatiotemporal queries. Section 4 describes the animation approach. Section 5 presents implementation results. Section 6 discusses the mapping from raster-based spatiotemporal objects to 2D parametric rectangles. Finally, Section 7 covers related work.

2 Parametric Rectangle Data Model

2.1 Parametric Rectangles

A n-dimensional rectangle is the cross product of n intervals, each in a different dimension. If the lower and upper bounds of the intervals are functions of time, then the rectangle is called a *parametric rectangle*. Formally, let $\mathbf{R}$ denote the set of real numbers, and $\mathbf{R}^+$ the set of non-negative real numbers.

Definition 1. A n-dimensional *parametric rectangle* r is a tuple:

$$\langle x_1^{[},\ x_1^{]}, \ldots, x_n^{[},\ x_n^{]},\ from,\ to \rangle$$

where for each $i = 1, \ldots, n$, the lower and the upper bounds of an interval in the ith dimension, denoted $x_i^{[}$ and $x_i^{]}$, are functions ($\mathbf{R}^+ \rightarrow \mathbf{R}$) of time t applicable when $t \in [from,\ to]$, and $from$ and to are constants in $\mathbf{R}^+$.

The semantics of r, denoted by $sem(r)$, is a polyhedron in $n+1$ dimensional space defined as follows:

$$sem(r) = \{\ (x_1, \ldots, x_n, t)\ |\ \forall_{1 \leq i \leq n}\ x_i \in [x_i^{[}(t),\ x_i^{]}(t)],\ t \in [from,\ to]\}$$

We call m-degree those parametric rectangles in which the bounds are at most m-degree polynomial functions of time. We also call $m = 1$ and $m = 2$ degree parametric rectangles linear and quadratic, respectively.

Example 1. The semantics of the parametric rectangle $r = \langle 5-t, 10+t, 4-t, 6+t, 0, 3 \rangle$, is the polyhedron in x, y and t dimensions as shown in Figure 1.

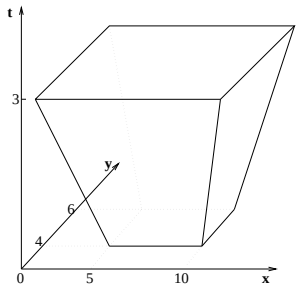


Fig. 1. Semantics of the Parametric Rectangle

Example 2. Suppose that a sail boat which at time $t = 0$ occupies the space $9 \leq x \leq 19$, $10 \leq y \leq 20$ first moves east with a speed of 5 ft/sec until $t = 10$. Then it goes northeast until $t = 20$, with a speed of 10 ft/sec in both the x and y axes. Finally, it goes north with a speed of 8 ft/sec until $t = 25$. We can represent the sail boat by 3 parametric rectangles, as shown in Table 1.

$x^{\lfloor}$	$x^{\rfloor}$	$y^{\lfloor}$	$y^{\rfloor}$	$from$	to
$9 + 5t$	$19 + 5t$	10	20	0	10
$59 + 10(t - 10)$	$69 + 10(t - 10)$	$10 + 10(t - 10)$	$20 + 10(t - 10)$	10	20
159	169	$110 + 8(t - 20)$	$120 + 8(t - 20)$	20	25

Table 1. Parametric rectangles for the sail boat

Example 3. Suppose that a plane drops a bomb at $t = 0$ to hit a target as shown in Figure 2. The bomb can be represented by a quadratic parametric rectangle as shown in Table 2.

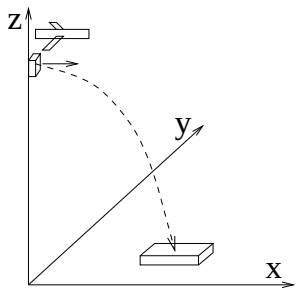


Fig. 2. The trajectory of the bomb

$x^{\lfloor}$	$x^{\rfloor}$	$y^{\lfloor}$	$y^{\rfloor}$	$z^{\lfloor}$	$z^{\rfloor}$	$from$	to
t	$t + 1$	t	t+1	$100 - 9.8t^2$	$102 - 9.8t^2$	0	3.19

Table 2. Parametric rectangles for the bomb

2.2 Periodic Parametric Rectangle

Some spatiotemporal objects can also move periodically, for example, shuttle buses and planets in the solar system. It is not possible to finitely represent

these periodic objects by the parametric rectangles that we have discussed so far. Hence we extend the parametric rectangle concept to *periodic parametric rectangle* by adding a parameter as follows:

Definition 2. A *periodic parametric rectangle* r is a tuple of the form

$$\langle x_1^{[},\ x_1^{]}, \ldots, x_n^{[},\ x_n^{]}, from, to, period \rangle$$

where *period* is a non-negative integer constant such that $period = 0$ or $period \geq (to - from)$, and all the other parameters are as in Definition 1.

The semantics of r is the semantics of a set of parametric rectangles $r_0, r_1, \ldots$ such that $r_0 \equiv \langle x_1^{[},\ x_1^{]}, \ldots, x_n^{[},\ x_n^{]}, from, to \rangle$ and r_k exists between $from + k * period$ and $to + k * period$. Further, r_k at time t is identical to r_0 at time $t - k * period$. More precisely:

$$sem(r) \equiv sem(\bigcup_{k \geq 0} \{\langle\ x_1^{[}(t - k * period),\ x_1^{]}(t - k * period), \ldots, x_n^{[}(t - k * period), \\ x_n^{]}(t - k * period),\ from + k * period,\ to + k * period\ \rangle\})$$

In particular, when $period = 0$, then $sem(r) \equiv sem(\{r_0\})$.

We call *non-periodic* parametric rectangles those in which $period = 0$.

Example 4. Suppose there is a shuttle bus running every 30 minutes around a route as shown in Figure 3. We can represent it by 6 periodic parametric

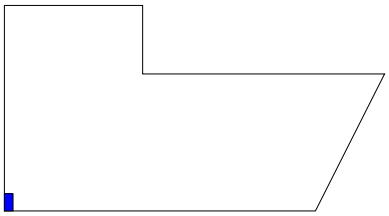


Fig. 3. Route of shuttle bus

rectangles, in a relation *shuttle*, as shown in Table 3.

2.3 Parametric Rectangle Database

Definition 3. Let $\mathcal{P}$ denote the set of all parametric rectangles, and $\mathcal{U}$ a finite set of attributes. Let $A_1, \ldots, A_k$ be elements of $\mathcal{U}$. For each A_i there is an attribute domain $Dom(A_i)$ associated with it. Let $\square$ be a special attribute such that $Dom(\square) = \mathcal{P}$ and $\square.x_i$ the interval for the ith dimension. A *parametric rectangle tuple* r on $\square, A_1, \ldots, A_k$ is a tuple in $Dom(\square) \times Dom(A_1) \times \ldots \times Dom(A_k)$.

$x^{\lfloor}$	$x^{\rfloor}$	$y^{\lfloor}$	$y^{\rfloor}$	$from$	to	$period$
0	1	$6t$	$6t+1$	0	5	30
$5(t-5)$	$1+5(t-5)$	30	31	5	9	30
20	21	$30-4(t-9)$	$31-4(t-9)$	9	11.5	30
$20+6(t-11.5)$	$21+6(t-11.5)$	20	21	11.5	19	30
$65-5(t-19)$	$66-5(t-19)$	$20-4(t-21)$	$21-4(t-21)$	19	24	30
$40-8(t-24)$	$41-8(t-24)$	0	1	24	29	30

Table 3. Periodic parametric rectangles for the *shuttle* relation

The semantics of a parametric rectangle tuple $r = \langle r_1, a_1, \ldots, a_k \rangle$ is the cross product of the semantics of the parametric rectangle r_1 with the values $a_1, \ldots, a_k$.

A *parametric rectangle relation* is a finite set R of parametric rectangle tuples. The semantics of R is the union of the semantics of each tuple in R. An instantiation of R at time t_1, denoted by $R(t_1)$ is the union of instantiations of all tuples in R at time t_1. A *parametric rectangle database* is a finite set of parametric rectangle relations. In the following, by relations we mean parametric rectangle relations.

3 Querying Parametric Rectangle Databases

3.1 Relational Algebra for the Parametric Rectangle Model

In this section we extend relational algebra to the parametric rectangle database model.

Definition 4. Let R_1 and R_2 be two relations over the same set of attributes $\Box, A_1, \ldots, A_k$.

- *projection* ($\hat{\pi}_Y$) Let $Y \subseteq \{\Box, \Box.x_1, \ldots, \Box.x_n, A_1, \ldots, A_k\}$. The projection of R_1 on Y, denoted by $\hat{\pi}_Y(R_1)$, is a relation R over the attributes Y such that

$$R = \{r : \exists r_1 \in R_1, \forall A \in Y, \text{ the values of } A \text{ in } r \text{ and } r_1 \text{ are equal}\}$$

- *selection* ($\hat{\sigma}$) Let E be the conjunction of a set of comparison predicates in the form $A\,\theta\,c$ or $A\,\theta\,B$, where c is a constant, $A, B \in \{A_1, \ldots, A_k\}$, $\theta \in \{=, <, <=, >, >=\}$ and A,B are distinct. The selection $\hat{\sigma}_E(R_1)$ is a relation R containing the parametric rectangle tuples in R_1 whose attribute values satisfy E.
- *intersection* ($\hat{\cap}$) The intersection of R_1, R_2, denoted by $R_1 \hat{\cap} R_2$, is a relation R over attributes $A_1, \ldots, A_k$ such that

$$sem(R) = sem(R_1) \cap sem(R_2)$$

– *union* ($\hat{\cup}$) The union of R_1, R_2, denoted by $R_1 \hat{\cup} R_2$, is a relation R over attributes $A_1, \ldots, A_k$ that contains all tuples in R_1 and R_2.

$$sem(R) = sem(R_1) \cup sem(R_2)$$

– *difference* ($\hat{-}$) The difference of R_1, R_2, denoted by $R_1 \hat{-} R_2$, is a relation R over attributes $A_1, \ldots, A_k$ such that

$$sem(R) = sem(R_1) \setminus sem(R_2)$$

– *complement* ($\hat{\neg}$) Let R be a relation with only the $\square$ attribute. The complement of R, denoted by $\hat{\neg} R$ is also a parametric rectangle relation R' with the $\square$ attribute, such that

$$sem(R') = \{(x, y, t) : (x, y, t) \notin sem(R)\}$$

The unary operators have higher precedence than the binary operators. Intersection ($\hat{\cap}$) has higher precedence than union ($\hat{\cup}$) and difference ($\hat{-}$). A *relational algebra expression* over parametric rectangle databases is built up in the standard way, using the operators in Definition 4.

Theorem 1. *For any fixed n, any relational algebra expression can be evaluated in PTIME in the size of the input quadratic non-periodic parametric rectangle database, where each parametric rectangle is within the same n dimensions.* $\square$

Theorem 2. *Linear periodic parametric rectangle databases are closed under the relational algebra operators.* $\square$

Example 5. Suppose there is a ship represented by the relation *ship*, and a torpedo has just been fired towards the ship. The torpedo is represented by the relation *torpedo*. The relations are shown in Table 4.

ship

$x^{\lfloor}$	$x^{\rfloor}$	$y^{\lfloor}$	$y^{\rfloor}$	*from*	*to*	*period*
$20+t$	$30+t$	20	25	0	25	0

torpedo

$x^{\lfloor}$	$x^{\rfloor}$	$y^{\lfloor}$	$y^{\rfloor}$	*from*	*to*	*period*
45	48	$45-t$	$51-t$	0	25	0

hit

$x^{\lfloor}$	$x^{\rfloor}$	$y^{\lfloor}$	$y^{\rfloor}$	*from*	*to*	*period*
45	48	$45-t$	25	20	25	0

Table 4. Parametric rectangles for the *ship* , *torpedo* and *hit* relations

Query: "Will the torpedo hit the ship?"
ship $\hat{\cap}$ *torpedo*
We can evaluate the intersection and represent it by a parametric rectangle relation *hit* as shown in Table 4.

Example 6. Suppose that the relation *clouds* has an attribute *humidity* which indicates the humidity of the cloud, and the clouds with humidity greater than 60 percent are called rain clouds.
Query: "Which of the clouds are rain clouds?"
$\hat{\sigma}_{humidity \geq 60}(clouds)$
Let *region* be a relation with an additional attribute *temperature*. Suppose that it rains when a rain cloud moves into a region where the temperature is between 0 and 20 degrees.
Query : "Which region is most likely to get rain?"
$\hat{\pi}_{\square}(\hat{\sigma}_{humidity \geq 60}(clouds)) \hat{\cap} \hat{\pi}_{\square}(\hat{\sigma}_{(temperature \geq 0 \wedge temperature \leq 20)}(region))$

Example 7. Consider the *shuttle* relation in Example 4. Let the relation *bus_stop* represent a bus-stop along the route of the shuttle bus. Suppose the relation *passenger* represents a man walking toward the bus-stop during some part of the day.
Query: "Will the passenger be able to catch the bus?"
$(shuttle \hat{\cap} bus\_stop) \hat{\cap} passenger$

Example 8. The nine planets of the solar system revolve around the sun in periodic orbits. They are represented by 3D periodic parametric rectangle relations $Mercury$, $Venus$, ..., $Pluto$. The motion of a comet is represented by the periodic relation *comet*.
Query: "Will the comet ever collide with any of the planets?"
$(Mercury \hat{\cup} Venus \hat{\cup} \ldots \hat{\cup} Pluto) \hat{\cap} comet)$

3.2 Block Operator

Some spatiotemporal applications need operators that are not provided in relational algebra. Let us consider the following example.

Example 9. There is a growing forest fire whose shape is approximated by the parametric rectangle tuple $\langle 4, 4, 20 + t, 20 + 0.5t, 0, 20, 0\rangle$. A plane drops foam to extinguish the fire. Let us assume that the foam is represented by the tuple $\langle 20, 25, 20 - t, 25, 0, 20, 0\rangle$. Let $fire$ and $foam$ be relations, containing the above tuples. Suppose that parts of the fire are extinguished when they meet the foam. Other parts continue to grow as before.
Query: "At time $t = 15$, what part of the forest is still on fire?"

Figure 4 (left) shows the instantiation of $fire - foam$ at time $t = 15$. The result does not correctly answer the query, because it fails to consider that some parts of the fire stop growing when they are extinguished.

To allow us to answer queries like the one above, we introduce a new operator called *block* that applies only to non-periodic 2D parametric rectangle relations. Note that the semantics of such relations allows us to view each non-periodic 2D parametric rectangle as a set of moving points (x, y) where x and y are linear functions of t.

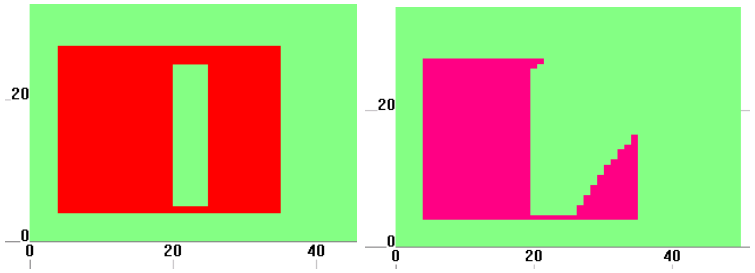


Fig. 4. The forest fire example with difference (left) and block (right)

Definition 5. Let R_1 and R_2 be two relations. R_1 *block* R_2 till some time t_k, denoted by $block(R_1, R_2, t_k)$, is the instantiation at t_k of the set of moving points in R_2 that did not intersect with R_1 any time before or at t_k. Formally,

$$\begin{aligned} block(R_1, R_2, t_k) = \{\ & ((x(t_k),\ y(t_k)) \mid \exists\ \langle x^{\lfloor}, x^{\rfloor}, y^{\lfloor}, y^{\rfloor}, from, to\rangle \in R_2, \\ & 0 \le \alpha \le 1,\ 0 \le \beta \le 1, \\ & x = \alpha x^{\lfloor} + (1-\alpha)x^{\rfloor},\ \ y = \beta y^{\lfloor} + (1-\beta)y^{\rfloor}, \\ & \nexists t'\ from \le t' \le t_k \le to,\ (x(t'), y(t')) \in R_1(t')\} \end{aligned}$$

Here is an efficient algorithm to approximate the result of *block* operation by recursion. In the algorithm we use a threshold *max_depth* to guarantee the termination of the recursion. Let R_1 and R_2 be the two input relations.

Procedure block($R_1,\ R_2,\ t_k$)
  for each tuple $r_2 \in R_2$ **do**
    $R' = R' \cup blockrect(R_1, r_2, t_k, 0)$
  return R'

Procedure blockrect (R_1, r_2, t_k, d)
  if $\{r_2\} \hat{\cap} R_1 = \emptyset$ **then return** $\{r_2\}$ at t_k
  else if at t_k $\{r_2\} \subseteq R_1$ **then return** $\emptyset$.
  else if $d < max\_depth$ **then** partition r_2 into quadrants $r_{21},\ r_{22},\ r_{23},\ r_{24}$
    return $blockrect(R_1, r_{21}, t_k, d+1) \cup \ldots \cup blockrect(R_1, r_{24}, t_k, d+1)$

Now the forest fire query in Example 9 (see also Figure 4) can be written as:

$$block(foam, fire, 15)$$

Example 10. Suppose that *tornado* is a relation that represents the movement of a tornado in an area represented by the relation *region*.
Query: "What is the trajectory of the tornado at time t = 20?"
$block(tornado, region, 20)$
The result of the query, evaluated in the PReSTO system, is shown in Figure 5.

Fig. 5. Result of $block(tornado, region, 20)$ in the PReSTO system

3.3 Collide Operator

Another possible interaction between spatiotemporal objects is collision. Here we only consider the elastic collision between spatiotemporal objects which do not change their extent, that is, in them $x^{\lfloor}$ and $x^{\rfloor}$ and also $y^{\lfloor}$ and $y^{\rfloor}$ have the same coefficients of t. (An elastic collision is one in which both momentum and energy are conserved.) Suppose two objects are represented by the parametric rectangles r_1 and r_2 with an attribute $mass$. We define the collision of r_1 and r_2, denoted by $collide(r_1, r_2)$, as follows:

$$collide(r_1, r_2) = \begin{cases} \{r_1,\ r_2\} & \text{if } r_1 \cap r_2 = \emptyset \\ \{r_1',\ r_1'',\ r_2',\ r_2''\} & \text{otherwise} \end{cases}$$

where r_1' and r_2' represent r_1 and r_2 before collision and r_1'' and r_2'' represent them after the collision. It is easy to see that all parameters of r_1' and r_2', except to, are the same as those of r_1 and r_2 respectively. r_1'' and r_2'' are computed by the following algorithm. r_1 and r_2 can be viewed as spherical masses, with the x and y components of each one's velocity equal to its coefficients of t in $x^{\lfloor}$ and $y^{\lfloor}$ respectively.

1. Compute the time of the collision $t_c = \min_{r \in r_1 \cap r_2}(r.from)$, then $r_1''.from = r_2''.from = t_c$.
2. Let LC be the line joining the centers of the two objects at $t = t_c$. Decomposite the velocity of r_1 and r_2 along LC and the direction orthogonal to LC as shown in Figure 6.
3. Consider the collision as a "head-on" collision [5] between the objects along the LC. Compute the velocities of r_1 and r_2 along the LC after the collision by momentum and energy conservation equations. The velocities of r_1 and r_2 along the direction orthogonal to the LC do not change.
4. Let v_x and v_y denote the components of the velocity of r_1 after collision on the x and y axes, respectively.

$r_1''.x^{\lfloor} = v_x(t - t_c) + r_1.x^{\lfloor}(t_c) \quad r_1''.x^{\rfloor} = v_x(t - t_c) + r_1.x^{\rfloor}(t_c)$
$r_1''.y^{\lfloor} = v_y(t - t_c) + r_1.y^{\lfloor}(t_c) \quad r_1''.y^{\rfloor} = v_y(t - t_c) + r_1.y^{\rfloor}(t_c)$
The bound functions of r_2'' are computed similarly.

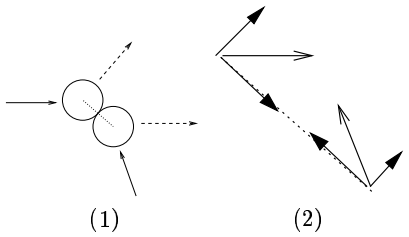


Fig. 6. (1) Line joining centers (LC) at t_c (dotted) (2) Collision along LC

3.4 Deflect Operator

Given a linear 2D parametric rectangle r, the deflect operator, $deflect(r,\ \theta)$, changes the direction of r (that is, the direction of its center point) counter-clockwise by the angle θ. The resulting parametric rectangle r' is on the same interval $[from, to]$ as r and can be computed as follows. At $t = from$, r and r' are the same, and at $t > from$ they have the same width and height but different locations. Then find $(v_x,\ v_y)$, the speed of the center of r along the x and y axes, and φ, the angle between the direction of r and the x-axis. The speed of the center of r' is $v_x' = \sqrt{v_x^2 + v_y^2}\cos(\varphi + \theta),\ v_y' = \sqrt{v_x^2 + v_y^2}\sin(\varphi + \theta)$. From this we can specify the bounds of r'.

For example, if $r = \langle 4t - 10,\ 6t + 10, -t - 5,\ t + 5,\ 0,\ 10\rangle$, then the operation $deflect(r, tan^{-1}(\frac{3}{4}))$ gives $r' = \langle 3t - 10,\ 5t + 10, 2t - 5,\ 4t + 5,\ 0,\ 10\rangle$.

3.5 Scale Operator

For an n-dimensional parametric rectangle r, the scale operator, denoted by $scale(r, i, \alpha)$, will change $x_i^{\lfloor}$ into $x_i^{\lfloor} - \frac{\alpha}{2}(\Delta x_i)$ and $x_i^{\rfloor}$ into $x_i^{\rfloor} + \frac{\alpha}{2}(\Delta x_i)$ where $\Delta x_i = x_i^{\rfloor} - x_i^{\lfloor}$.

3.6 Temporal Operators

Given a parametric rectangle tuple $r = \langle x_1^{\lfloor},\ x_1^{\rfloor}, \ldots, x_n^{\lfloor},\ x_n^{\rfloor},\ from,\ to\rangle$, the operator $shift(r, \beta)$ sets the time back β units. The result of this operator is the tuple: $\langle x_1^{\lfloor}(t+\beta),\ x_1^{\rfloor}(t+\beta), \ldots, x_n^{\lfloor}(t+\beta),\ x_n^{\rfloor}(t+\beta),\ from - \beta,\ to - \beta\rangle$. We also define the operator $granulate(r, \alpha)$ to change the time units to be *alpha* times the current one. The result is: $\langle x_1^{\lfloor}(\frac{t}{\alpha}),\ x_1^{\rfloor}(\frac{t}{\alpha}), \ldots, x_n^{\lfloor}(\frac{t}{\alpha}),\ x_n^{\rfloor}(\frac{t}{\alpha}),\ \alpha from,\ \alpha to\rangle$.

4 Animation of Parametric Rectangle Databases

By animation, we mean a display of the instantiation of the relations in the database at successive time instants. Parametric rectangle databases can be easily animated. In order to display a relation at a time instant t_i, we need to first perform a check on each of its parametric rectangle tuples as follows:

If the tuple has $period = 0$, $t'_i = t_i$

Otherwise, since the tuple is periodic, we need to compute t'_i such that

$$t'_i = ti - \lfloor \frac{t_i - from}{period} \rfloor \cdot period$$

Now we can check if $from \leq t'_i \leq to$. If so, we instantiate the variable t by t'_i and obtain a rectangle defined by $\langle x^{\lfloor}(t'_i), x^{\rfloor}(t'_i), y^{\lfloor}(t'_i), y^{\rfloor}(t'_i)\rangle$.

Proceeding in this manner, we obtain a set of rectangles corresponding to the relation at time t_i. These can be displayed using standard graphics routines.

5 Implementation Results

We implemented the query language and animation algorithm in PReSTO (short for *Parametric Rectangle Spatio-Temporal Objects*) using Microsoft Visual C++.

Table 5 shows the execution times for the evaluation of three examples from Section 3. The torpedo-ship example was extended and the solar system example was projected into 2D. The results are shown in . The PReSTO system ran in Windows NT, on a 266 MHz Pentium II PC with 64 MB RAM.

Example	Number of Tuples	Running Time (milliseconds)
torpedo-ship (Ex. 5 extended)	12	60
shuttle bus (Ex. 7)	35	200
solar system (Ex. 8 in 2D)	513	7500

Table 5. Query Evaluation Times

We provided a graphical user interface through which the user can specify the following parameters: the name of the parametric rectangle relation, the initial time, the time period, the number of time-steps and the minimum delay time, which controls the speed of the animation. Each snapshot of the relation is displayed as soon as the animation algorithm returns the corner vertices of the rectangles to be displayed. The animation is extremely fast. Hence we did not include animation time as a part of the running time.

To demonstrate the animation capability of PReSTO, we have included two snapshots of a cloud moving over the United States, as shown in Figure 7.

Fig. 7. Cloud at $t = 25$ (left) and $t = 182$ (right) in the PReSTO system

6 Mapping of Spatiotemporal Objects to Parametric Rectangles

Given the initial and final raster snapshots of a spatiotemporal object, we present an algorithm which runs in time linear in the size of the raster images, to approximate its motion by parametric rectangles.

For simplicity, let us assume that the given raster images are square, say $Init\_Pic[i, i]$, $Final\_Pic[f, f]$ where i, f are powers of 2. Also, we know the coordinates of the left lower corners of both images.

Without loss of generality, suppose $i \geq f$.

Procedure mapping
Input: $Init\_Pic[i, i]$, $Final\_Pic[f, f]$, t_{init}, t_{final}
Output: Parametric Rectangle Relation

1. We rectangulate the raster images as follows (See Figure 8):
   (a) Take the bigger image (here $Init\_Pic[i, i]$) and divide it into squares of size $r \times r$, where r equals $\frac{2i}{f}$. The number of such parametric rectangles will be $(\frac{i}{r})^2$. The number of rows of squares will be $\frac{i}{r}$.
   (b) For each of the squares, if the majority of the raster points of the square are within the object, mark the square as 'On'.
   (c) For each row of squares, combine adjacent squares that are marked 'On', to get one or more rectangles in each row.
   (d) Repeat Steps (a), (b) and (c) for the smaller image (here $Final\_Pic[f, f]$), using $r = 2$.

   Note: Step 1 will ensure that both $Init\_Pic$ and $Final\_Pic$ have the same number of rows of rectangles.
2. Next, we pair rectangles of the initial image with those of the final image. Assume that Step 1 results in the structures $Init[]$ and $Final[]$, where $Init[k]$ and $Final[k]$ refer to the k^{th} row of rectangles in the initial and final images respectively. Repeat the following for each k:

Let m and n be the number of rectangles in $Init[k]$ and $Final[k]$ respectively. Let us assume $Init[k]$ has m horizontal rectangles, and $Final[k]$ has n horizontal rectangles

if $m = n$ then we simply pair the j^{th} rectangle of $Init[k]$, with the j^{th} rectangle of $Final[k]$ for $1 \leq j \leq m$.

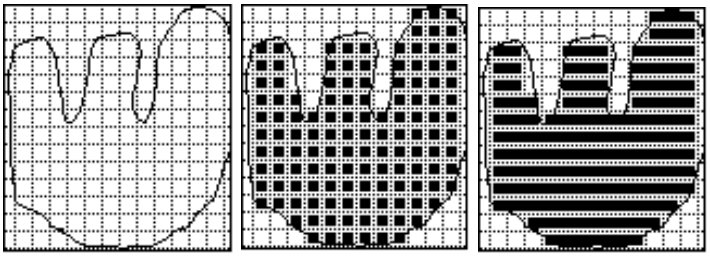
else if $m < n$ then, let $d = \lfloor \frac{m}{n} \rfloor$. We pair the first rectangle of $Init[k]$ with the first d rectangles of $Final[k]$, the second rectangle with the second d rectangles and so on until we come to the m^{th} rectangle of $Init[k]$, which we pair with all the remaining unpaired rectangles of $Final[k]$.

else if $m > n$, we can do the pairing similarly to the above.

3. Finally we compute the output relation, which consists of parametric rectangles corresponding to each pair of rectangles.
   (a) Suppose the rectangles in a pair are $[x_1, x_2] \times [y_1, y_2]$ and $[x'_1, x'_2] \times [y'_1, y'_2]$, where the first one is from $Init[]$ and the second from $Final[]$.
   (b) Let the parametric rectangle corresponding to this pair be

$$\langle x^{\lfloor}, x^{\rfloor}, y^{\lfloor}, y^{\rfloor}, from, to \rangle$$

Clearly, $from = t_{init}$ and $to = t_{final}$. We know that $x^{\lfloor}$ is a linear function of t, say $at + b$. This implies that
$x^{\lfloor}(t_{init}) = a\ t_{init} + b$ and $x^{\lfloor}(t_{final}) = a\ t_{final} + b$
We also know that $x^{\lfloor}(t_{init}) = x_1$ and $f_l(t_{final}) = x'_1$. Since we know x_1, x'_1, t_{init} and t_{final}, we can determine $x^{\lfloor}$. Similarly, we can determine $x^{\rfloor}, y^{\lfloor}$ and $y^{\rfloor}$.



Step 1(a) Step 1(b) Step 1(c)

Fig. 8. Rectangulation of the image

Note: In the above algorithm, if the raster images dimensions are not powers of two, then we can use standard algorithms [10] to stretch or shrink a given image to a square image with dimension as the nearest power of 2. Our mapping provides a highly accurate approximation for a reasonable time interval. If the approximation is needed over larger intervals, it would be necessary to provide more snapshots during the interval, and call the algorithm for each sub-interval.

7 Related work

Constraint databases with rational linear or real polynomial constraints can be used to represent spatiotemporal objects with continuous change [13, 17]. They can be queried by relational algebra, which is quite powerful, although it cannot express some queries like parity and transitive closure [1].

The parametric 2-spaghetti data model [7] generalizes the 2-spaghetti data model [16] by allowing the corner vertices to be represented as linear functions of time. The parametric 2-spaghetti data model cannot represent polynomial (Example 3) and periodic (Example 4) parametric rectangles and cannot be queried by relational algebra, because it is not closed under intersection [7]. However, this model can represent linear constraint databases over two spatial and one temporal dimension [7] and can be used to animate such databases [6].

[8] represents spatiotemporal objects by a composition of a reference spatial extent at some reference time and various types of transformation functions. In this model, it is easy to obtain any snapshot of a spatiotemporal object, making animation straightforward. However, in some cases this model also may not be closed under intersection [8], and the query complexity may be high.

[9] defines continuously moving points and regions and an extended SQL query language on these objects. However, changing of object shape (shrinking and growing) and animation are not considered in this model.

[21] can only represent spatiotemporal objects with discrete change. This model can be queried by an extended relational algebra. [11] proposes another spatiotemporal data model based on constraints in which, like in [21], only discrete change can be modeled. An SQL-based query language is also presented.

There are many data models to represent and query only temporal or only spatial data. For example, [2, 12, 19] can represent temporal objects with discrete change and periods. These models can be queried by either relational algebra or Datalog. Some other purely temporal or purely spatial data models are reviewed in [16, 18, 22]. We also did not deal with issues like visual query languages [3], indefinite information [15], query processing [4], nearest neighbor [14] and approximate queries [20]. We are currently investigating extensions in these directions.

References

1. M. Benedikt, G. Dong, L. Libkin, and L. Wong. Relational Expressive Power of Constraint Query Languages. *Journal of the ACM*, 45:1, pp. 1-34, 1998.
2. E. Bertino, C. Bettini, E. Ferrari, and P. Samarati. An Access Control Model Supporting Periodicity Constraints and Temporal Reasoning. *ACM Transactions on Database Systems*, 23:3, pp. 231-285, 1998.
3. C. Bonhomme, C. Trépied, M-A. Aufaure, R. Laurini. A Visual Language for Querying Spatio-Temporal Databases. *Proc. 7th ACM Symposium on Geographic Information Systems*, 34-39, Kansas City, MO, November 1999.
4. A. Brodsky, J. Jaffar, M. Maher. Towards Practical Query Evaluation for Constraint Databases. *Constraints*, 2:3-4, pp. 279-304, 1997.

5. M. Casco Associates. *Linear Momentum and Collisions: A Mechanics Course*, available at `http://www.mcasco.com/p1lmc.html`.
6. J. Chomicki, Y. Liu, and P.Z. Revesz. Animating Spatiotemporal Constraint Databases. In: *Proc. Workshop on Spatio-Temporal Database Management*, Springer-Verlag LNCS 1678, pp. 224-241, Edinburgh, Scotland, Sept. 1999.
7. J. Chomicki and P.Z. Revesz. Constraint-based Interoperability of Spatiotemporal Databases, *Geoinformatica*, 3:3, 1999. (Preliminary version In: *Proc. International Symposium on Large Spatial Databases*, Springer-Verlag LNCS 1262, pp. 142-161, Berlin, Germany, July 1997.)
8. J. Chomicki and P.Z. Revesz. A Geometric Framework for Specifying Spatiotemporal Objects. In: *Proc. International Workshop on Time Representation and Reasoning*, pp. 41-46, Orlando, Florida, May 1999.
9. M. Erwig, R.H. Güting, M.M. Schneider and M. Vazirgiannis. Spatio-Temporal Data Types: An Approach to Modeling and Querying Moving Objects in Databases. In: *Proc. ACM Symposium on Geographic Information Systems*, November 1998.
10. R. Gonzalez, R. Woods. *Digital Image Processing*, Addison-Wesley, 1998.
11. S. Grumbach, P. Rigaux, and L. Segoufin. Spatio-Temporal Data Handling with Constraints. In: *Proc. 6th ACM Symposium on Geographic Information Systems*, November 1998.
12. F. Kabanza, J-M. Stevenne, and P. Wolper. Handling Infinite Temporal Data. *Journal of Computer and System Sciences*, 51:1, pp. 1-25, 1995.
13. P. C. Kanellakis, G. M. Kuper, and P.Z. Revesz. Constraint Query Languages. *Journal of Computer and System Sciences*, 51:1, pp. 26-52, 1995.
14. G. Kollios, D. Gunopulos, and V.J. Tsotras. Nearest Neighbor Queries in a Mobile Environment. In: *Proc. Workshop on Spatio-Temporal Database Management*, Springer-Verlag LNCS 1678, pp. 119-134, Edinburgh, Scotland, September 1999.
15. M. Koubarakis and S. Skiadopoulos. Tractable Query Answering in Indefinite Constraint Databases: Basic Results and Applications to Querying Spatio-Temporal Information. In: *Proc. Workshop on Spatio-Temporal Database Management*, Springer-Verlag LNCS 1678, pp. 204-223, Edinburgh, Scotland, September 1999.
16. R. Laurini and D. Thompson. *Fundamentals of Spatial Information Systems*. Academic Press, 1992.
17. P.Z. Revesz. *Introduction to Constraint Databases*. Springer-Verlag, 1999.
18. A. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R.T. Snodgrass. *Temporal Databases: Theory, Design, and Implementation*. Benjamin/Cummings Inc., Redwood City, California, 1993.
19. D. Toman, J. Chomicki, and D.S. Rogers. Datalog with Integer Periodicity Constraints. *Proc. International Symposium on Logic Programming*, pp. 189-203, Ithaca, New York, 1994.
20. D. Vasilis, M. Christos, and S. Spiros. A Provably Efficient Computational Model For Approximate Spatiotemporal Retrieval. In: *Proc. 7th ACM Symposium on Geographic Information Systems*, pp. 40-46, Kansas City, Missouri, November 1999.
21. M. F. Worboys. A Unified Model for Spatial and Temporal Information. *Computer Journal*, 37:1, pp. 25-34, 1994.
22. M. F. Worboys. *GIS: A Computing Perspective*, Taylor & Francis, 1995.