Robot Vision Systems

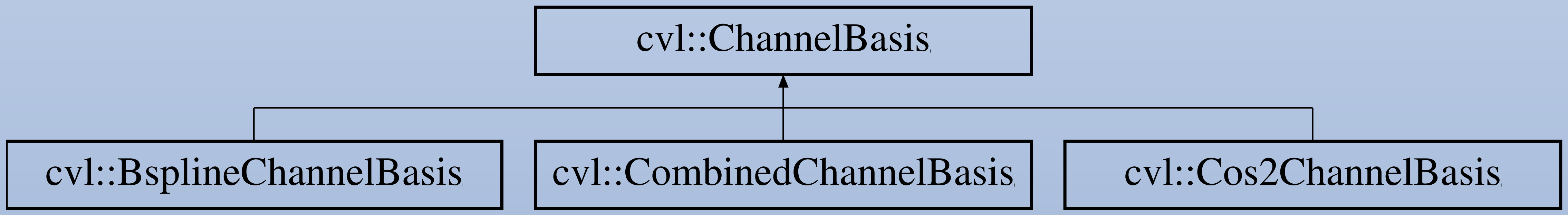
Lecture 5: Building your own Representation in OpenCV

Michael Felsberg

michael.felsberg@liu.se

Channel Toolbox

- Tutorial
- Based on Mat/SparseMat
- Makes use of separate channel basis class



- cvl::ChannelVector inherits from Mat_<float>
- cvl::ChannelSVector inherits from SparseMat_<float>

ChannelVector

- ChannelVector () *Standard constructor.*
- ChannelVector (ChannelBasis ∗chBasis) *Constructor with basis initialization.*
- ChannelVector (ChannelBasis ∗chBasis, const cv::Mat_< float > coeffs) *Constructor with basis initialization and copying coefficient matrix.*
- void addSample (const cv::Mat &vals) *Add sample(s) functionality.*
- void decode (cv::Mat &res, const int nrModes=1) const *Decoding functionality.*

ChannelVector

- void normalize () *Normalize the vector.*
- void setChannelBasis (ChannelBasis ∗chBasis) *Set or change the channel basis to be used.*
- void channelImage (cv::Mat &res) const *Generate different layout, suitable for channel smoothing.*
- void histogramMatrix (cv::Mat &res) const *Generate different layout, compatible with histograms (N-D matrices); last dimension corresponds to the previous rows.*

Code examples

```
cvl::ChannelVector::ChannelVector(
    ChannelBasis *chBasis):
    cv::Mat_<float>(1,chBasis->getNrChannels(),0.0f){
        m_chBasis = chBasis;
        m_support.resize(2);
        m_support[0] = 1;
        m_support[1] = 1;
}
```

This?

```
void cvl::ChannelVector::setChannelBasis(
    ChannelBasis *chBasis){
        m_chBasis=chBasis;
        ~~this->~~create(1,
            m_chBasis->getNrChannels());
        ~~this->~~setTo(0.0f);
        return;
}
```

Design Questions

```
void cvl::ChannelVector::normalize(){
    float nrm=norm(*this,cv::NORM_L1);
    if (nrm>0)
        (*this)*=(m_chBasis->getNorm()/nrm);
    else
        setTo(m_chBasis->getNorm()/cols);
    return;
}
```

Design Questions

```
void cvl::ChannelVector::channelImage(
    cv::Mat &res) const {
    res = reshape(cols,m_support[0]);
}
```

(the usual design is 1-channel, cols = number of channels, rows = number of samples)

Design Questions

```
void cvl::ChannelVector::histogramMatrix(cv::Mat &res) const {
    std::vector<int> sizes(1);
    m_chBasis->getNrChannelsVec(sizes);
    sizes.push_back(rows);
    cv::Mat resTmp((int)sizes.size(),&sizes[0],CV_32F,data);
    res = resTmp;
}
```

ChannelSVector

- ChannelSVector () *Standard constructor.*
- ChannelSVector (ChannelBasis ∗chBasis) *Constructor with basis initialization.*
- void addSample (const cv::Mat &vals) *Add sample(s) functionality.*
- void decode (cv::Mat_< float > &res, const int nrModes=1) const *Decoding functionality.*
- void normalize () *Normalize the vector.*
- void setChannelBasis (ChannelBasis ∗chBasis) *Set or change the channel basis to be used.*

Code Samples

```
cvl::ChannelSVector::ChannelSVector(
    ChannelBasis *chBasis):cv::SparseMat_<float>()
{
    m_chBasis=chBasis;
    int sizes[] = {chBasis->getNrChannels()};
    create(1,sizes);
    m_support.resize(2);
    m_support[0] = 1;
    m_support[1] = 1;
}
```

Code Samples

```
std::ostream &operator << (std::ostream &os, const
                                        cvl::ChannelSVector &v) {
    int nrR = (int)v.size(0);
    os << '[' << nrR << "](";
    for (cv::SparseMatConstIterator_<float> i = v.begin(); i !=
                                        v.end(); ++i) {
        const cv::SparseMat::Node* n = i.node();
        if (i == v.begin())
            os << n->idx[0] << ":" << *i;
        else
            os << ", " << n->idx[0] << ":" << *i;
    }
    os << ')';
    return os;
}
```

Code Samples

```
void cvl::…::decode(cv::Mat_<float> &res, const int nrModes) const {
    std::vector<int> sizes(1);
    m_chBasis->getNrChannelsVec(sizes);
    int nrChannels = nrModes*(sizes.size()+1);
    CV_Assert(nrChannels<=CV_CN_MAX);
    int nrEls = m_support[0]*m_support[1];
    res.create(nrEls,nrChannels);
    cv::Mat_<float> res_col(1,nrChannels);
    std::vector<cv::SparseMat_<float> > tmpM(size(1));
    int sizeTmp[] = {size(0)};
    for (int cdx = 0; cdx<size(1); cdx++)
        tmpM[cdx].create(1,sizeTmp);
    for (cv::SparseMatConstIterator_<float> it = begin(); it != end(); ++it)
        (tmpM[it.node()->idx[1]]).ref(it.node()->idx[0])=*it;
    for (int cdx = 0; cdx<size(1); cdx++) {
        m_chBasis->decode(res_col,tmpM[cdx],nrModes);
        res_col.copyTo(res(cv::Range(cdx,cdx+1),cv::Range::all()));
    }
    res.reshape(nrChannels,m_support[0]);}
```

Extra modules

Possibility to add new modules
without putting them into the OpenCV tree:

```
opencv/
   modules/
      core/
         include/, doc/, src/, test/, …
         CMakeLists.txt
      imgproc
      …

my_extra_modules/
   sfm/
     include/, doc/, src/, test/, …
     CMakeLists.txt
   …
```

Experimental or proprietary code.

```
$ cmake -D OPENCV_EXTRA_MODULES_PATH=~/my_extra_modules …
```