#pragma once

#ifdef \_\_cplusplus

#include <twml/optim.h>

namespace twml {

enum InterpolationMode {LINEAR, NEAREST};

template<typename Tx, typename Ty>

static Tx interpolation(const Tx \*xsData, const int64\_t xsStride,

const Ty \*ysData, const int64\_t ysStride,

const Tx val, const int64\_t mainSize,

const InterpolationMode mode,

const int64\_t lowest,

const bool return\_local\_index = false) {

int64\_t left = 0;

int64\_t right = mainSize-1;

if (val <= xsData[0]) {

right = 0;

} else if (val >= xsData[right\*xsStride]) {

left = right;

} else {

while (left < right) {

int64\_t middle = (left+right)/2;

if (middle < mainSize - 1 &&

val >= xsData[middle\*xsStride] &&

val <= xsData[(middle+1)\*xsStride]) {

left = middle;

right = middle + 1;

break;

} else if (val > xsData[middle\*xsStride]) {

left = middle;

} else {

right = middle;

}

}

if (lowest) {

while (left > 0 &&

val >= xsData[(left - 1) \* xsStride] &&

val == xsData[left \* xsStride]) {

left--;

right--;

}

}

}

Ty out = 0;

if (return\_local\_index) {

out = left;

} else if (mode == NEAREST) {

out = ysData[left\*ysStride];

} else {

int64\_t leftys = left\*ysStride;

int64\_t rightys = right\*ysStride;

int64\_t leftxs = left\*xsStride;

int64\_t rightxs = right\*xsStride;

if (right != left+1 ||

xsData[leftxs] == xsData[rightxs]) {

out = ysData[leftys];

} else {

Tx xLeft = xsData[leftxs];

Tx xRight = xsData[rightxs];

Tx yLeft = ysData[leftys];

Tx ratio = (val - xLeft) / (xRight - xLeft);

out = ratio\*(ysData[rightys] - yLeft) + yLeft;

}

}

return out;

}

} // namespace twml

#endif