Task: Write a MATLAB function that implements a n-object tracker using the global nearest neighbour algorithm.

Your implementation (for each filtering recursion) should follow these steps:

1. implement ellipsoidal gating for each predicted local hypothesis seperately, see *Note* below for details;
2. construct 2D cost matrix of size (number of objects, number of measurements that at least fall inside the gates + number of objects);
3. find the best assignment matrix using a 2D assignment solver;
4. create new local hypotheses according to the best assignment matrix obtained;
5. extract object state estimates;
6. predict each local hypothesis.

The 2D assignment solver has been provided as a reference function.

```
[col4row,row4col,gain]=assign2D(C)
```

`ASSIGN2D` solves the two-dimensional assignment (minimization) problem with a rectangular cost matrix `C`.

INPUT: `C`: A `numRowXnumCol` cost matrix.

OUTPUT: `col4row`: A numRowX1 vector where the entry in each element is an assignment of the element in that row to a column. 0 entries signify unassigned rows. If the problem is infeasible, this is an empty matrix.

`row4col`: A `numColX1` vector where the entry in each element is an assignment of the element in that column to a row. 0 entries signify unassigned columns. If the problem is infeasible, this is an empty matrix.

`gain`: The sum of the values of the assigned elements in `C`. If the problem is infeasible, this is -1.

Note:

1. When constructing your 2D cost matrix, if measurement j does not fall inside the gate of object i, set the corresponding entry (i, j) to $+\infty$.
2. Remove columns in the cost matrix with only $+\infty$ entries. In other words, we do not consider measurements that do not fall inside any object gates.
3. The third output argument `gain` should never be -1 if the implementation is correct.
4. We can use gating to further group objects into subsets and process each subset independently. However, this is **NOT** covered in this task.

Hint: If you want to apply a function to each element of a struct array, you can use MATLAB command `arrayfun`, which makes your implementation faster than using `for` loops.

Files referenced:

- *`modelgen.m`*
- *`GaussianDensity.m`*
- *`hypothesisReduction.m`*
- *`motionmodel.m`*
- *`normalizeLogWeights.m`*
- *`log_mvnpdf.m`*

- *`assign2D.m`*

Note that `obj.density` is a function handle bound to MATLAB class `GaussianDensity`. For instance, you can simply call `obj.density.update` to perform a Kalman update instead of using your own code.