

3.2.4 Anfragen an die nahe Zukunft

– Beispiel:

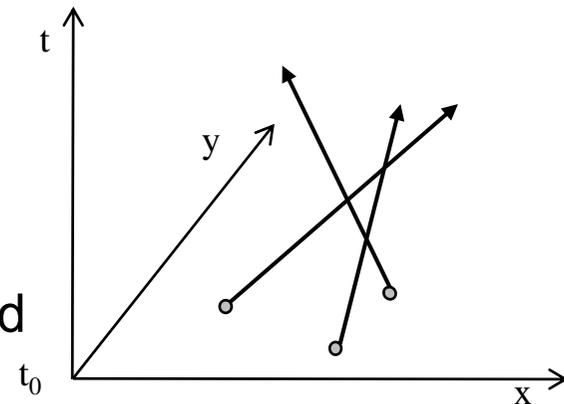
- Wer befindet sich in der nächsten Stunde in Region A?
- Welche Objekte treffen in der nächsten Stunde aufeinander?

– Eigenschaften:

- Zukünftige Bewegung von Objekten wird über Geschwindigkeitsvektoren vorhergesagt.
- Die Vorhersage ist gültig bis zu einem gewissen Zeithorizont.

– Darstellung der nahen Zukunft

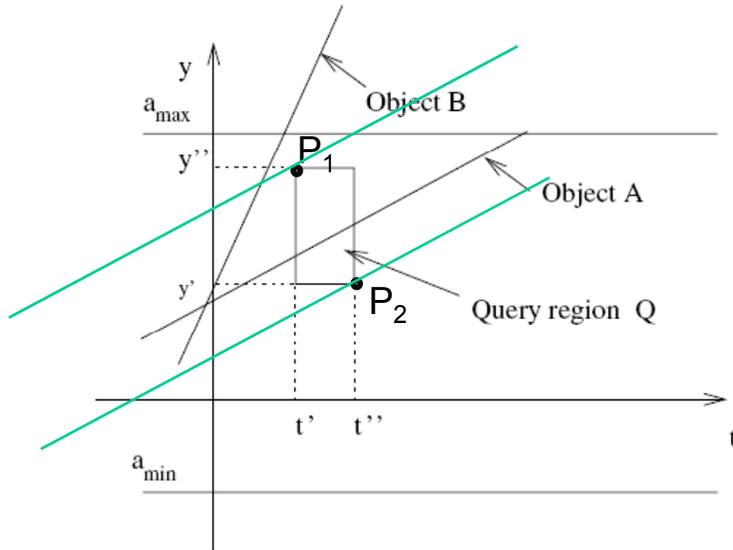
- Zeit wird wieder als zusätzliche Dimension betrachtet.
- Zukünftige Bewegung über aktuelle Position (Position zum Zeitpunkt t_0) und **linearer** Bewegungsrichtung (gültig bis zu einem bestimmten Zeithorizont oder der nächsten Aktualisierung der Position (Update)).



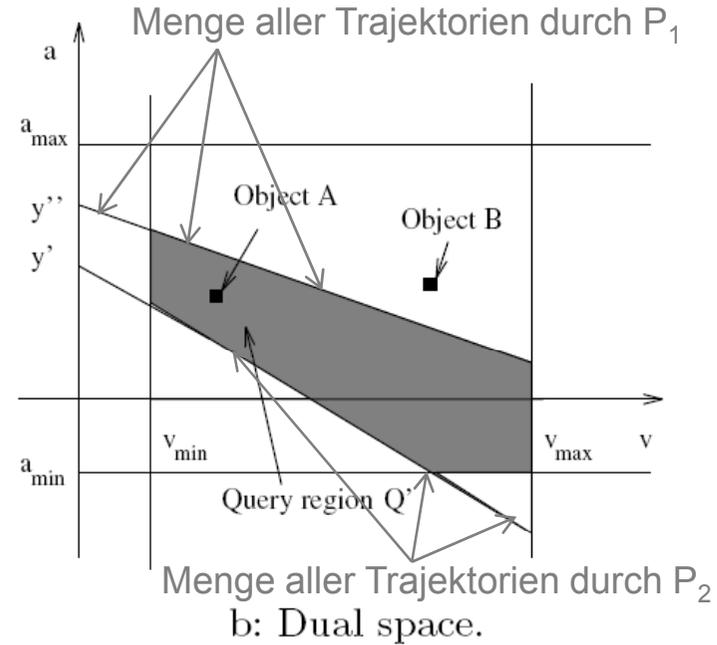
– Effiziente Verwaltung der Zukunft

- Indexstrukturen die lineare Bewegungen effizient Verwalten können:
 - Indexstrukturen basierend auf Dual-Raum Transformation:
Beispiele: **STRIPES** [PCC04], MB-Index [EEK03].
 - Indexstrukturen basierend auf zeit-parametrisierte Seitenregionen:
Beispiele: **TPR-Baum** [SJLL00] und Varianten.
 - Nur für Anfragen in die Zukunft, keine Verwaltung der Vergangenheit.
- Dual-Raum Transformation
 - Idee:
 - Transformation aller Trajektorien der Form $y(t) = vt + a$ in den Dual-Raum $V \times A$ (V = Menge aller Geschwindigkeiten, A = Menge aller Positionen im (nativen) Raum)
Trajektorie $y(t) = vt + a \rightarrow$ Punkt (v,a) .
 - Betrachtung von nur positiven Geschwindigkeiten. Dazu werden alle Geschwindigkeiten vorab normalisiert mit $v' = v + v_{\max} \Rightarrow v' \in [0, 2v_{\max}]$.
 - Verwaltung der Trajektorien im Dual-Raum über Punktzugriffsstrukturen (z.B: Quadtree, R-Baum, etc.)
 - Anfragerechteck \rightarrow Anfrage-Polygon im Dual-Raum

– Beispiel:



a: Native space.

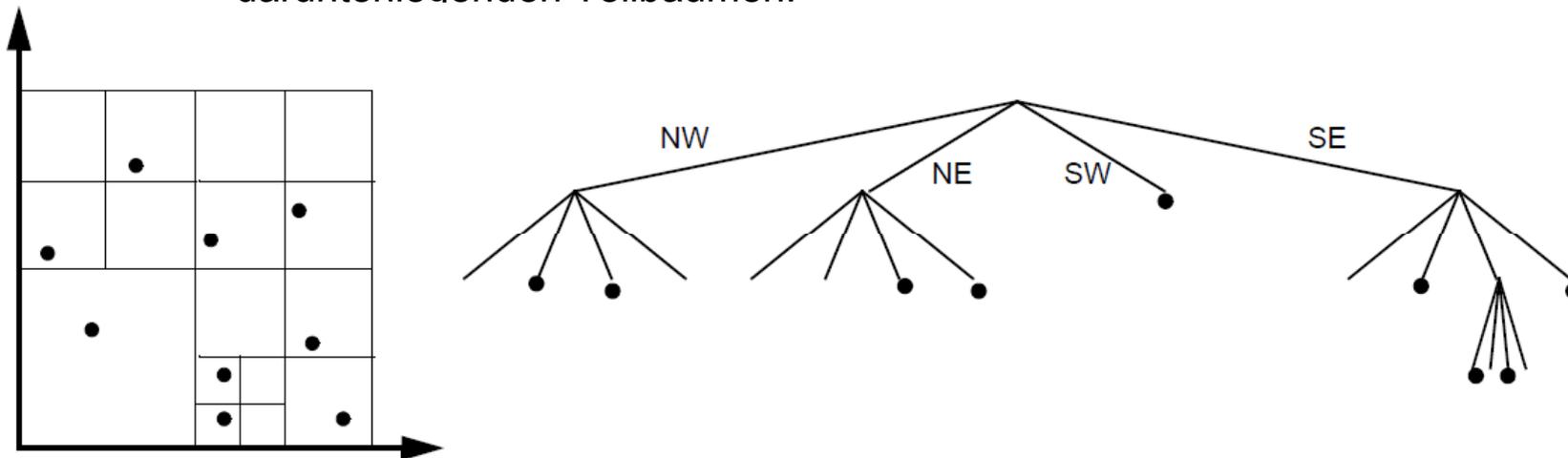


b: Dual space.

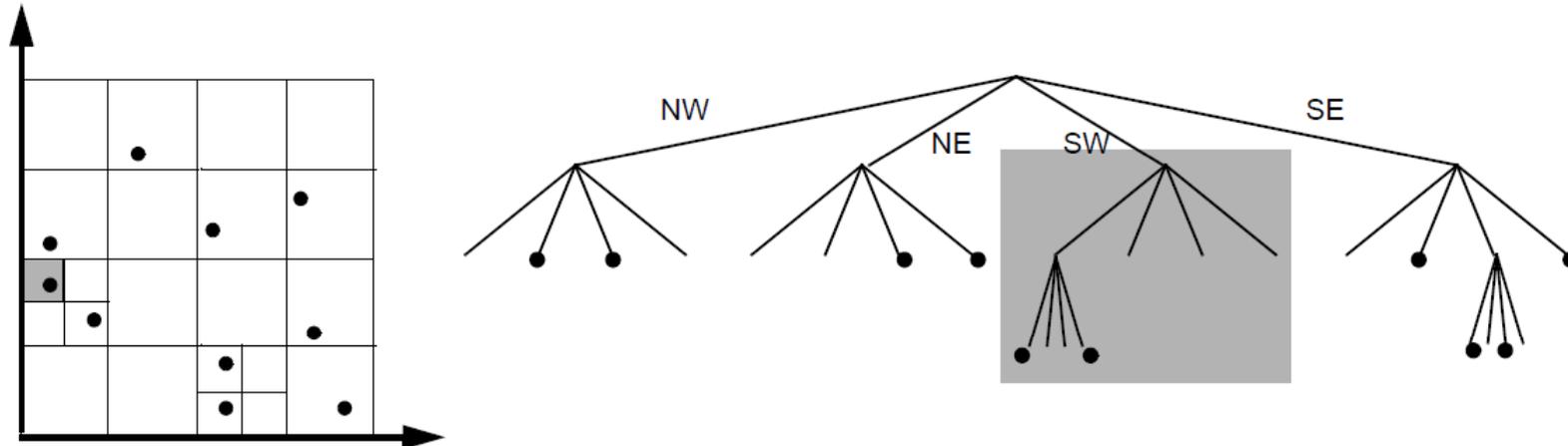
- Trajektorien im ursprünglichen Raum (native space) als Punkte im Dual-Raum (dual space).
- Fensteranfrage im ursprünglichen Raum entspricht Polygon im Dual-Raum.
 - Für jede Trajektorie $y=vt+a$ die das Anfrage-Fenster schneidet:
 - » existiert eine Trajektorie $y=vt+a''$ mit der gleichen Geschwindigkeit die durch Punkt P_1 geht mit $a < a''$, d.h. $a'' = y'' - vt'$ für $P_1=(t',y'')$.
 - » existiert eine Trajektorie $y=vt+a'$ mit der gleichen Geschwindigkeit die durch Punkt P_2 geht mit $a' < a$, d.h. $a' = y' - vt''$ für $P_2=(t'',y')$.

– Der STRIPES Index [PCC04]

- Basiert auf einer Sekundärspeicher-basierten Point-Region (PR) Bucket Quadtree Struktur.
- Exkurs: Point-Region Quadtree
 - variable Auflösung des Datenraums
 - Komprimierung eines internen Knotens falls im Teilbaum höchstens m Datensätze vorhanden sind (m = Kapazität einer Datenseite)
 - Dann werden die Datensätze direkt in dem internen Knoten abgespeichert und dessen vier Kinderknoten werden freigegeben.
 - Jeder interne Knoten besitzt mindestens $(m+1)$ Punkte in den darunterliegenden Teilbäumen.



– Einfügen



- Suche den Einfügeknoten N.
- Falls N weniger als m Punkte enthält, so füge den Datensatz in N ein.
- Andernfalls teile den Datenraum des Teilbaums von N solange rekursiv auf, bis die enthaltenen Punkte in mind. zwei unterschiedlichen Quadranten (Knoten) aufgeteilt sind.

- Der STRIPES Index (Fortsetzung)
 - Basiert auf einer Sekundärspeicher-basierten Point-Region (PR) Bucket Quadtree Struktur.
 - Für den $(d+1)$ dimensionalen (Original-) Raum (native space) betrachte man d Dual-Unterräume.
 - Jeder dieser d Dual-Unterräume wird jeweils in 4 gleichgroße Quadranten unterteilt.
 - Der Dual-Raum wird somit bei jeder Quadtreezellenzerlegung in $4^d = 2^{2d}$ Partitionen zerlegt.
 - Jeder interne Knoten des Quadtree hat somit 4^d Kinder.
 - Jeder Blattknoten enthält bis zu m Punktobjekte.
 - Ein Blattknoten speichert zudem:
 - *level*: Level auf dem sich der Blattknoten befindet (= definiert die Seitenregion).
 - *grid*: Angabe um welche Quadtree-Zelle es sich bei dem Knoten handelt.
 - *size*: Die Anzahl der enthaltenen Punktobjekte.

- Ein interner Knoten speichert:
 - *level*: Level auf dem sich der interne Knoten befindet (= definiert die Seitenregion).
 - *grid*: Angabe um welche Quadtree-Zelle es sich bei dem Knoten handelt.
 - *children_ptr*: Array der Größe 2^{2d} mit den Zeigern zu den Kindknoten.
 - *isleaf_array*: Array der Größe 2^{2d} mit Angabe ob die Zeiger auf die Kindseiten auf interne Knoten oder auf Blattknoten zeigen.
 - *size*: Die Anzahl der in den darunterliegenden Teilbaum enthaltenen Punktobjekte.

- Einfügen in STRIPES:
 - Suche rekursiv nach dem entsprechenden Blattknoten.
 - Falls das Blatt Platz hat füge den Punkt ein.
 - Ansonsten: Wandle den Blattknoten in ein internen Knoten und erzeuge 2^{2d} Kindknoten auf die die Einträge des Blattknotens entsprechend der Partitionierung aufzuteilen sind. (evtl. muß der Split rekursiv fortgesetzt werden).

- Entfernen in STRIPES:
 - Suche rekursiv nach dem entsprechenden Blattknoten.
 - Falls bei dem Durchlauf der internen Knoten die Anzahl der darunterliegenden Punkte (Attribut *size*) die Zahl $m+2$ unterschritten wird, werden alle darunterliegenden Einträge gesammelt, der aktuelle Knoten zum Blattknoten gewandelt und die gesammelten Einträge dort eingefügt.
 - Der Eintrag wird in dem final erreichten Blattknoten gelöscht.

- Anfragen in STRIPES:
 - Unterstützte Anfragen:
 - » Time-Slice Anfrage (Anfragen beziehen sich auf einen Zeitpunkt).
 - » ST Fenster Anfrage (siehe Übung) (Anfragen beziehen sich auf ein Zeitintervall).
 - » Dynamische ST Anfrage (ST Fenster Anfragen wobei das räumliche Anfragefenster über die Zeit hinweg variiert).

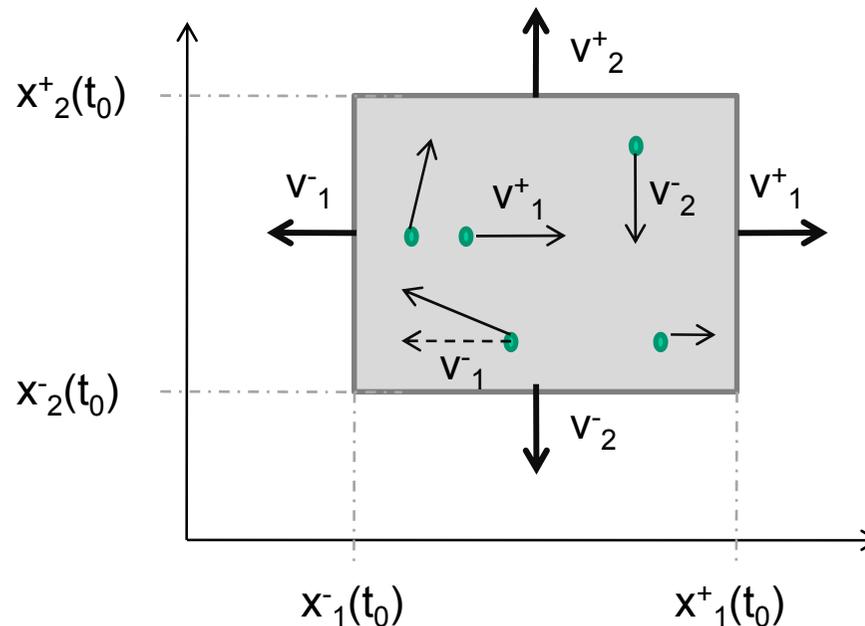
- Der Time-Parametrized R-Baum (TPR-Baum) [SJLL00]
 - R-Baum-Variante, d.h. Prinzip der überlappenden Seitenregionen.
 - Dimensionalität der Seitenregionen äquivalent zur Dimensionalität des räumlichen Attributs.
 - Prinzip der zeitparametrisierten Seitenregionen
 - Seitenregionen speichern neben der Ausdehnung zur Zeit t_0 einen Vektor über die (lineare) Veränderung der Ausdehnung über die Zeit.
 - Koordinaten der Seitenregionen sind Funktionen der Zeit.
 - Eine Seitenregion garantiert die konservative Approximation der enthaltenen Objekte, solange der Bewegungsvektor der enthaltene Objekte konstant bleibt.

- Seitenregion eines Blattknotens

- Achsenparalleles Rechteck, mit zeitparametrisierten Koordinaten

$[(x^-_1(t), x^+_1(t)), (x^-_2(t), x^+_2(t)), \dots, (x^-_d(t), x^+_d(t))]$ wobei

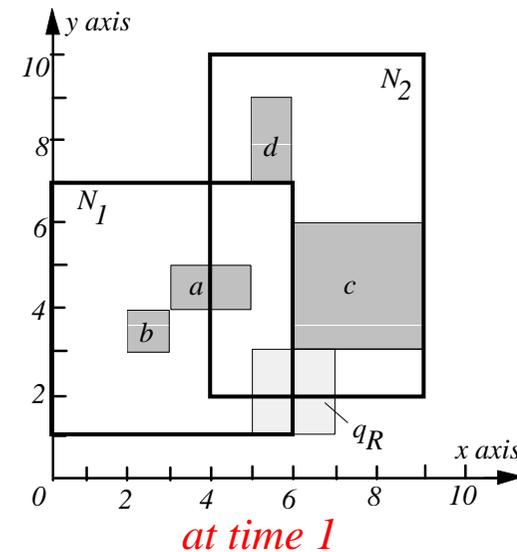
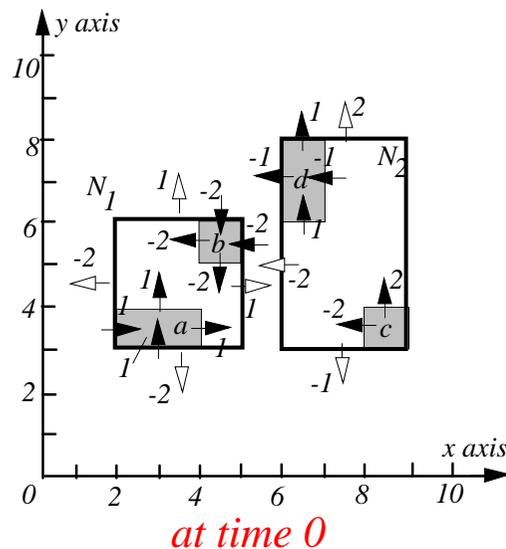
$(x^-_i(t), x^+_i(t)) = (x^-_i(t_0) + v^-_i(t-t_0), x^+_i(t_0) + v^+_i(t-t_0)), 1 \leq i \leq d.$



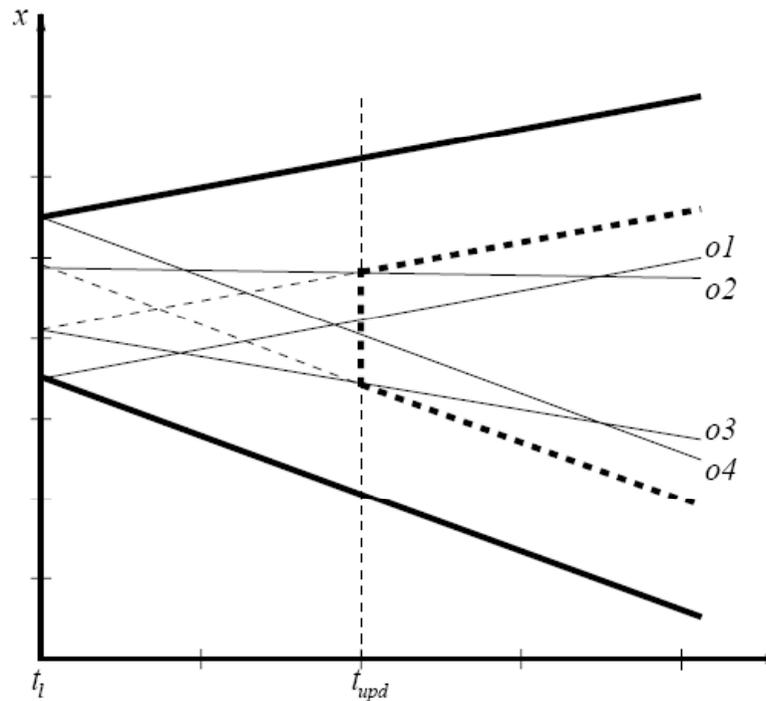
- v^-_i (v^+_i): minimale (maximale) Geschwindigkeit der enthaltenen Objekte bzgl. Dimension $1 \leq i \leq d.$

(Vorzeichen der Geschwindigkeit bestimmt die Bewegungsrichtung, z.B. negative Geschwindigkeit -5m/s bedeutet Objekt bewegt sich nach links mit der Geschwindigkeit 5m/s)

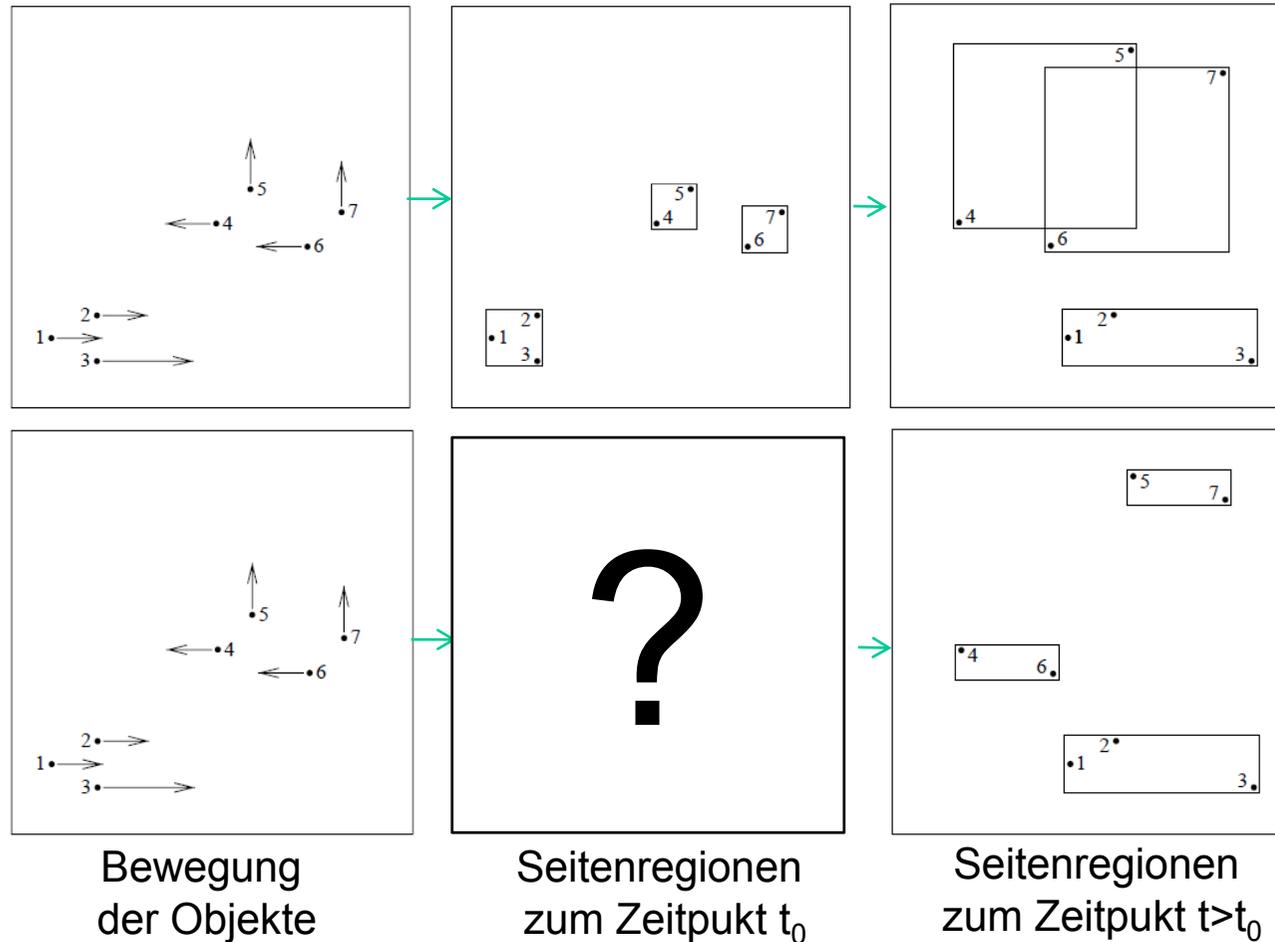
- Seitenregion eines Directory Knotens
 - Ebenfalls Achsenparalleles Rechteck, mit zeitparametrisierten Koordinaten
 - v_i (v_i^+): minimale (maximale) Ausdehnungs-Geschwindigkeit der enthaltenen Seitenregionen.
- Eigenschaften der Seitenregionen:
 - Zum **Zeitpunkt t_0** (Gegenwart) werden alle enthaltenen Objekte **minimal** umfasst.
 - Zum **Zeitpunkt $t > t_0$** werden die Objekte zwar **konservativ** aber **nicht** mehr unbedingt **minimal** umfasst
 - Beispiel:



- Aktualisierung der Seitenregionen
 - Seitenregionen wachsen kontinuierlich
=> **kontinuierliche Degenerierung der Approximationsgüte**
 - Seitenregion ist solange gültig bis zu einem Zeitpunkt t_{upd} (Zeithorizont) zu dem es sich lohnt diese zu erneuern
=> **Steigerung der Approximationsgüte**
 - Beispiel:



- Einfügestrategie
 - Welche Objekte sollen zusammen in einen Knoten eingefügt werden?
 - Räumliche Nähe nicht unbedingt optimal:



- Performanz des TPR-Baum stark abhängig vom Zeithorizont H :
 - » $H \approx 0$ -> Anfragen beziehen sich auf Zeitpunkte nahe bei Referenzzeitpunkt t_0 , damit kein negativer Einfluss veränderter Bounding Rectangles durch hohe Überschneidung.
 - » $H \gg 0$ -> Gruppiere Punkte so, dass die Größe der Seitenregionen zu allen Zeiten $[t_0, t_0+H]$ möglichst gering ist.
 - Gute Performanz, wenn v_{\min} - v_{\max} für jede Dimension gering ist (Seitenregion verändert dann ihre Größe nur geringfügig)
 - Minimiere Überschneidung, Umfang und Volumen wie bei der R^* -Baum Einfügestrategie (beachte aber dabei auch die Zeitliche Komponente in $[t_0, t_0+H]$)
-
- Unterstützte Anfragen:
 - Time-Slice Anfragen (Anfragen beziehen sich auf einen Zeitpunkt).
 - ST Fenster Anfragen (Anfragen beziehen sich auf ein Zeitintervall).
 - Dynamische Anfragen (ST Fenster Anfragen wobei das räumliche Anfragefenster über die Zeit hinweg variiert).

- Diskussion:
 - Nachdem der TPR-Baum Bewegungen vorhersagt sind die Ergebnisse einer Query immer approximativ.
 - Objekte im TPR-Baum können zur Laufzeit korrigiert werden
 - Eine Query kann also zu unterschiedlichen Zeiten unterschiedliche Ergebnisse liefern!
 - Queries weit in der Zukunft also wenig sinnvoll

- Gesamt-Überblick über S-, T- und ST-Indexstrukturen

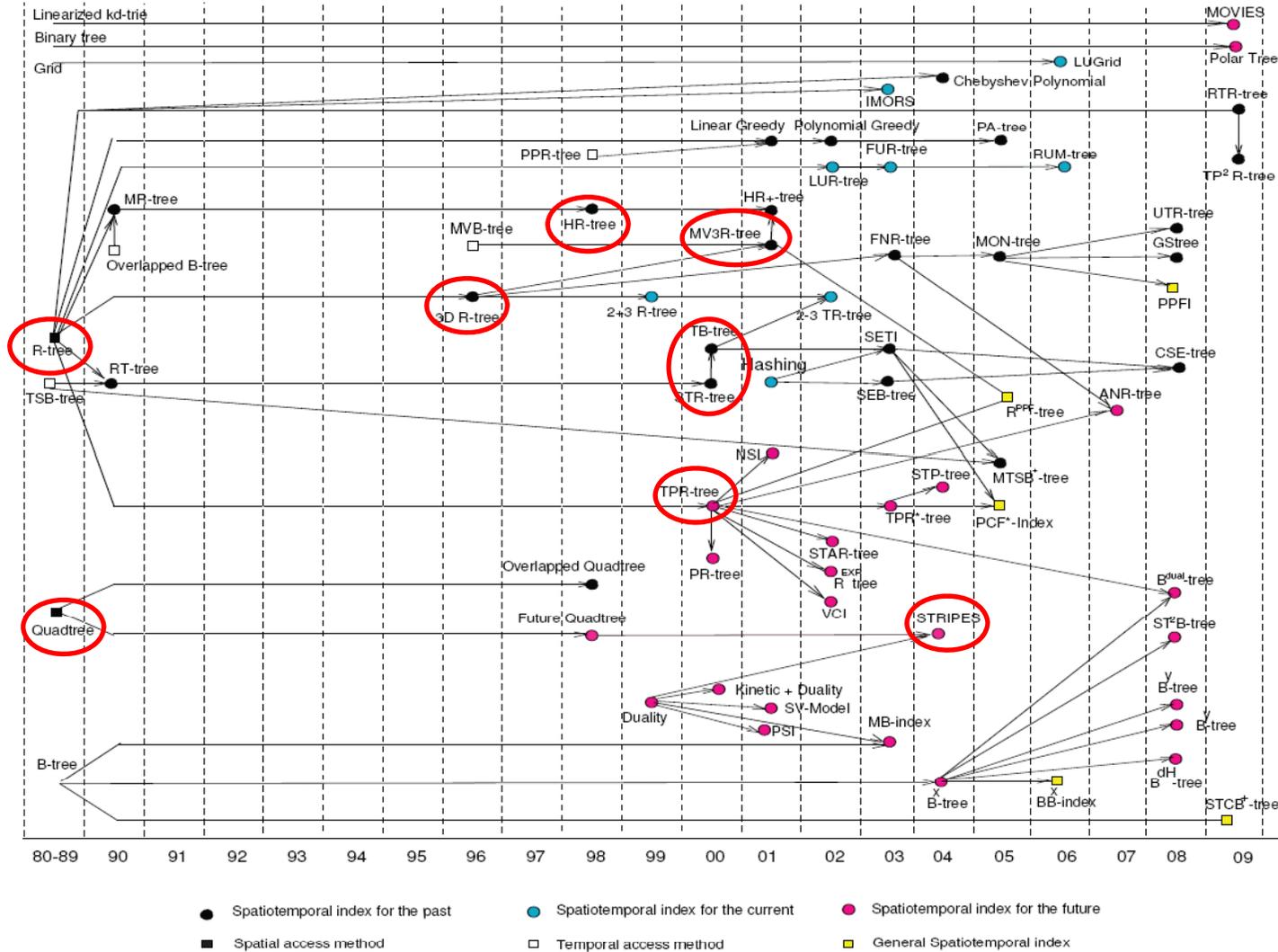


Figure 1: Survey of Spatiotemporal Access Methods.

[Quelle: NAM10]

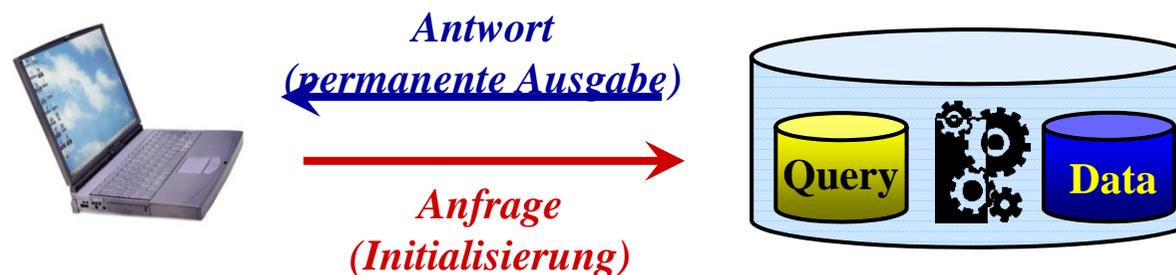
– Referenzen:

- [TVS96] Yannis Theodoridis, Michael Vazirgiannis, and Timos Sellis. Spatio-temporal Indexing for Large Multimedia Applications. In Proceeding of the IEEE Conference on Multimedia Computing and Systems, ICMCS, pages 441-448, Hiroshima, Japan, June 1996.
- [PJT00]: D. Pfoser, C. S. Jensen, Y. Theodoridis. Novel Approaches to the Indexing of Moving Object Trajectories. In Proc. of VLDB, 2000.
- [NAM10]: Long-Van Nguyen-Dinh, Walid G. Aref, Mohamed F. Mokbel: Spatio-Temporal Access Methods: Part 2 (2003 - 2010). IEEE Data Eng. Bull. 33(2): 46-55 (2010).
- [PCC04]: J.Patel,Y.Chen,andV.Chakka. STRIPES: An efficient index for predicted trajectories. SIGMOD 2004.
- [EEK03]: K.M.Elbasioni, A.Elmasry, and I.Kamel. An efficient indexing scheme for multi-dimensional moving objects. In ICDT, pages 422–436, 2003.
- [SJLL00]: S.Saltenis, C.Jensen, S.Leutenegger, and M.Lopez. Indexing the positions of continuously moving objects. In SIGMOD, pages 331–342, 2000.

3.3 Methoden für kontinuierliche Anfragen (Continuous (Monitoring) Queries)

3.3.1 Allgemeines:

- Eine kontinuierliche Anfrage gibt zu einem gegebenen räumlichen Anfrageprädikat P die entsprechenden Ergebnisse für alle Zeitpunkte innerhalb eines (oder mehrere) gültige(n) Zeitbereiche(s)



- Typen von Objekten:
 - 1) Trajektorien aller beweglichen Objekte sind bekannt
 - 2) Objekte bewegen sich frei im Raum

3.3.2 Ansätze zu Methoden für kontinuierliche Anfragen

– Naiver Ansatz:

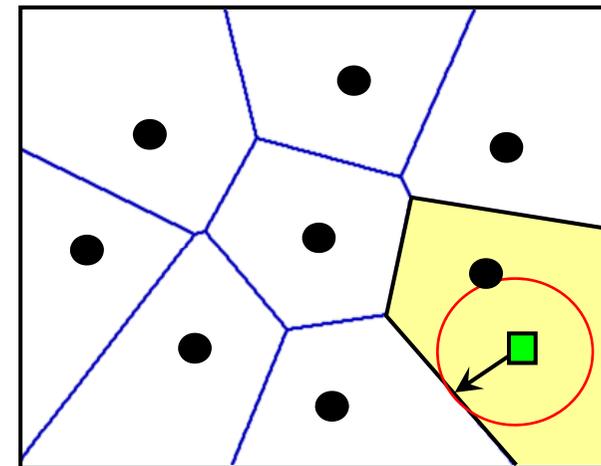
- Periodische Auswertung von Snapshot-Anfragen in der Gegenwart (siehe Kap. 3.2.3).
- Starke Redundanz von Auswertungsoperationen.
- Es werden jedesmal ähnliche Antworten ausgegeben (Großer Overhead in der Antwortmenge).
- Starke Verzögerung bei der Beantwortung von Anfragen (Echtzeitfähigkeit gefährdet!!!).

– Weitere Ansätze:

- *Result Validation:*
- *Result Caching:*
- *Result Prediction:*
- *Incremental Evaluation:*

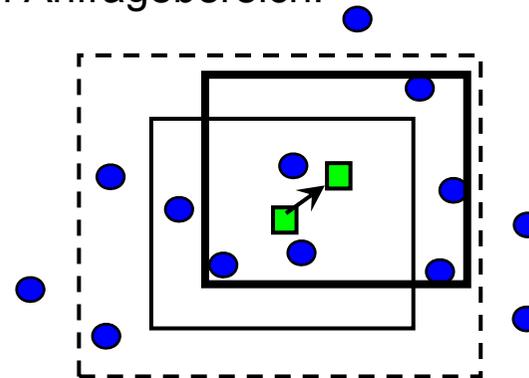
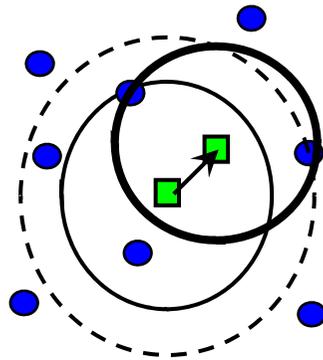
– Prinzip des *Result Validation*:

- Idee:
 - Verknüpfung von Anfrageergebnisse an Gültigkeitsbedingungen.
- Gültigkeitsattribute:
 - **Gültigkeitszeit** t_{valid} : Bestimmt das Zeitintervall für das die aktuelle Antwort ihre Gültigkeit behält.
 - **Gültigkeitsregion** r_{valid} (safe region): Solange ein Objekt sich in seiner Gültigkeitsregion aufhält behält die aktuelle Antwort ihre Gültigkeit.
- Herausforderung: Bestimmung der Gültigkeits
- Beispiel: Prinzip der Safe Regions bei der NN-Anfrage
 - DB-Objekte stationär.
 - Anfrageobjekt bewegt sich .
 - Über Voronoi-Diagramm ist die NN-Nachbarschaft aller DB-Objekte vorberechnet.
 - Solange sich das Anfrage-Objekt innerhalb einer Voronoi-Zelle (safe region) aufhält muß keine Aktualisierung des aktuellen Ergebnis erfolgen.



– Prinzip des *Result Caching*:

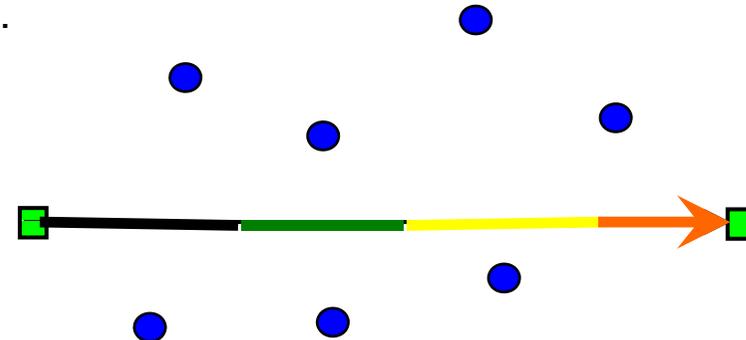
- Idee:
 - Antwortmengen von direkt aufeinanderfolgenden Anfragen sind sehr ähnlich zueinander.
 - Bei der Bearbeitung einer Anfrage werden mehr Objekte gesucht als für die aktuelle Antwort nötig sind (*caching* möglicher Ergebnisse zukünftiger Anfrageiterationen).
- Beispiele:
 - kNN Anfragen: Suche mehr als die k nächsten Nachbarn.
 - Bereichsanfragen: Vergrößere den Anfragebereich.



- Herausforderung:
 - Bestimmung wieviel Vorberechnet werden muß (Cache Overhead)

– Prinzip des *Result Prediction*:

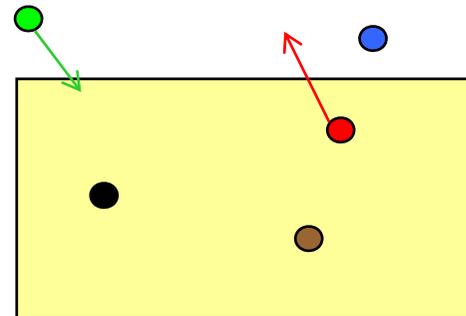
- Idee:
 - Falls die zukünftige Bewegung der Objekte bekannt ist, kann das Anfrageergebnis vorberechnet werden.
 - Berechnung kann z.B. durch den TPR-Baum unterstützt werden.
- Beispiel:
 - Kontinuierliche NN Anfrage.
 - Die Trajektorie ist in Segmente aufgeteilt, sodaß für jedes Segment das NN Anfrageergebnis eindeutig ist.



- Die Anfrage wird als Snapshot-Anfrage ausgeführt.
- Sie behält Ihre Gültigkeit für einen längeren Zeitraum.
- Sobald sich die Trajektorie ändert wird das Anfrageergebnis aktualisiert.

– Prinzip des *Incremental Evaluation*:

- Idee:
 - Änderungen sind oft nur sehr lokal.
 - Anfrage wird einmalig ausgeführt.
 - Dann wird lediglich die Antwortmenge aktualisiert falls notwendig.
- Unterscheidung zwischen *positiven* und *negativen* Aktualisierungen.
- Positive Aktualisierung: Hinzufügen von Objekten
- Negative Aktualisierung: Entfernen von Objekten
- Nur Objekte die die Grenze der Anfrageregion überschreiten sind zu berücksichtigen.



- Herausforderung:
 - Erkennung wann ein Objekt die Grenze der Anfrageregion überschreitet.