

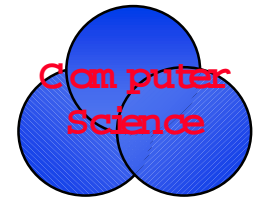
---

# Putting the I into VS PL

**Charles C. (Chip) Weems**  
**Computer Science Dept.**  
**University of Massachusetts**  
**Amherst, MA 01003**  
**[weems@cs.umass.edu](mailto:weems@cs.umass.edu)**



# Why I'm Here



---

## Email from Randall Janka:

"We really need you and Bruce there to help us characterize the IP content of the VSIPL interface"

## Email from Bruce Draper:

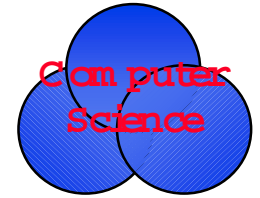
"I hate to bail on you, but NIMA has asked me to come to Washington July 24-26"

**OK, So I'd characterize it as minimal but adequate.**

**Can I go home now?**



# Obviously not...



---

Email from Randall Janka:

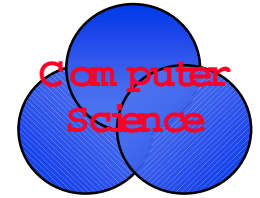
`"give a one-hour presentation on what  
y'all think would be a viable way of  
adding good IP support into VSIPL, given  
its current incarnation"`

**"A professor is someone who can take any subject and  
reduce it to a five minute talk or expand it to a one-hour  
lecture."**

**What's your preference?**



# Past Experience

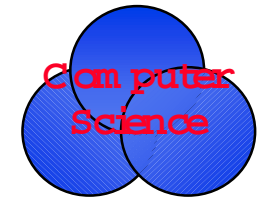


- 
- **DARPA IU Benchmarks**
  - **Image Understanding Architecture software**
  - **Beginnings of DARPA IUE**
  - **Consulting w/ KBVision developers**
  - **Chair of CAMP '97 workshop**
  - **15 years of research in architectures for IU**

**Somebody obviously thinks this all matters...**



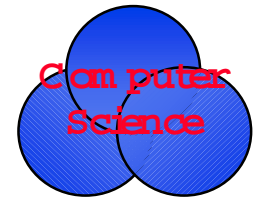
# What I Think They Expect



<b>Transformations</b>	<b>Edges</b>	<b>Regions</b>
Threshold	Kirsh Operator	Connected Components
Gaussian Convolution	Sobel Operator	Local Histogram Segment.
Laplacian Convolution	Roberts Cross	Feature Space Clusters
Difference of Gaussians	Canny Operator	Multiresolution Segment.
Median Filter	Nevatia-Babu Operator	Region Growing
Bandpass Filter	Hough Transform	Region Splitting
Histogram	Template Matching	Blackboard Based Segment.
Statistical Measures	Edge Linking	Expand and Contract
Covariance	Edge Thinning	Medial Axis
Correlation	Contour Tracing	Bounding Rectangle
Fourier Transform	Chain Coding	Convex Hull
Gray-level Correction	Spline Fitting	Area
Hadamard Transform	First Difference	Eccentricity
Hilbert Transform	Fourier Curve Fitting	Euler Number
Walsh Transform	Conic Curve Fitting	Compactness
Kalman Filter	End-point Fitting	Quad-tree Encoding
Predictive Compression	Average Contrast	Region Adjacency List
Geocorrection	Length	Concavity Tree
	Straight Line Fitting	Boundary Tracing



# ...and more

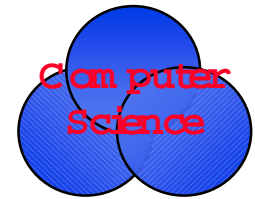


---

<b>Surface</b>	<b>Stereo</b>
Surface Detection	Correspondence Matching
Shape from Shading	Stereo Displacement Field
Shape from Texture Gradient	Depth from Stereo
Shape from Depth Map	<b>Texture</b>
Plane Fitting	First Order Texture Measures
Generalized Cone Fitting	Second Order Texture Measures
Fourier Shape Description	Energy Measures (Entropy, etc.)
<b>Motion</b>	Fourier Texture Measures
Moravec Interest Operator	Markov Random Field Measures
Flow Field Normalization	Structural Texture Descriptions
Flow Field Decomposition	Texture Segmentation
Focus of Expansion Determination	<b>Miscellaneous</b>
Depth from Motion	Orthogonal & Oriented Windowing
Motion Detection	Corner Detection
Moving Point Trajectory	Std. Deviation of Euclidean Distance



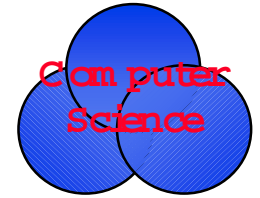
# Don't Even Think About It



- 
- I'm here to tell you what not to do.
  - **First classic mistake in a project like this**
    - Asking multiple researchers to give you advice
      - » Guaranteed to get multiple conflicting opinions
  - **Second classic mistake**
    - Expecting practical feedback from a researcher
      - » What do we know about real-world applications?
      - » We talk to people like DARPA, after all!
  - **Third classic mistake**
    - Trying to “get it right” before it's released
      - » Get over it. It will never be right. Release it and let it evolve.



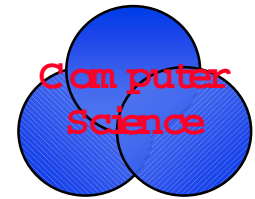
# DARPA IUE Experience



- **Years of “tiger team” meetings**
  - Defining standards and specifications
  - Arguing philosophical differences
- **Millions of dollars and years of development**
  - Delivered on time, within budget, precisely as specified
  - Nobody cared. Remained largely unused.
  - Result was a huge and unwieldy system
- **Classic “design by committee” example**
  - But it was “right”
  - As defined by a narrow and transient constituency



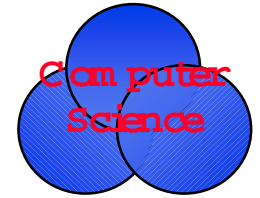
# DARPA IUB Experience



- **Defined by two research groups**
  - Only three meetings and some e-mail
  - One group did design, another reviewed and commented
  - Took just a few months
- **Implemented by one research group**
  - About \$150K add-on to existing contract
  - Finished in 9 months
- **Used by about 100 groups over 10 years**
  - Never forget: This is a **small** community!
  - Didn't try to "get it right"
  - Got it done and let it evolve over time



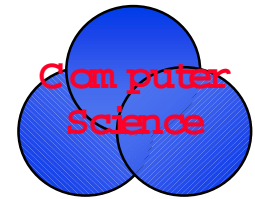
# IUA Software Experience



- **Low-level libraries are a waste of effort**
  - Hand-optimized individual routines don't port
  - Fail to take advantage of interloop cache locality
  - Compilers don't do enough interprocedural optimization
  - Languages like C make interprocedural analysis of libraries problematic except in the simplest cases
- **Developed compiler to recognize library class**
  - Good interprocedural analysis and code performance
  - Class was portable to other platforms (standard C++)
  - Major compiler effort, only delivering performance on one platform
- **Too costly for widespread use (but “right”)**



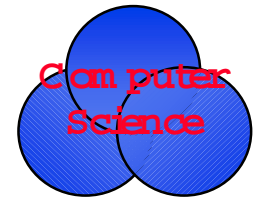
# KBVision Experience



- 
- **Commercial development environment**
  - **Good user interface, complete library**
  - **Easy to use after initial learning curve**
  - **Not a viable product**
    - Never forget: It's a **small** community!
    - Good for designing solutions, but no embedded performance
  - **Illustrates costly design/implementation split**
    - Raw embedded libraries are hard to work with
    - Cushy design environments lead to hard-to-embed solutions



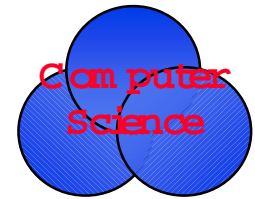
# CAMP Experience



- **Community isn't crying out for a standard**
  - Happy to use vendor tools and libraries
  - Many solutions are problem-specific and one-time use
  - Every configuration is different
  - Willingness to build from scratch
- **No expectation of portability**
  - Buy whatever HW gets the job done at best price
  - Stay with one vendor to avoid learning new tools/uniqueness
- **No expectation of reuse**
  - If a new problem needs special hardware, it's all different
  - Obsolete hardware is replaced by incompatible hardware



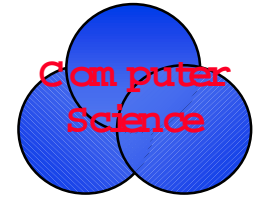
# Research Experience



- **Haven't used earlier commercial hardware for IP**
  - Too slow (built custom instead)
  - Too expensive (used general purpose HW instead)
  - Software inadequate (often easier to build from scratch for a simpler environment)
- **Getting better**
  - Almost fast enough
  - Getting harder to build custom that's faster
  - Funding agencies still balk at cost
  - Wouldn't base choice on availability of an IP library
  - Wouldn't build applications to be library-pure
  - Rest of SW environment, especially good compiler and debugging tools, is more important



# VS IPL Advice



- 
- **Don't put a lot of effort into IP library**
    - Maybe add some conversions to/from standard file formats
  - **Keep it simple enough to implement easily**
  - **Get prototype implementation distributed**
  - **Set up an “open source” like group to oversee evolution**
    - Likely to be little work
  - **Don't be discouraged by low level of use**
    - Never forget: It's a **small** community.
  - **Send me home (please!)**