spanny

abusing mdspan is within arms reach

October 4, 2023

Griswald Brooks
Senior Software Engineer
griswald.brooks@picknik.ai

mdspan

```
// Allocate the map
std::array map_storage{...};
// Create a 2D view of the map
auto occupancy_map = std::mdspan(map_storage.data(), 384, 384);
// Update the map
for (std::size_t i = 0; i != occupancy_map.extent(0); i++) {
    for (std::size_t j = 0; j != occupancy_map.extent(1); j++) {
        occupancy_map[i, j] = get_occupancy_from_laser_scan(...)
    }
}
// TODO: Add turtlebot map
```

github.com/griswaldbrooks/spanny

Policy Injection

```
template <
  class T,
  class Extents,
  class LayoutPolicy = std::layout_right,
  class Accessor = std::default_accessor
>
class mdspan;
```

github.com/griswaldbrooks/spanny

Bin Accessor Policy

```
template <int nbins>
struct bin_checker {
  using element_type = tl::expected<bin_state, std::string>;
  using reference = tl::expected<bin_state, std::string> const&;
  using data_handle_type = robot_arm*;

  inline static element_type const no_bin = tl::make_unexpected("Invalid bin");
  mutable element_type recent_ = tl::make_unexpected("Bin has not been accessed");

  reference access(data_handle_type arm, std::ptrdiff_t offset) const {
    // Return an error if the offset is out of bounds
    if (offset < 0 || offset >= nbins) return no_bin;
    // Set the bin state in a member of the accessor policy
    recent_ = arm->is_bin_occupied(static_cast<int>(offset));
    // This is because we return a reference
    return recent_;
  }
};
```

Getting Started

```
using ext_t = stdex::extents<uint32_t, 2, 3>;
using acc_t = bin_checker<6>;
using bin_view = stdex::mdspan<bin_state, ext_t, stdex::layout_right, acc_t>;

int main(int, char **) {
  auto arm = robot_arm{"/dev/ttyACM0", 9600};
  auto bins = bin_view(&arm, {}, bin_checker<6>{});
  while(true) {
    for (std::size_t i = 0; i != bins.extent(0); ++i) {
      for (std::size_t j = 0; j != bins.extent(1); ++j) {
        std::cout << "Bin " << i << ", " << j << " is ";
        bins(i, j).and_then(print_state).or_else(shrug);
        std::cout << "\n";
      }
    }
    std::cout << "====================\n";
  }
  return 0;
}
```

github.com/griswaldbrooks/spanny

Demo Time

github.com/griswaldbrooks/spanny

Thank you!

github.com/griswaldbrooks/spanny