Because γ_1, γ_2, and γ_3 are unit vectors, one can represent γ_1 and γ_2 in spherical coordinate system:

$$\gamma_1 = (\sin\alpha\cos\eta, \sin\alpha\sin\eta, \cos\alpha)^T$$

$$\gamma_2 = (\sin\psi\cos\delta, \sin\psi\sin\delta, \cos\psi)^T$$

So γ_1 is uniquely defined using co-latitude α and longitude η. γ_2 is uniquely defined using co-latitude ψ and longitude δ. However, because γ_1 is perpendicular to γ_2, their dot product is zero and this results in a dependent variable. If α, η, ψ are independent variables, then δ is given as follows:

$$\gamma_1^T\gamma_2 = \sin\alpha\sin\psi\cos(\delta-\eta) + \cos\alpha\cos\psi = 0$$

$$\text{Therefore,} \quad \cos(\delta-\eta) = -\cot\alpha\cot\psi$$

The above equation imposes a constraint on the range of values δ can take. More importantly,

$$|\cot\alpha\cot\psi| \leq 1 \tag{0.1}$$

γ_1, γ_2 require three parameters to be uniquely identified. γ_3 is computed because it is orthogonal to both γ_1 and γ_2. In the event the inequality 0.1 is not satisfied, the program crashes because $\cos(\delta-\eta)$ will be greater than 1.

Problem with *dlib*: From my understanding, the library cannot support constraints of this kind (0.1). So if I simply run an unconstrained optimization, in the course of optimizing the objective function, there are cases where 0.1 (using the intermediate solution) does not hold and this creates a problem. In certain instances, the program runs through smoothly but there are quite a number of cases where the program fails.

So if there is a library which can facilitate the programming of such kind of constraints, that would potentially solve the problem.