Slide 1: Hi my name is Mattias Heinrich. I am a lecturer at the University of Lübeck and for this MICCAI Educational Challenge I would like to introduce you to the concept of discrete optimisation and how we can use it for medical image registration. Let me start by first motivating why we are doing medical image registration.

Slide 2: Image registration is useful for a number of important clinical applications, including: the fusion of images acquired with different modalities, for image guided surgery. To compensate motion for adaptive radiotherapy e.g. using the cyberknife. For this we need to estimate the respiratory motion of a patient using 4DCT. And it can be used to automatically segment images using expert annotations of training data and atlas-based segmentation. For all these tasks of image registration, we need to establish correspondences across scans.

Slide 3: These correspondences should match related anatomical locations in the fixed and moving image. A dense correspondence field is used to define a transformation consisting of an identity (transform) and a displacement field u(x). This can be done automatically, by optimising a cost function, which usually consists of at least two terms: a similarity and regularisation term.
Let's have a look at this exemplary cost function plotted against a transformation parameter. We can find a minimum by using continuous optimisation. For this we initialise the parameter value, calculate the derivative of the cost function and follow downwards along the gradient until we reach a minimum after a number of iterations. In this case it only leads to a local minimum.
With discrete optimisation in contrast, we can sample a range of quantised transformation parameters and optimise over all of them simultaneously. Therefore, we can avoid the local minimum and find a solution very close to the global optimum. A discrete optimisation problem can be solved using a graphical model (e.g. a Markov random field).

Slide 4: A graphical model, consists of a number of nodes, which correspond in our case to voxels or control points. I have called them vp, vq, vr, and so on and are drawing some examples. We introduce edges, which connect nodes and are drawn with green lines. They model the interactions between nodes. Thirdly, we have a set of labels (li,lj,lk,..) which correspond to 3D displacement vectors. The aim is now, to find the optimal labelling which assigns a displacement label to each node. The quality of the labelling is evaluated using a cost function, which consists of two terms: R(fp,fq) is called the pair-wise potential and D(fp) - the unary term.

Slide 5: The unary term usually models the similarity between the two images. Since, it's independent for each voxel or control point, we have great control over the space of potential displacements. We can use an sampling along the axes only, a planar sampling or evaluate every possible 3D displacement within a cube of a certain search range. The animation on the left shows the calculation of similarity between the red block in the fixed image and a green block which is shifted densely over the moving image. Because, we don't require a derivate of the similarity term, we can use any point- or patch-wise metric: e.g. sum of absolute differences, normalised cross correlation, or the self-similarity context. This yields us a 3D similarity distribution (as shown on the left) for every control point. But so far we have not included any regularisation.

Slide 6: This means a resulting displacement field, which is shown here in a specific colour coding for a 4DCT registration, contains many implausible irregularities. Such as folds in the transformation or unrealistic local volume change. So we would like to use the pair-wise potentials to include a penalty for deviations of the displacement of connected nodes. A potential choice would be total variation, which is based on the absolute difference of two displacements. It's penalty function looks as follows. Or we can employ diffusion regularisation, which penalises the squared difference of displacements. If our algorithm consists of multiple steps it's always important to perform an incremental regularisation. For general graphs this optimisation is unfortunately NP-hard and its solution can only approximated by using e.g. graph cuts or message passing.
However, for a certain relaxed graph structure, which contains no loops, namely a tree, we can find the exact solution in linear time using dynamic programming.

Slide 7: I've prepared a very simple example consisting of only 4 connected nodes, which can take one of the five displacements between minus and plus 2. We use total variation as regularisation penalty and the unary potentials are represented by the height of these 3D bars. We aim to find the path with minimal cost across these obstacles. To make this more accessible I've written down the unary potentials in an array and we'll work our way from the green node up to the blue node.
Let's first try this specific path, which always takes the fifth label. It has unary potentials, which add up to 8.5. A second path, which goes diagonally from 5→4→3→2 has a total unary cost of 4 plus a regularisation penalty of 3. We see that by increasing the number of nodes the potential solutions increase exponentially and no useful problem could be solved using a brute-force approach. We therefore explore a recursive solution on the next slide.

Slide 8: To work out the recursive solution, we need to define two additional matrices: one which stores the cost of any optimal path up to a certain point and another one which stores the indices of the labels which got us there. The equation for the recursive formula calculates a message vector by minimising over the unary potentials plus the message pointing to this node and all possible pair-wise combinations of the regularisation term. The green node has no previous dependencies, so we can simply copy the unary potentials here. Let's evaluate the message for the third label of the yellow node. There are five potential paths leading here and we have to calculate the cost for each of them. The first one has a cost of 0.5+2, the second one 1+1, the third one has only a unary cost which is 3, and the other ones both result to 3.5.
Once, we're finished we simply select the minimum and write down the argmin (the label the path originated from at the green node) into the respective box of the index matrix. The cost matrix stores the sum of unaries and messages so we sum up 1.5 and 2.0 and write 3.5 into the yellow box. We repeat the process for the remaining labels of the yellow node and fill out the optimal costs and label indices. We can now perform the message calculations following the same principal for the red — and then blue node. And since the blue node doesn't have any further dependencies, we simply select the minimum with a value of 6.5 and the label 2. To find the optimal path we now need to trace backwards through our index matrix and select the second label of the red row (3), the third label of the yellow row (3) and again the third label of the green row (2). This leads to the path 2-3-3-2 and we can easily check that it's cost is equal to 6.5. To speed up the message computations for commonly used regularisation penalties, we can use the fast distance transforms introduced by Pedro Felszenswalb.

Next, I'd like to show a real-time example of this approach using our software framework deeds.

Slide 9/10: This is a live demonstration of the software package deeds (short for dense displacement sampling) that I have developed during my DPhil at Oxford. I first start running the algorithm in the terminal window and explain the graphical user interface, in the meantime. In the first two rows you can specify input and output images, here I have selected the inhale and exhale phase of a 4DCT scan of the DIRlab dataset. Next we can set the parameters for the control point spacing and the search space. The framework uses 5 scales in this example, so we have to set 5 values each. Finally, the similarity metric and its relative weighting are chosen: here I have used the self-similarity context.
Now the registration has finished in shortly over 30 seconds, and we can look at the visual outcome it generated using animated gifs: The top row shows the overlay of the images before registration using green and magenta in an axial fly-through. Below we see how the differences almost entirely disappear after our deformable registration. On the right hand side the magnitude of the estimated deformations is shown in a false-colour coding for a coronal view, it can be seen that the sliding motion of the lungs is well preserved.

Slide 11: Finally, I would like to summarise the key advantages of discrete optimisation: First it does not require a derivative of the cost function, which makes it very flexible. Second, we can approximately find a global optimum and third if we use an efficient implementation we can achieve a very fast registration.
There are many active topics of research in this area: the use of advanced transformation models, e.g. curvature regularisation and linear or symmetric transforms; exploiting uncertainty estimates: for a dynamic search range refinement or to improve segmentation propagation; the estimation of extra parameters in addition to the geometric displacements e.g. a segmentation label or functional imaging parameters; and finally the use of image adaptive non-grid graphs, such as supervoxels or finite element meshes.
For further information about this mini-lecture you can have a look at the website: mpheinrich.de/mec.html where you'll find source code, additional references and a transcript