Modeling and Querying Vague Spatial Objects Using Shapelets

Daniel Zinn[1] Jim Bosch[2] Michael Gertz[1]

[1]Department of Computer Science
[2]Department of Physics

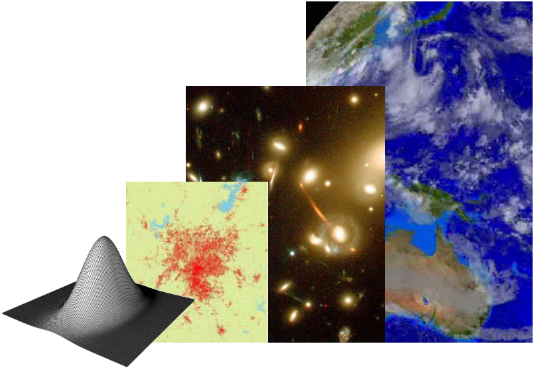
University of California at Davis

Vague Spatial Objects

. . . are localized objects with uncertainties

Examples

- Astronomical objects
- Meteorological phenomena
- Demographic regions
- Eco-regions
- Probability for “X”



Objective: Data and query model for vague spatial objects

Outline

Outline

1. Related Work and Contributions
2. High-Level and Low-Level Operations
3. Shapelets
4. Prototype and Evaluation
5. Summary and Ongoing Work

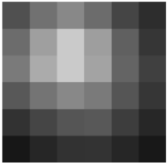
Related Work

Fuzzy Regions [Schneider et al. '97-'05]

- Contour representation
- Operations scale with number of vertices
- High number of vertices for smooth objects
- Discrete Values

Pixel Representations

- Field data
- Operations scale with number of pixels
- Many pixels necessary for smooth objects
- Discrete and fixed squares in x/y area



Both representations are built on a discrete "basis"

Contributions

Data and Query Model

- Represent vague spatial objects with *shapelets*
- Characterized a comprehensive set of low-level operations
- Build high-level operations from low-level operations

Realization and Evaluation

- Implementation in PostgreSQL
- Shapelet as column data type
- 29 low-level operations implemented
- Stored procedures for high-level operations
- Sample queries and performance experiments
- [Indexing technique based on ε-bounding boxes]

Outline

High-level Operations on Vague Spatial Objects

Standard topological Operations

- Window and point operations
- Union, intersection, overlap

Metric Operations

- Area
- Width, height
- Centroid

Geometric Transforms

- Scale
- Translate
- Rotate

Low-Level Operations

- Point-wise evaluation, and arithmetic ops.
- min/max ops.
- Integrals and integral moments
- Value-based and integral-based contours
- Scale, translate, rotate

High-level Operations on Vague Spatial Objects

Standard topological Operations

- Window and point operations
- Union, intersection, overlap

Metric Operations

- Area
- Width, height
- Centroid

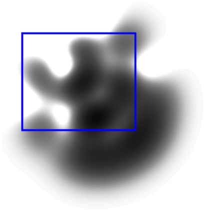
Geometric Transforms

- Scale
- Translate
- Rotate

Low-Level Operations

- Point-wise evaluation, and arithmetic ops.
- min/max ops.
- Integrals and integral moments
- Value-based and integral-based contours
- Scale, translate, rotate

Examples: Overlap and Width



Overlap

Does/How much does rectangle R overlap with vague object F?

- $overlap(F, R) := \iint_R f(x, y)$, or
- $overlap(F, R) := \max_R\{f(x, y)\}$

Width

How long is F along the x-dimension?

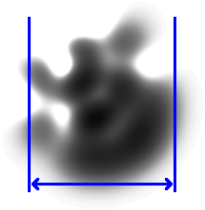
- Value-based contour to measure width of crisp contour $width(F) := width(contour(F, t))$, or
- Root-mean-square width, i.e. standard derivation $width(F) := \left[\frac{\iint x^2 f(x,y)dx\,dy}{\iint f(x,y)dx\,dy}\right]^{1/2}$

Examples: Overlap and Width

Overlap

Does/How much does rectangle R overlap with vague object F?

- $overlap(F, R) := \iint_R f(x, y)$, or
- $overlap(F, R) := \max_R\{f(x, y)\}$



Width

How long is F along the x-dimension?

- Value-based contour to measure width of crisp contour $width(F) := width(contour(F, t))$, or
- Root-mean-square width, i.e. standard derivation $width(F) := \left[\frac{\iint x^2 f(x,y) dx\, dy}{\iint f(x,y) dx\, dy}\right]^{1/2}$

Examples: Overlap and Width

Overlap

Does/How much does rectangle R overlap with vague object F?

- $overlap(F, R) := \iint_R f(x, y)$, or
- $overlap(F, R) := \max_R\{f(x, y)\}$

Width

How long is F along the x-dimension?

- Value-based contour to measure width of crisp contour $width(F) := width(contour(F, t))$, or
- Root-mean-square width, i.e. standard derivation

$$width(F) := \left[\frac{\iint x^2 f(x,y)dx\,dy}{\iint f(x,y)dx\,dy}\right]^{1/2}$$

Outline

Shapelets – *Intro*

Shapelets

- Image compression technique, which has been developed in astronomy
- Distorted 2-dimensional Gaussian functions

Our Contributions

- Use shapelets for representing general vague objects
- Refined math for low-level operations (eg. for overlap)
- Developed math for ε-bounding boxes

Shapelets – *Series Expansion*

Series expansion

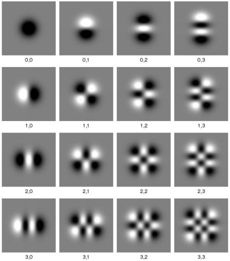
$$f(x,y) = \sum_{n=0}^{\infty} a_n \phi_n(x,y)$$

- Basis functions weighted by coefficients
- Representation of arbitrary vague objects

$f(x,y)$

$$= \begin{array}{c} a_n \\ \boxed{\begin{matrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{matrix}} \end{array} *$$

$\phi_n(x,y)$



Shapelets – *Localized, Smooth Basis Functions*

1D Shapelet Basis Functions

$$\phi_n(x) = \left[2^n \pi^{1/2} n!\right]^{-1/2} H_n(x) e^{-\frac{x^2}{2}}$$

- Hermite Polynomials, weighted by a Gaussian
- $H_{n+1}(x) = xH_n(x) - H'_n(x)$

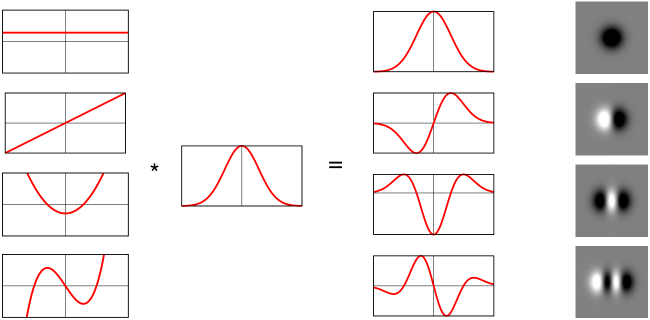
Hermite Polynomials | Gaussian | 1D Shapelet | 2D Shapelet

$H_0 = 1$

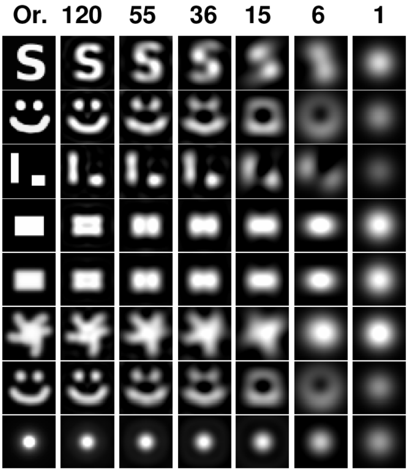
$H_1 = x$

$H_2 = x^2 - 1$

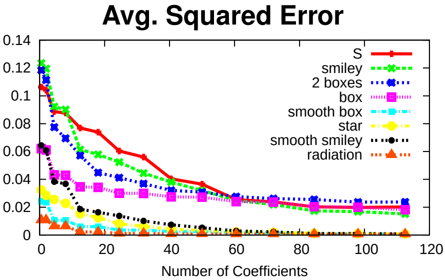
$H_3 = x^3 - 3x$



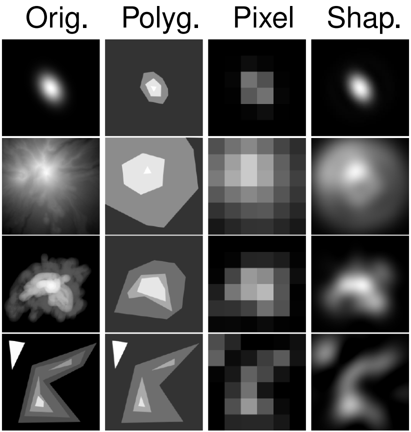
Representing Arbitrary Objects



- Arbitrary smooth objects
- Quality improves with number of coefficients
- Excellent for smooth objects, OK for crisp objects



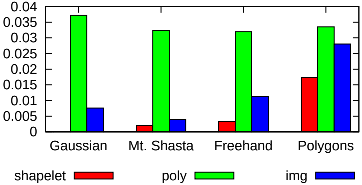
Shapelets vs. Pixels vs. Polygons



Each representation is limited to the same amount of memory (36 floating point values).

- Outstanding for Gaussian-like objects
- Outperform pixel representation
- Outperform polygon representation – even for the polygons

Avg. Squared Error



Revisiting Low-level Operations

Example: Integral-Operator

Recursion Relations for Hermite Polynomials

- $H_n(x) = 2xH_{n-1}(x) - 2(n-1)H_{n-2}(x)$
- $\frac{dH_n(x)}{dx} = 2nH_{n-1}(x)$

Recursion Relations for Integration over Shapelets

- $I_n = \int_a^b \phi_n(x)$
- $I_n = -\sqrt{2/n}[\phi_{n-1}(x)]_a^b + I_{n-2}\sqrt{1-1/n}$
- $I_0 = \sqrt{\pi^2/2}[erf(x/\sqrt{2})]_a^b$
- $I_1 = -\sqrt{2}[\phi_0(x)]_a^b$

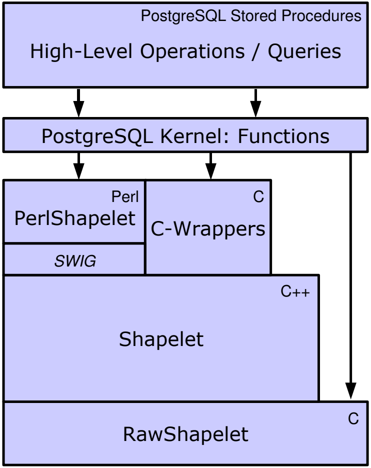
Operation scales linearly with number of coefficients!

Outline

PostgreSQL Implementation

Architecture

- New column datatype: *Shapelet*
- Stored Procedures for high-level operations
- C++ class Shapelet, which implements 29 low-level operations
- GNU Scientific Library (GSL) for matrix operations
- C struct RawShapelet as data container



Example: Neighbor Overlap

C-Binding for Low-level Operations

```
CREATE FUNCTION s_overlap(shapelet, shapelet) RETURNS FLOAT8 AS
 '_OBJWD_/shapelet', 'RawShapelet_overlap'
LANGUAGE C IMMUTABLE STRICT;
```

High-level to Low-level Mapping: Overlap

```
CREATE FUNCTION overlap_Symmetric(shapelet,shapelet)
RETURNS double precision AS
 'SELECT s_overlap($1,$2)/
         ( s_integrateAll($1) * s_integrateAll($2) )
  AS result;'
LANGUAGE SQL;

CREATE FUNCTION overlap_Asymmetric(shapelet,shapelet)
RETURNS double precision AS
  'SELECT s_overlap($1,$2)/s_integrateAll($1)^2 AS result;'
LANGUAGE SQL;
```

Implemented Low-level Operations

	Signature
Input/Output	importString(t):s
	exportString(s):t
	importPNG(t,p,f,i):s
	exportPNG(s,i,i,b,t)
	makeGaussian(p,f,f):s
Get/Set	getCenter(s):p
	setCenter(s,p)
	getBeta(s):f
	setBeta(s,f)
Info	evalAtPoint(s,p):f
	integrateBox(s,b):f
	integrateAll(s):f
Index	getIntBBox(s,f):b
	getMaxBBox(s,f):b
	getEpsBBox(s,f):b

	Signature
Arithmetic	multiplyScalar(s,f):s
	multiply(s,s):s
	add(s,s):s
	subtract(s,s):s
	normalize(s):s
Topo-logical	intersection(s,s):s
	union(s,s):s
	overlap(s,s):f
Geometric Transforms	scale(s,f,f,bl):s
	uniformScale(s,f,bl):s
	rescale(s,f,i):s
	recenter(s,p):s
	translate(s,p):s
	rotate(s,f):s

s:Shapelet. t:Text. f:Float. p:Point. b:Box. i:Int. bl:Bool

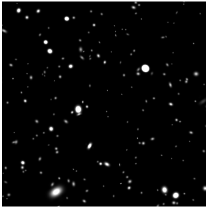
Sample Queries

Inserting galaxies from an astronomical catalog

```
CREATE TABLE galaxies AS (
  SELECT rotate(scale(makeGaussian(c,1.0,f),a,b,TRUE),theta)
  FROM catalog );
```

Galaxies, having >50% of their brightness in a certain box

```
SELECT * FROM galaxies
WHERE integrateBox(g, BOX '((5000,5000),(5500,5500))')
  > 0.5*integrateAll(g);
```



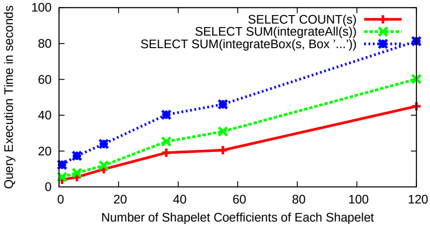
PNG export

```
SELECT shapelet_exportPNG(allgal, 100,100,
                          5000,5000, 5500,5500,
                          '/tmp/out.png')
FROM (
       SELECT array_accum(g * 250000 * 200 ) AS allgal
       FROM galaxies
       WHERE bbox_0_005 && '(5000,5000,5500,5500)'
     ) AS foo;
```

Performance Experiments

Experimental Setup

- Varying number of coefficients (1 . . . 120)
- 1 million shapelets of each resolution
- Measured query runtime (with PostgreSQL query statistics)



Result

- Operations scale linearly with number of coefficients
- Operations *integrateAll* and *integrateBox* comparable to *count*

Outline

Summary

High-level vs. Low-level Operations

- A specific set of low-level operations is sufficient to provide a basis for implementing important high-level operations useful in several application areas

Series Expansion on Shapelet Basis

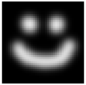
- Arbitrary objects can be represented
- Localized, *smooth* set of basis functions
- "Nice" mathematical properties

PostgreSQL Implementation

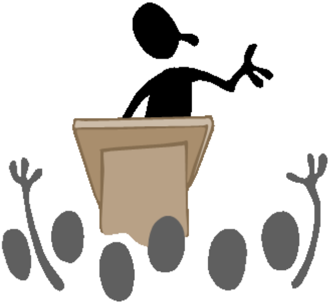
- Ready-to-use implementation for PostgreSQL
- Indexing with ε-bounding boxes

Future Work

- Multi-shapelets
- Contouring shapelets
- Providing a full set of high-level operations for a specific application domain
- GIST for indexing shapelets



Questions?



Acknowledgments

This work is in part supported by the National Science Foundation under Awards IIS-0326517 and ATM-0619139

More Source Code

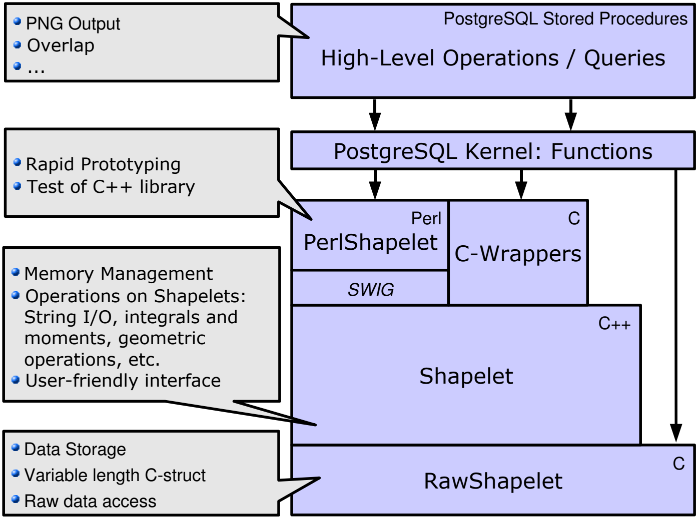
C implementation for RawShapelet_integrateAll

```
PG_FUNCTION_INFO_V1(RawShapelet_integrateAll);
Datum RawShapelet_integrateAll(PG_FUNCTION_ARGS) {
  RawShapelet *s = (RawShapelet *) PG_GETARG_POINTER(0);
  double result;
  RawShapeletIntegrateAll(s, &result);
  PG_RETURN_FLOAT8(result);
}
```

RawShapelet

```
typedef struct RawShapelet {
  int size; double beta, x, y;
  double data; // starting element for data array
} RawShapelet;
// Low-level data access methods
inline void setData(RawShapelet *s, int offs, double v)
{ (&(s->data))[offs] = v; }
```

Architecture



Intersection/Union with Multiplication and Addition